Preface

EUROCRYPT 2000, the nineteenth annual Eurocrypt Conference, was sponsored by the International Association for Cryptologic Research (IACR), in cooperation with the Katholieke Universiteit Leuven in Belgium (research group for Computer Security and Industrial Cryptography, COSIC).

The first conference with the name 'Eurocrypt' took place in 1983, but the 1982 Workshop at Burg Feuerstein was the first open meeting in Europe on cryptology; it has been included in Lecture Notes in Computer Science 1440, which contains an electronic proceedings and index of the Crypto and Eurocrypt conferences 1981–1997.

The program committee considered 150 papers and selected 39 for presentation. One paper was withdrawn by the authors. The program also included invited talks by Michael Walker ("On the Security of 3GPP Networks") and Tony Sale ("Colossus and the German Lorenz Cipher – Code Breaking in WW II"). In addition, Andy Clark kindly agreed to chair the traditional rump session for informal presentations of recent results.

The selection of the program was a challenging task, as many high quality submissions were received. Each submission was reviewed by at least three reviewers and most reports had four or more reviews (papers with program committee members as a co-author had at least six reviews). The program committee worked very hard to evaluate the papers with respect to quality, originality, and relevance to cryptology. In most cases they were able to provide extensive comments to the authors (about half a Megabyte of comments for authors has been written). Subsequently, the authors of accepted papers have made a substantial effort to take into account the comments in the version submitted to these proceedings. In a limited number of cases, these revisions have been checked by members of the program committee.

First and foremost I would like to thank the members of the program committee for the many hours spent on reviewing and discussing the papers, and for helping me with the difficult decisions.

I gratefully acknowledge the help of a large number of colleagues who reviewed submissions in their area of expertise: Masayuki Abe, N. Asokan, Olivier Baudron, Josh Benaloh, Eli Biham, Simon Blake-Wilson, Johan Borst, Emmanuel Bresson, Jan Camenisch, Ivan Damgård, Anand Desai, Yvo Desmedt, Glenn Durfee, Serge Fehr, Matthias Fitzi, Pierre-Alain Fouque, Matt Franklin, Steven Galbraith, Juan A. Garay, Louis Granboulan, Stuart Haber, Shai Halevi, Martin Hirt, Fredrik Jönsson, Mike Jacobson, Jens G. Jensen, Ari Juels, Jonathan Katz, Robert Lambert, Julio Lopez Hernandez, Phil MacKenzie, Julien Marcil, Willi Meier, Preda Mihailescu, Serge Mister, Fabian Monrose, Sean Murphy, Siaw-Lynn Ng, Phong Nguyen, Valtteri Niemi, Tatsuaki Okamoto, Thomas Pornin, Guillaume Poupard, Bartek Przydatek, Omer Reingold, Vincent Rijmen, Louis Salvail, Tomas Sander, Berry Schoenmakers, Dan Simon, Ben Smeets,

Michael Steiner, Jacques Stern, Martin Strauss, Katsuyuki Takashima, Edlyn Teske, Barry Trager, Ramarathnam Venkatesan, Frederik Vercauteren, Susanne Wetzel, Mike Wiener, Peter Wild, Adam Young. I apologise for any inadvertent omissions.

By now, electronic submissions have become a tradition for Eurocrypt. I would like to thank Joe Kilian, who did an excellent job in running the electronic submission server of ACM's SIGACT group. Only five contributions were submitted in paper form; for three of these, I obtained an electronic copy from the authors. The remaining two papers were scanned in to make the process uniform to reviewers. As a first for IACR sponsored conferences, we developed a web interface for entering reviews and discussing papers. Special thanks go to Joris Claessens and Wim Moreau who spent several weeks developing my rough specifications into a flawless program with a smooth user interface. This work made the job of the program committee much easier, as we could focus on the content of the discussion rather than on its organisation. This software will be made available to all IACR sponsored conferences.

My ability to run the program committee was increased substantially by the effort and skills provided by the members of COSIC: Vincent Rijmen put together the LaTeX version of the proceedings, Joris Claessens helped with processing the submissions, Johan Borst converted a paper to LaTeX, Péla Noë assisted with organising the program committee meeting, and (last but not least) Wim Moreau helped with the electronic processing of the submissions and final versions, and with the copyright forms.

I would like to thank Joos Vandewalle, general chair, the members of the organising committee (Joris Claessens, Danny De Cock, Erik De Win, Marijke De Soete, Keith Martin, Wim Moreau, Péla Noë, Jean-Jacques Quisquater, Vincent Rijmen, Bart Van Rompay, Karel Wouters), and the other members of COSIC for their support. I also thank Elvira Wouters, who took care of the accounting, and Anne De Smet (Momentum), who was responsible for the hotel bookings and the social program. For the first time, the registrations of Eurocrypt were handled by the IACR General Secretariat in Santa Barbara (UCSB); I would like to thank Micky Swick and Sally Vito for the successful collaboration. The organising committee gratefully acknowledges the financial contributions of our sponsors: Isabel, Ubizen, Europay International, Cryptomathic Belgium, PriceWaterhouseCoopers, Utimaco, and the Katholieke Universiteit Leuven.

Finally, I wish to thank all the authors who submitted papers, making this conference possible, and the authors of accepted papers for their cooperation. Special thanks go to Alfred Hofmann and his colleagues at Springer-Verlag for the timely production of this volume.

March 2000

Bart Preneel
Program Chair
Eurocrypt 2000

EUROCRYPT 2000

May 14–18, 2000, Bruges, Belgium

Sponsored by the

International Association for Cryptologic Research

General Chair

Joos Vandewalle, Katholieke Universiteit Leuven, Belgium

Program Chair

Bart Preneel, Katholieke Universiteit Leuven, Belgium

Program Committee

Simon Blackburn ..... Royal Holloway Univ. of London, UK
Dan Boneh ..... Stanford Univ., USA
Christian Cachin ..... IBM Research, Switzerland
Don Coppersmith ..... IBM Research, USA
Ronald Cramer ..... ETH Zurich, Switzerland
Hans Dobbertin ..... BSI, Germany
Markus Jakobsson ..... Bell Laboratories, USA
Thomas Johansson ..... Lund Univ., Sweden
Joe Kilian ..... NEC Research Institute, USA
Lars Knudsen ..... Univ. of Bergen, Norway
Mitsuru Matsui ..... Mitsubishi, Japan
Alfred Menezes ..... Univ. of Waterloo, Canada
Moni Naor ..... Weizmann Institute of Science, Israel
Kaisa Nyberg ..... Nokia Research Center, Finland
Paul van Oorschot ..... Entrust Technologies, Canada
Torben Pedersen ..... Cryptomathic, Denmark
David Pointcheval ..... ENS, France
Moti Yung ..... Certco, USA

Table of Contents

Factoring and Discrete Logarithm

Factorization of a 512–bit RSA Modulus 1
Stefania Cavallar (CWI, The Netherlands), Bruce Dodson (Lehigh University, USA), Arjen K. Lenstra (Citibank, USA), Walter Lioen (CWI, The Netherlands), Peter L. Montgomery (Microsoft Research, USA and CWI, The Netherlands), Brian Murphy (Computer Sciences Laboratory, Australia), Herman te Riele (CWI, The Netherlands), Karen Aardal (Utrecht University, The Netherlands), Jeff Gilchrist (Entrust Technologies Ltd., Canada), Gérard Guillerm (École Polytechnique, France), Paul Leyland (Microsoft Research Ltd., UK), Joël Marchand (École Polytechnique/CNRS, France), François Morain (École Polytechnique, France), Alec Muffett (Sun Microsystems Professional Services, UK), Chris and Craig Putnam (USA), Paul Zimmermann (Inria Lorraine and Loria, France)

An Algorithm for Solving the Discrete Log Problem on Hyperelliptic Curves 19
Pierrick Gaudry (École Polytechnique, France)

Analysis and Optimization of the TWINKLE Factoring Device 35
Arjen K. Lenstra (Citibank, USA), Adi Shamir (The Weizmann Institute, Israel)

Cryptanalysis I: Digital Signatures

Noisy Polynomial Interpolation and Noisy Chinese Remaindering 53
Daniel Bleichenbacher (Bell Laboratories, USA), Phong Q. Nguyen (École Normale Supérieure, France)

A Chosen Messages Attack on the ISO/IEC 9796–1 Signature Scheme 70
François Grieu (Innovatron, France)

Cryptanalysis of Countermeasures Proposed for Repairing ISO 9796-1 81
Marc Girault, Jean-François Misarsky (France Télécom - CNET, France)

Security Analysis of the Gennaro-Halevi-Rabin Signature Scheme 91
Jean-Sébastien Coron (École Normale Supérieure, France), David Naccache (Gemplus Card International, France)

Invited Talk

On the Security of 3GPP Networks 102
Michael Walker (Vodafone and Royal Holloway College, University of London, UK)

Private Information Retrieval

One-Way Trapdoor Permutations Are Sufficient for Non-trivial Single-Server Private Information Retrieval 104
Eyal Kushilevitz (IBM T.J. Watson Research Center, USA), Rafail Ostrovsky (Telcordia Technologies Inc., USA)

Single Database Private Information Retrieval Implies Oblivious Transfer 122
Giovanni Di Crescenzo (Telcordia Technologies, Inc., USA), Tal Malkin (Massachusetts Institute of Technology and AT&T Labs Research), Rafail Ostrovsky (Telcordia Technologies, Inc., USA)

Key Management Protocols

Authenticated Key Exchange Secure Against Dictionary Attacks 140
Mihir Bellare (University of California at San Diego, USA), David Pointcheval (École Normale Supérieure, France), Phillip Rogaway (University of California at Davis, USA)

Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman 157
Victor Boyko (Massachusetts Institute of Technology, USA), Philip MacKenzie (Bell Laboratories, USA), Sarvar Patel (Bell Laboratories, USA)

Fair Encryption of RSA Keys 173
Guillaume Poupard, Jacques Stern (École Normale Supérieure, France)

Threshold Cryptography and Digital Signatures

Computing Inverses over a Shared Secret Modulus 191
Dario Catalano (Università di Catania, Italy), Rosario Gennaro (IBM T.J.Watson Research Center, USA), Shai Halevi (IBM T.J.Watson Research Center, USA)

Practical Threshold Signatures 209
Victor Shoup (IBM Zurich Research Laboratory, Switzerland)

Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures 223
Stanisław Jarecki, Anna Lysyanskaya (Massachusetts Institute of Technology, USA)

Confirmer Signature Schemes Secure against Adaptive Adversaries 246
Jan Camenisch (IBM Zurich Research Laboratory, Switzerland), Markus Michels (Entrust Technologies, Switzerland)

Public-Key Encryption

Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements 262
Mihir Bellare (University of California at San Diego, USA), Alexandra Boldyreva (University of California at San Diego, USA), Silvio Micali (Massachusetts Institute of Technology, USA)

Using Hash Functions as a Hedge against Chosen Ciphertext Attack 279
Victor Shoup (IBM Zurich Research Laboratory, Switzerland)

Quantum Cryptography

Security Aspects of Practical Quantum Cryptography 294
Gilles Brassard (Université de Montréal, Canada), Norbert Lütkenhaus (Helsinki Institute of Physics, Finland), Tal Mor (University of California at Los Angeles, USA and College of Judea and Samaria, Israel), Barry C. Sanders (Macquarie University, Australia)

Perfectly Concealing Quantum Bit Commitment from any Quantum One-Way Permutation 305
Paul Dumais (Université de Montréal, Canada), Dominic Mayers (NEC Research Institute, USA), Louis Salvail (BRICS, University of Århus, Denmark)

Multi-Party Computation and Information Theory

General Secure Multi-Party Computation from any Linear Secret-Sharing Scheme 321
Ronald Cramer (BRICS, Aarhus University, Denmark), Ivan Damgård (BRICS, Aarhus University, Denmark), Ueli Maurer (ETH Zurich, Switzerland)

Minimal-Latency Secure Function Evaluation 340
Donald Beaver (CertCo Inc., USA)

Information-Theoretic Key Agreement: From Weak to Strong Secrecy for Free 356
Ueli Maurer, Stefan Wolf (ETH Zurich, Switzerland)

Cryptanalysis II: Public-Key Encryption

New Attacks on PKCS#1 v1.5 Encryption 374
Jean-Sébastien Coron (École Normale Supérieure and Gemplus Card International, France), Marc Joye (Gemplus Card International, France),

David Naccache (Gemplus Card International, France), Pascal Paillier (Gemplus Card International, France)

A NICE Cryptanalysis 388
Éliane Jaulmes, Antoine Joux (SCSSI, France)

Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations 398
Nicolas Courtois (Toulon University and Bull CP8, France), Alexander Klimov (Moscow State University, Russia), Jacques Patarin (Bull CP8, France), Adi Shamir (The Weizmann Institute of Science, Israel)

Cryptanalysis of Patarin's 2-Round Public Key System with S Boxes (2R) 414
Eli Biham (Technion, Israel)

Invited Talk

Colossus and the German Lorenz Cipher – Code Breaking in WW II 423
Anthony E Sale (Bletchley Park Trust)

Zero-Knowledge

Efficient Concurrent Zero-Knowledge in the Auxiliary String Model 424
Ivan Damgård (BRICS, Aarhus University, Denmark)

Efficient Proofs that a Committed Number Lies in an Interval 437
Fabrice Boudot (France Télécom - CNET, France)

Symmetric Cryptography

A Composition Theorem for Universal One-Way Hash Functions 451
Victor Shoup (IBM Zurich Research Laboratory, Switzerland)

Exposure-Resilient Functions and All-Or-Nothing Transforms 459
Ran Canetti (IBM T.J. Watson Research Center, USA), Yevgeniy Dodis (Massachusetts Institute of Technology, USA), Shai Halevi (IBM T.J. Watson Research Center, USA), Eyal Kushilevitz (IBM T.J. Watson Research Center, USA and Technion, Israel), Amit Sahai (Massachusetts Institute of Technology, USA)

The Sum of PRPs is a Secure PRF 476
Stefan Lucks (Universität Mannheim, Germany)

Boolean Functions and Hardware

Construction of Nonlinear Boolean Functions with Important Cryptographic Properties 491
Palash Sarkar, Subhamoy Maitra (Indian Statistical Institute, India)

Propagation Characteristics and Correlation-Immunity of Highly Nonlinear Boolean Functions 513
Anne Canteaut (INRIA, France), Claude Carlet (Université de Caen, France), Pascale Charpin (NRIA, France), Caroline Fontaine (Université des Sciences et Technologies de Lille, France)

Cox-Rower Architecture for Fast Parallel Montgomery Multiplication 529
Shinichi Kawamura, Masanobu Koike, Fumihiko Sano, Atsushi Shimbo (Toshiba Corporation, Japan)

Voting Schemes

Efficient Receipt-Free Voting Based on Homomorphic Encryption 545
Martin Hirt (ETH Zurich, Switzerland), Kazue Sako (NEC Corporation, Japan)

How to Break a Practical MIX and Design a New One 563
Yvo Desmedt (Florida State University, USA and Royal Holloway, University of London, UK), Kaoru Kurosawa (Tokyo Institute of Technology, Japan)

Cryptanalysis III: Stream Ciphers and Block Ciphers

Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5 579
Anne Canteaut (INRIA, France), Michaël Trabbia (INRIA and École Polytechnique, France)

Advanced Slide Attacks 595
Alex Biryukov (Technion, Israel), David Wagner (University of California at Berkeley, USA)

Author Index 613

Factorization of a 512–bit RSA Modulus *

Stefania Cavallar[3], Bruce Dodson[8], Arjen K. Lenstra[1], Walter Lioen[3], Peter L. Montgomery[10], Brian Murphy[2], Herman te Riele[3], Karen Aardal[13], Jeff Gilchrist[4], Gérard Guillerm[11], Paul Leyland[9], Joël Marchand[5], François Morain[6], Alec Muffett[12], Chris and Craig Putnam[14], and Paul Zimmermann[7]

[1] Citibank, 1 North Gate Road, Mendham, NJ 07945–3104, USA
arjen.lenstra@citicorp.com
[2] Computer Sciences Laboratory, ANU, Canberra ACT 0200, Australia
murphy@cslab.anu.edu.au
[3] CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
{cavallar,walter,herman}@cwi.nl
[4] Entrust Technologies Ltd., 750 Heron Road, Suite E08, Ottawa, ON, K1V 1A7, Canada
Jeff.Gilchrist@entrust.com
[5] Laboratoire Gage, École Polytechnique/CNRS, Palaiseau, France
Joel.Marchand@medicis.polytechnique.fr
[6] Laboratoire d'Informatique, École Polytechnique, Palaiseau, France
morain@lix.polytechnique.fr
[7] Inria Lorraine and Loria, Nancy, France
Paul.Zimmermann@loria.fr
[8] Lehigh University, Bethlehem, PA, USA
bad0@Lehigh.edu
[9] Microsoft Research Ltd, Cambridge, UK
pleyland@microsoft.com
[10] 780 Las Colindas Road, San Rafael, CA 94903–2346 USA
Microsoft Research and CWI
pmontgom@cwi.nl
[11] SITX (Centre of IT resources), École Polytechnique, Palaiseau, France
Gerard.Guillerm@polytechnique.fr
[12] Sun Microsystems, Riverside Way, Watchmoor Park, Camberley, UK
alec.muffett@uk.sun.com
[13] Dept. of Computer Science, Utrecht University,
P.O. Box 80089, 3508 TB Utrecht, The Netherlands
aardal@cs.uu.nl
[14] 59 Rangers Dr., Hudson, NH 03051, USA
craig.putnam@swift.mv.com

* Breakdown of individual contributions to this project:
Management: Te Riele; *polynomial selection algorithm*: Montgomery, Murphy; *polynomial selection computations*: Dodson, Lenstra, Montgomery, Murphy; *sieving codes*: Lenstra, Montgomery; *sieving*: Aardal, Cavallar, Dodson, Gilchrist, Guillerm, Lenstra, Leyland, Lioen, Marchand, Montgomery, Morain, Muffett, Putnam, Zimmermann; *filtering*: Cavallar, Montgomery; *linear algebra*: Leyland, Montgomery; *square root*: Montgomery; *data collection, analysis of data and running the NFS code at CWI and SARA*: Cavallar, Lioen, Montgomery; *technical support*: Lioen.
This is a slightly abridged version of the paper which was originally submitted to Eurocrypt 2000: http://www.cwi.nl/~herman/RSA155/EuCr2000orig.ps .

Abstract. This paper reports on the factorization of the 512–bit number RSA–155 by the Number Field Sieve factoring method (NFS) and discusses the implications for RSA.

1 Introduction

On August 22, 1999, we completed the factorization of the 512–bit 155–digit number RSA–155 by NFS. The number RSA–155 was taken from the RSA Challenge list [34] as a representative 512–bit RSA modulus. Our result is a new record for factoring general integers. Because 512–bit RSA keys are frequently used for the protection of electronic commerce—at least outside the USA—this factorization represents a breakthrough in research on RSA–based systems.

The previous record, factoring the 140–digit number RSA–140 [8], was established on February 2, 1999, also with the help of NFS, by a subset of the team which factored RSA–155. The amount of computing time spent on RSA–155 was about 8400 MIPS years[1], roughly four times that needed for RSA–140; this is about half of what could be expected from a straightforward extrapolation of the computing time spent on factoring RSA–140 and about a quarter of what would be expected from a straightforward extrapolation of the computing time spent on RSA–130 [11]. The speed-up is due to a new polynomial selection method for NFS of Murphy and Montgomery which was applied for the first time to RSA–140 and now, with improvements, to RSA–155.

Section 2 discusses the implications of this project for the practical use of RSA–based cryptosystems. Section 3 has the details of our computations which resulted in the factorization of RSA–155.

2 Implications for the practice of RSA

RSA is widely used today [17]. The best size for an RSA key depends on the security needs of the user and on how long his/her information needs to be protected.

The amount of CPU time spent to factor RSA–155 was about 8400 MIPS years, which is about four times that used for the factorization of RSA–140. On the basis of the heuristic complexity formula [7] for factoring large N by NFS:

$$\exp\left((1.923 + o(1))\,(\log N)^{1/3}(\log\log N)^{2/3}\right), \tag{1}$$

one would expect an increase in the computing time by a factor of about seven.[2] This speed-up has been made possible by algorithmic improvements, mainly in

[1] One *MIPS year* is the equivalent of a computation during one full year at a sustained speed of one **M**illion **I**nstructions **P**er **S**econd.

[2] By "computing time" we mean the *sieve* time, which dominates the total amount of CPU time for NFS. However, there is a trade-off between polynomial search time and sieve time which indicates that a non-trivial part of the total amount of computing time should be spent to the polynomial search time in order to minimize the sieve

the polynomial generation step [26, 29, 30], and to a lesser extent in the filter step of NFS [9].

The complete project to factor RSA–155 took seven calendar months. The polynomial generation step took about one month on several fast workstations. The most time-consuming step, the sieving, was done on about 300 fast PCs and workstations spread over twelve "sites" in six countries. This step took 3.7 calendar months, in which, summed over all these 300 computers, a total of 35.7 years of CPU-time was consumed. Filtering the relations and building and reducing the matrix corresponding to these relations took one calendar month and was carried out on an SGI Origin 2000 computer. The block Lanczos step to find dependencies in this matrix took about ten calendar days on one CPU of a Cray C916 supercomputer. The final square root step took about two days calendar time on an SGI Origin 2000 computer.

Based on our experience with factoring large numbers we estimate that within three years the algorithmic and computer technology which we used to factor RSA–155 will be widespread, at least in the scientific world, so that by then 512–bit RSA keys will certainly not be safe any more. This makes these keys useless for authentication or for the protection of data required to be secure for a period longer than a few days.

512–bit RSA keys protect 95% of today's E-commerce on the Internet [35]—at least outside the USA—and are used in SSL (Secure Socket Layer) handshake protocols. Underlying this undesirable situation are the old export restrictions imposed by the USA government on products and applications using "strong" cryptography like RSA. However, on January 12, 2000, the U.S. Department of Commerce Bureau of Export Administration (BXA) issued new encryption export regulations which allow U.S. companies to use larger than 512–bit keys in RSA–based products [38]. As a result, one may replace 512–bit keys by 768–bit or even 1024–bit keys thus creating much more favorable conditions for secure Internet communication.

In order to attempt an extrapolation, we give a table of factoring records starting with the landmark factorization in 1970 by Morrison and Brillhart of $F_7 = 2^{128} + 1$ with help of the then new Continued Fraction (CF) method. This table includes the complete list of factored RSA–numbers, although RSA–100 and RSA–110 were not absolute records at the time they were factored. Notice that RSA–150 is still open. Some details on recent factoring records are given in Appendix A to this paper.

Based on this table and on the factoring algorithms which we currently know, we anticipate that within ten years from now 768–bit (232–digit) RSA keys will become unsafe.

Let D be the number of decimal digits in the largest "general" number factored by a given date. From the complexity formula for NFS (1), assuming Moore's law (computing power doubles every 18 months), Brent [5] expects $D^{1/3}$

time. See Subsection *Polynomial Search Time vs. Sieving Time* in Section 3.1. When we use (1) for predicting CPU times, we neglect the $o(1)$–term, which, in fact, is proportional to $1/\log(N)$. All logarithms have base e.

Table 1. Factoring records since 1970

# decimals	date or year	algorithm	effort (MIPS years)	reference
39	Sep 13, 1970	CF		$F_7 = 2^{2^7} + 1$ [27, 28]
50	1983	CF		[6, pp. xliv–xlv]
55–71	1983–1984	QS		[12, Table I on p. 189]
45–81	1986	QS		[36, p. 336]
78–90	1987–1988	QS		[37]
87–92	1988	QS		[32, Table 3 on p. 274]
93–102	1989	QS		[21]
107–116	1990	QS	275 for C116	[22]
RSA–100	Apr 1991	QS	7	[34]
RSA–110	Apr 1992	QS	75	[14]
RSA–120	Jun 1993	QS	835	[13]
RSA–129	Apr 1994	QS	5000	[2]
RSA–130	Apr 1996	NFS	1000	[11]
RSA–140	Feb 1999	NFS	2000	[8]
RSA–155	Aug 1999	NFS	8400	this paper

to be roughly a linear function of the calendar year Y. From the data in Table 1 he derives the linear formula

$$Y = 13.24D^{1/3} + 1928.6.$$

According to this formula, a general 768–bit number (D=231) will be factored by the year 2010, and a general 1024–bit number (D=309) by the year 2018.

Directions for selecting cryptographic key sizes now and in the coming years are given in [23].

The vulnerability of a 512–bit RSA modulus was predicted long ago. A 1991 report [3, p. 81] recommends:

> *For the most applications a modulus size of 1024 bit for RSA should achieve a sufficient level of security for "tactical" secrets for the next ten years. This is for long-term secrecy purposes, for short-term authenticity purposes 512 bit might suffice in this century.*

3 Factoring RSA–155

We assume that the reader is familiar with NFS [19], but for convenience we briefly describe the method here. Let N be the number we wish to factor, known to be composite. There are four main steps in NFS: polynomial selection, sieving, linear algebra, and square root.

The *polynomial selection step* selects two irreducible polynomials $f_1(x)$ and $f_2(x)$ with a common root m mod N. The polynomials have as many smooth values as practically possible over a given factor base.

The *sieve step* (which is by far the most time-consuming step of NFS), finds pairs (a, b) with $\gcd(a, b) = 1$ such that both

$$b^{\deg(f_1)} f_1(a/b) \text{ and } b^{\deg(f_2)} f_2(a/b)$$

are smooth over given factor bases, i.e., factor completely over the factor bases. Such a pair (a, b) is called a *relation*. The purpose of this step is to collect so many relations that several subsets S of them can be found with the property that a product taken over S yields an expression of the form

$$X^2 \equiv Y^2 \pmod{N}. \tag{2}$$

For approximately half of these subsets, computing $\gcd(X - Y, N)$ yields a non-trivial factor of N (if N has exactly two distinct factors).

The *linear algebra step* first filters the relations found during sieving, with the purpose of eliminating duplicate relations and relations containing a prime or prime ideal which does not occur elsewhere. In addition, certain relations are *merged* with the purpose of eliminating primes and prime ideals which occur exactly k times in k different relations, for $k = 2, \ldots, 8$. These merges result in so-called relation–sets, defined in Section 3.3, which form the columns of a very large sparse matrix over $\mathcal{F}_2$. With help of an iterative block Lanczos algorithm a few dependencies are found in this matrix: this is the most time– and space–consuming part of the linear algebra step.

The *square root step* computes the square root of an algebraic number of the form

$$\prod_{(a,b)\in S} (a - b\alpha),$$

where α is a root of one of the polynomials $f_1(x), f_2(x)$, and where for RSA–155 the numbers a, b and the cardinality of the set S can all be expected to be many millions. All $a - b\alpha$'s have smooth norms. With the mapping $\alpha \mapsto m \bmod N$, this leads to a congruence of the form (2).

In the next four subsections, we describe these four steps, as carried out for the factorization of RSA–155.

3.1 Polynomial selection

This section has three parts. The first two parts are aimed at recalling the main details of the polynomial selection procedure, and describing the particular polynomials used for the RSA–155 factorization.

Relatively speaking, our selection for RSA–155 is approximately 1.7 times better than our selection for RSA–140. We made better use of our procedure for RSA–155 than we did for RSA–140, in short by searching longer. This poses a new question for NFS factorizations—what is the optimal trade-off between increased polynomial search time and the corresponding saving in sieve time? The third part of this section gives preliminary consideration to this question as it applies to RSA–155.

The Procedure Our polynomial selection procedure is outlined in [8]. Here we merely restate the details. Recall that we generate two polynomials f_1 and f_2, using a base-m method. The degree d of f_1 is fixed in advance (for RSA–155 we take $d = 5$). Given a potential a_5, we choose an integer $m \approx (N/a_d)^{1/d}$. The polynomial

$$f_1(x) = a_d x^d + a_{d-1} x^{d-1} + \ldots + a_0 \tag{3}$$

descends from the base-m representation of N, initially adjusted so that $|a_i| \leq m/2$ for $0 \leq i \leq d-1$.

Sieving occurs over the homogeneous polynomials $F_1(x, y) = y^d f_1(x/y)$ and $F_2(x, y) = x - my$. The aim for polynomial selection is to choose f_1 and m such that the values $F_1(a, b)$ and $F_2(a, b)$ are simultaneously smooth at many coprime integer pairs (a, b) in the sieving region. That is, we seek F_1, F_2 with good *yield*. Since F_2 is linear, we concentrate on the choice of F_1.

There are two factors which influence the yield of F_1, *size* and *root properties*, so we seek F_1 with a good *combination* of size and root properties. By size we refer to the magnitude of the values taken by F_1. By root properties we refer to the extent to which the distribution of the roots of F_1 modulo small p^n, for p prime and $n \geq 1$, affects the likelihood of F_1 values being smooth. In short, if F_1 has many roots modulo small p^n, the values taken by F_1 "behave" as if they are much smaller than they actually are. That is, on average, the likelihood of F_1-values being smooth is increased.

Our search is a *two stage* process. In the first stage we generate a large sample of good polynomials (polynomials with good combinations of size and root properties). In the second stage we identify *without* sieving, the best polynomials in the sample. We concentrate on *skewed* polynomials, that is, polynomials $f_1(x) = a_5 x^5 + \ldots + a_0$ whose first few coefficients (a_5, a_4 and a_3) are small compared to m, and whose last few coefficients (a_2, a_1 and a_0) may be large compared to m. Usually $|a_5| < |a_4| < \cdots < |a_0|$. To compensate for the last few coefficients being large, we sieve over a skewed region, i.e., a region that is much longer in x than in y. We take the region to be a rectangle whose width-to-height ratio is s.

The first stage of the process, generating a sample of polynomials with good yield, has the following main steps ($d = 5$):

- Guess leading coefficient a_d, usually with several small prime divisors (for projective roots).
- Determine initial m from $a_d m^d \approx N$. If the approximation $(N - a_d m^d)/m^{d-1}$ to a_{d-1} is not close to an integer, try another a_d. Otherwise use (3) to determine a starting f_1.
- Try to replace the initial f_1 by a smaller one. This numerical optimization step replaces $f_1(x)$ by

  $$f_1(x + k) + (cx + d) * (x + k - m)$$

  and m by $m - k$, sieving over a region with skewness s. It adjusts four real parameters c, d, k, s, rounding the optimal values (except s) to integers.

– Make adjustments to f_1 which cause it to have exceptionally good root properties, *without* destroying the qualities inherited from above. The main adjustment is to consider integer pairs j_1, j_0 (with j_1 and j_0 small compared to a_2 and a_1 respectively) for which the polynomial

$$f_1(x) + (j_1 x - j_0) \cdot (x - m)$$

has exceptionally good root properties modulo many small p^n. Such pairs j_1, j_0 are identified using a sieve-like procedure. For each promising (j_1, j_0) pair, we revise the translation k and skewness s by repeating the numerical optimization on these values alone.

In the second stage of the process we rate, without sieving, the yields of the polynomial pairs F_1, F_2 produced from the first stage. We use a parameter which quantifies the effect of the root properties of each polynomial. We factor this parameter into estimates of smoothness probabilities for F_1 and F_2 across a region of skewness s.

At the conclusion of these two stages we perform short sieving experiments on the top-ranked candidates.

Results Four of us spent about 100 MIPS years on finding good polynomials for RSA–155. The following pair, found by Dodson, was used to factor RSA–155:

$$\begin{aligned} F_1(x,y) = {} & 11\,93771\,38320\,x^5 \\ & -80\,16893\,72849\,97582\,x^4y \\ & -66269\,85223\,41185\,74445\,x^3y^2 \\ & +1\,18168\,48430\,07952\,18803\,56852\,x^2y^3 \\ & +745\,96615\,80071\,78644\,39197\,43056\,x\,y^4 \\ & -40\,67984\,35423\,62159\,36191\,37084\,05064\,y^5 \end{aligned}$$

$$F_2(x,y) = x - 3912\,30797\,21168\,00077\,13134\,49081\,y$$

with $s \approx 10800$.

For the purpose of comparison, we give statistics for the above pair similar to those we gave for the RSA–140 polynomials in [8]. Denote by a_{max} the largest $|a_i|$ for $i = 0, \ldots, d$. The un-skewed analogue, $F_1(104x, y/104)$, of F_1 has $a_{max} \approx 1.1 \cdot 10^{23}$, compared to the typical case for RSA–155 of $a_{max} \approx 2.4 \cdot 10^{25}$. The un-skewed analogue of F_2 has $a_{max} \approx 3.8 \cdot 10^{26}$. Hence, F_1 values have shrunk by approximately a factor of 215, whilst F_2 values have grown by a factor of approximately 16. F_1 has real roots x/y near -11976, -2225, 1584, 12012 and 672167.

With respect to the root properties of F_1 we have $a_5 = 2^4 \cdot 3^2 \cdot 5 \cdot 11^2 \cdot 19 \cdot 41 \cdot 1759$. Also, $F_1(x, y)$ has 20 roots x/y modulo the six primes from 3 to 17 and an additional 33 roots modulo the 18 primes from 19 to 97. As a result of its root properties, F_1-values have smoothness probabilities similar to those of random integers which are smaller by a factor of about 800.

Polynomial Search Time vs. Sieving Time The yield of our two RSA–155 polynomials is approximately 13.5 times that of a skewed pair of average yield for RSA–155 (about half of which comes from root properties and the other half from size). The corresponding figure for the RSA–140 pair is approximately 8 (about a factor of four of which was due to root properties and the remaining factor of 2 to size). From this we deduce that, relatively speaking, our RSA–155 selection is approximately 1.7 times "better" than our RSA–140 selection.

Note that this is consistent with the observed differences in sieve time. As noted above, straightforward extrapolation of the NFS asymptotic run-time estimate (1) suggests that sieving for RSA–155 should have taken approximately 7 times as long as RSA–140. The actual figure is approximately 4. The difference can be approximately reconciled by the fact that the RSA–155 polynomial pair is, relatively, about 1.7 times "better" than the RSA–140 pair.

Another relevant comparison is to the RSA–130 factorization. RSA–130 of course was factorized *without* our improved polynomial selection methods. The polynomial pair used for RSA–130 has a yield approximately 3.2 times that of a random (un-skewed) selection or RSA–130. Extrapolation of the asymptotic NFS run-time estimate suggests that RSA–140 should have taken about 4 times as long as RSA–130, whereas the accepted difference is a factor of about 2. The difference is close to being reconciled by the RSA–140 polynomial selection being approximately 2.5 times better than the RSA–130 selection. Finally, to characterize the overall improvement accounted for by our techniques, we note that the RSA–155 selection is approximately 4.2 times better (relatively) than the RSA–130 selection.

Since the root properties of the non-linear polynomials for RSA–140 and RSA–155 are similar, most of the difference between them comes about because the RSA–155 selection is relatively "smaller" than the RSA–140 selection. This in turns comes about because we conducted a longer search for RSA–155 than we did for the RSA–140 search, so it was more likely that we would find good size and good root properties coinciding in the same polynomials. In fact, we spent approximately 100 MIPS years on the RSA–155 search, compared to 60 MIPS years for RSA–140.

Continuing to search for polynomials is worthwhile only as long as the saving in sieve time exceeds the extra cost of the polynomial search. We have analyzed the "goodness" distribution of all polynomials generated during the RSA–155 search. Modulo some crude approximations, the results appear in Table 2. The table shows the expected benefit obtained from κ times the polynomial search effort we actually invested (100 MY), for some useful κ. The second column gives the change in search time corresponding to the κ-altered search effort. The third column gives the expected change in sieve time, calculated from the change in yield according to our "goodness" distribution. Hence, whilst the absolute benefit may not have been great, it would probably have been worthwhile investing up to about twice the effort than we did for the RSA–155 polynomial search. We conclude that, in the absence of further improvements, it is worthwhile using our

Table 2. Effect of varying the polynomial search time on the sieve time

κ	**change** in search time (in MY)	**change** in sieve time (in MY)
0.2	−80	+260
0.5	−50	+110
1	0	0
2	+100	−110
5	+400	−260
10	+900	−380

method to find polynomials whose yields are approximately 10–15 times better than a random selection.

3.2 Sieving

Two sieving methods were used simultaneously: lattice sieving and line sieving. This is probably more efficient than using a single sieve, despite the large percentage of duplicates found (about 14%, see Section 3.3): both sievers deteriorate as the special q, resp. y (see below) increase, so we exploited the most fertile parts of both. In addition, using two sievers offers more flexibility in terms of memory: lattice sieving is possible on smaller machines; the line siever needs more memory, but discovers each relation only once.

The lattice siever fixes a prime q, called the special q, which divides $F_1(x_0, y_0)$ for some known nonzero pair (x_0, y_0), and finds (x, y) pairs for which both $F_1(x, y)/q$ and $F_2(x, y)$ are smooth. This is carried out for many special q's. Lattice sieving was introduced by Pollard [31] and the code we used is the implementation written by Arjen Lenstra and described in [18, 11], with some additions to handle skewed sieving regions efficiently.

The line siever fixes a value of y (from $y = 1, 2, \ldots$ up to some bound) and finds values of x in a given interval for which both $F_1(x, y)$ and $F_2(x, y)$ are smooth. The line siever code was written by Peter Montgomery, with help from Arjen Lenstra, Russell Ruby, Marije Elkenbracht-Huizing and Stefania Cavallar.

For the lattice sieving, both the rational and the algebraic factor base bounds were chosen to be $2^{24} = 16\,777\,216$. The number of primes was about one million in each factor base. Two large primes were allowed on each side in addition to the special q input. The reason that we used these factor base bounds is that we used the lattice sieving implementation from [18] which does not allow larger factor base bounds. That implementation was written for the factorization of RSA–130 and was never intended to be used for larger numbers such as RSA–140, let alone RSA–155. We expect that a rewrite of the lattice siever that would allow larger factor base bounds would give a much better lattice sieving performance for RSA–155.

Most of the line sieving was carried out with two large primes on both the rational and the algebraic side. The rational factor base consisted of 2 661 384 primes $< 44\,000\,000$ and the algebraic factor base consisted of 6 304 167 prime

ideals of norm $< 110\,000\,000$ (including the seven primes which divide the leading coefficient of $F_1(x, y)$). Some line sieving allowed *three* large primes instead of *two* on the algebraic side. In that case the rational factor base consisted of 539 777 primes $< 8\,000\,000$ and the algebraic factor base of 1 566 598 prime ideals of norm $< 25\,000\,000$ (including the seven primes which divide the leading coefficient of $F_1(x, y)$).

For both sievers the large prime bound $1\,000\,000\,000$ was used both for the rational and for the algebraic primes.

The lattice siever was run for most special q's in the interval $[2^{24}, 3.08 \times 10^8]$. Each special q has at least one root r such that $f_1(r) \equiv 0 \bmod q$. For example, the equation $f_1(x) \equiv 0 \bmod q$ has five roots for $q = 83$, namely $x = 8$, 21, 43, 54, 82, but no roots for $q = 31$. The total number of special q–root pairs (q, r) in the interval $[2^{24}, 3.08 \times 10^8]$ equals about 15.7M. Lattice sieving ranged over a rectangle of 8192 by 5000 points per special q–root pair. Taking into account that we did not sieve over points (x, y) where both x and y are even, this gives a total of 4.8×10^{14} sieving points. With lattice sieving a total of 94.8M relations were generated at the expense of 26.6 years of CPU time. Averaged over all the CPUs on which the lattice siever was run, this gives an average of 8.8 CPU seconds per relation.

For the line sieving with two large primes on both sides, sieving ranged over the regions[3]:

$$\begin{aligned} |x| &\le 1\,176\,000\,000, & 1 &\le y \le 25\,000, \\ |x| &\le 1\,680\,000\,000, & 25\,001 &\le y \le 110\,000, \\ |x| &\le 1\,680\,000\,000, & 120\,001 &\le y \le 159\,000, \end{aligned}$$

and for the line sieving with three large primes instead of two on the algebraic side, the sieving range was:

$$|x| \le 1\,680\,000\,000, \quad 110\,001 \le y \le 120\,000.$$

Not counting the points where both x and y are even, this gives a total of 3.82×10^{14} points sieved by the line siever. With line sieving a total of 36.0M relations were generated at the expense of 9.1 years of CPU time. Averaged over all the CPUs on which the line siever was run, it needed 8.0 CPU seconds to generate one relation.

Sieving was done at twelve different locations where a total of 130.8M relations were generated, 94.8M by lattice sieving and 36.0M by line sieving. Each incoming file was checked at the central site for duplicates: this reduced the total number of useful incoming relations to 124.7M. Of these, 88.8M (71%) were found by the lattice siever and 35.9M (29%) by the line siever. The breakdown of the 124.7M relations (in %) among the twelve different sites[4] is given in Table 3.

Calendar time for the sieving was 3.7 months. Sieving was done on about 160 SGI and Sun workstations (175–400 MHz), on eight R10000 processors

[3] The somewhat weird choice of the line sieving intervals was made because more contributors chose line sieving than originally estimated.

[4] Lenstra sieved at two sites, viz., Citibank and Univ. of Sydney.

Table 3. Breakdown of sieving contributions

%	number of CPU days sieved	La(ttice) Li(ne)	Contributor
20.1	3057	La	Alec Muffett
17.5	2092	La, Li	Paul Leyland
14.6	1819	La, Li	Peter L. Montgomery, Stefania Cavallar
13.6	2222	La, Li	Bruce Dodson
13.0	1801	La, Li	François Morain and Gérard Guillerm
6.4	576	La, Li	Joël Marchand
5.0	737	La	Arjen K. Lenstra
4.5	252	Li	Paul Zimmermann
4.0	366	La	Jeff Gilchrist
0.65	62	La	Karen Aardal
0.56	47	La	Chris and Craig Putnam

(250 MHz), on about 120 Pentium II PCs (300–450 MHz), and on four Digital/Compaq boxes (500 MHz). The total amount of CPU-time spent on sieving was 35.7 CPU years.

We estimate the equivalent number of MIPS years as follows. For each contributor, Table 4 gives the number of million relations generated (rounded to two decimals), the number of CPU days d_s sieved for this and the estimated average speed s_s, in million instructions per seconds (MIPS), of the processors on which these relations were generated. In the last column we give the corresponding number of MIPS years $d_s s_s/365$. For the time counting on PCs, we notice that on PCs one usually get *real times* which may be higher than the CPU times.

Summarizing gives a total of 8360 MIPS years (6570 for lattice and 1790 for line sieving). For comparison, RSA–140 took about 2000 MIPS years and RSA–130 about 1000 MIPS years.

A measure of the "quality" of the sieving may be the average number of points sieved to generate one relation. Table 5 gives this quantity for RSA–140 and for RSA–155, for the lattice siever and for the line siever. This illustrates that the sieving polynomials were better for RSA-155 than for RSA–140, especially for the line sieving. In addition, the increase of the linear factor base bound from 500M for RSA–140 to 1000M for RSA–155 accounts for some of the change in yield. For RSA–155, the factor bases were much bigger for line sieving than for lattice sieving. This explains the increase of efficiency of the line siever compared with the lattice siever from RSA–140 to RSA–155.

3.3 Filtering and finding dependencies

The filtering of the data and the building of the matrix were carried out at CWI and took one calendar month.

Table 4. # MIPS years spent on lattice (La) and line (Li) sieving

Contributor	# relations	# CPU days sieved	average speed of processors in MIPS	# MIPS years
Muffett, La	27.46M	3057	285	2387
Leyland, La	19.27M	1395	300	1146
Leyland, Li	4.52M	697	300	573
CWI, La	1.60M	167	175	80
CWI, Li, 2LP	15.64M	1160	210	667
CWI, Li, 3LP	1.00M	492	50	67
Dodson, La	10.28M	1631	175	782
Dodson, Li	7.00M	591	175	283
Morain, La	15.83M	1735	210	998
Morain, Li	1.09M	66	210	38
Marchand, La	7.20M	522	210	300
Marchand, Li	1.11M	54	210	31
Lenstra, La	6.48M	737	210	424
Zimmermann, Li	5.64M	252	195	135
Gilchrist, La	5.14M	366	350	361
Aardal, La	0.81M	62	300	51
Putnam, La	0.76M	47	300	39

Table 5. Average number of points sieved per relation

	lattice siever	line siever
RSA–140	1.5×10^6	3.0×10^7
RSA–155	5.1×10^6	1.1×10^7

Filtering Here we describe the filter strategy which we used for RSA–155. An essential difference with the filter strategy used for RSA–140 is that we applied k-way merges (defined below) with $2 \leq k \leq 8$ for RSA–155, but only 2- and 3-way merges for RSA–140.

First, we give two definitions. A *relation–set* is one relation, or a collection of two or more relations generated by a merge. A *k-way merge* ($k \geq 2$) is the action of combining k relation–sets with a common prime ideal into $k-1$ relation–sets, with the purpose of eliminating that common prime ideal. This is done such that the weight increase is minimal by means of a *minimum spanning tree* algorithm [9].

Among the 124.7M relations collected from the twelve different sites, 21.3M duplicates were found generated by lattice sieving, as well as 17.9M duplicates caused by the simultaneous use of the lattice and the line siever.

During the first filter round, only prime ideals with norm > 10M were considered. In a later stage of the filtering, this 10M-bound was reduced to 7M, in order to improve the possibilities for merging relations. We added 0.2M *free relations* for prime ideals of norm > 10M (cf. [16, Section 4, pp. 234–235]). From the resulting 85.7M relations, 32.5M singletons were deleted, i.e., those relations

with a prime ideal of norm > 10M which does not occur in any other undeleted relation.

We were left with 53.2M relations containing 42.6M different prime ideals of norm > 10M. If we assume that each prime and each prime ideal with norm < 10M occurs at least once, then we needed to reserve at least $(2 - \frac{1}{120})\pi(10^7)$ excess relations for the primes and the prime ideals of norm smaller than 10M, where $\pi(x)$ is the number of primes below x. The factor 2 comes from the *two* polynomials and the correction factor 1/120 takes account of the presence of free relations, where 120 is the order of the Galois group of the algebraic polynomial. With $\pi(10^7) = 664\,579$ the required excess is about 1.3M relations, whereas we had 53.2M − 42.6M = 10.6M excess relations at our disposal.

In the next *merging* step 33.0M relations were removed which would have formed the heaviest relation–sets when performing 2-way merges, reducing the excess from 10.6M to about 2M relations. So we were still allowed to discard about 2.0M − 1.3M = 0.7M relations. The remaining 20.1M non-free relations[5] having 18.2M prime ideals of norm > 10M were used as input for the merge step which eliminated prime ideals occurring in up to eight different relation–sets. During this step we looked at prime ideals of norm > 7M. Here, our approach differs from what we did for RSA–140, where only primes occurring twice or thrice were eliminated. Applying the new filter strategy to RSA–140 would have resulted in a 30% smaller (3.3M instead of 4.7M columns) but only 20% heavier matrix than the one actually used for the factorization of RSA–140 and would have saved 27% on the block Lanczos run time. The k ($k \leq 8$) relations were combined into the lightest possible $k - 1$ relation–sets and the corresponding prime ideal (row in the matrix) was "balanced" (i.e., all entries of the row were made 0). The overall effect was a reduction of the matrix size by one row and one column while increasing the matrix weight when $k > 2$, as described below. We did not perform all possible merges. We limited the program to only do merges which caused a weight increase of at most 7 original relations. The merges were done in ascending order of weight increase.

Since each k-way merge causes an increase of the matrix weight of about $(k - 2)$ times the weight of the lightest relation–set, these merges were not always executed for higher values of k. For example, 7- and 8-way merges were not executed if all the relation–sets were already-combined relations. We decided to discard relation–sets which contained more than 9 relations and to stop merging (and discarding) after 670K relations were discarded. At this point we should have slightly more columns than rows and did not want to lose any more columns. The maximum discard threshold was reached during the 10th pass through the 18.6M prime ideals of norm > 7M, when we allowed the maximum weight increase to be about 6 relations. This means that no merges with weight increase of 7 relations were executed. The filter program stopped with 6.7M relation sets.

For more details and experiments with RSA–155 and other numbers, see [9].

[5] The 0.1M free relations are not counted in these 20.1M relations because the free relations are generated during each filter run.

Finding dependencies From the matrix left after the filter step we omitted the small primes < 40, thus reducing the weight by 15%. The resulting matrix had 6 699 191 rows, 6 711 336 columns, and weight 417 132 631 (62.27 non–zeros per row). With the help of Peter Montgomery's Cray implementation of the block Lanczos algorithm (cf. [25]) it took 224 CPU hours and 2 Gbytes of central memory on the Cray C916 at the SARA Amsterdam Academic Computer Center to find 64 dependencies among the rows of this matrix. Calendar time for this job was 9.5 days.

In order to extract from these 64 dependencies some dependencies for the matrix *including the primes* < 40, quadratic character checks were used as described in [1], [7, §8, §12.7], and [15, last paragraph of Section 3.8 on pp. 30–31]. This yielded a dense 100×64 homogeneous system which was solved by Gaussian elimination. That system turned out to have 14 independent solutions, which represent linear combinations of the original 64 dependencies.

3.4 The square root step

On August 20, 1999, four different square root (cf. [24]) jobs were started in parallel on four different 300 MHz processors of an SGI Origin 2000, each handling one dependency. One job found the factorization after 39.4 CPU-hours, the other three jobs found the trivial factorization after 38.3, 41.9, and 61.6 CPU-hours (different CPU times are due to the use of different parameters in the four jobs).

We found that the 155–digit number

RSA–155 =
10941738641570527421809707322040357612003732945449205990913842131476349984288\
9\
34784717997257891267332497625752899781833797076537244027146743531593354333897

can be written as the product of two 78-digit primes:

$p =$
102639592829741105772054196573991675900716567808038066803341933521790711307779

and

$q =$
106603488380168454820927220360012878679207958575989291522270608237193062808643.

Primality of the factors was proved with the help of two different primality proving codes [4, 10]. The factorizations of $p \pm 1$ and $q \pm 1$ are given by

$p - 1 = 2 \cdot 607 \cdot$
$\cdot 305999 \cdot 27629703635780610779648399797990013970853704055088589435565914357547 3$
$p + 1 = 2^2 \cdot 3 \cdot 5 \cdot$
$\cdot 5253077241827 \cdot 32564910084983334243687187047739463487939806729537209529153126 9$

$q - 1 = 2 \cdot 241 \cdot$
$\cdot 4300281522612815813261715143129859438007775343751663992501292842228559750 11$
$q + 1 = 2^2 \cdot 3 \cdot 130637011 \cdot$
$\cdot 237126941204057 \cdot 102002421552989178717 97 \cdot 2811464174834353160353366747817 3$

Acknowledgements. Acknowledgements are due to the Dutch National Computing Facilities Foundation (NCF) for the use of the Cray C916 supercomputer at SARA, and to (in alphabetical order)

the Australian National University (Canberra),
Centre Charles Hermite (Nancy, France),
Citibank (Parsippany, NJ, USA),
CWI (Amsterdam, The Netherlands),
École Polytechnique/CNRS (Palaiseau, France),
Entrust Technologies Ltd. (Ottawa, Canada),
Lehigh University (Bethlehem, PA, USA),
the Magma Group of John Cannon at the University of Sydney,
the Medicis Center at École Polytechnique (Palaiseau, France),
Microsoft Research (Cambridge, UK),
the Putnams (Hudson, NH, USA),
Sun Microsystems Professional Services (Camberley, UK), and
Utrecht University (The Netherlands),

for the use of their computing resources.

References

1. L.M. Adleman. Factoring numbers using singular integers. In *Proc. 23rd Annual ACM Symp. on Theory of Computing (STOC)*, pages 64–71, ACM, New York, 1991.
2. D. Atkins, M. Graff, A.K. Lenstra, and P.C. Leyland. THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE. In J. Pieprzyk and R. Safavi-Naini, editors, *Advances in Cryptology – Asiacrypt '94*, volume 917 of *Lecture Notes in Computer Science*, pages 265–277, Springer-Verlag, Berlin, 1995.
3. Th. Beth, M. Frisch, and G.J. Simmons, editors. *Public–Key Cryptography: State of the Art and Future Directions*, volume 578 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1992. Report on workshop at Oberwolfach, Germany, July, 1991.
4. Wieb Bosma and Marc-Paul van der Hulst. *Primality proving with cyclotomy*. PhD thesis, University of Amsterdam, December 1990.
5. Richard P. Brent. Some parallel algorithms for integer factorisation. *Proc. Europar'99 (Toulouse, Sept. 1999)*, volume 1685 of *Lecture Notes in Computer Science*, pages 1–22, Springer-Verlag, Berlin, 1999.
6. J. Brillhart, D.H. Lehmer, J.L. Selfridge, B. Tuckerman, and S.S. Wagstaff, Jr. *Factorizations of* $b^n \pm 1, b = 2, 3, 5, 6, 7, 10, 11, 12$ *up to high powers*, volume 22 of *Contemporary Mathematics*. American Mathematical Society, second edition, 1988.

7. J.P. Buhler, H.W. Lenstra, Jr., and Carl Pomerance. Factoring integers with the number field sieve. Pages 50–94 in [19].
8. S. Cavallar, B. Dodson, A. Lenstra, P. Leyland, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, and P. Zimmermann. Factorization of RSA–140 using the number field sieve. In Lam Kwok Yan, Eiji Okamoto, and Xing Chaoping, editors, *Advances in Cryptology – Asiacrypt '99 (Singapore, November 14–18)*, volume 1716 of *Lecture Notes in Computer Science*, pages 195–207, Springer-Verlag, Berlin, 1999.
9. S. Cavallar. Strategies for filtering in the Number Field Sieve. Preprint, to appear in the Proceedings of ANTS-IV (*Algorithmic Number Theory Symposium IV*, Leiden, The Netherlands, July 2–7, 2000), *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2000.
10. H. Cohen and A.K. Lenstra. Implementation of a new primality test. *Mathematics of Computation*, 48:103–121, 1987.
11. James Cowie, Bruce Dodson, R.-Marije Elkenbracht-Huizing, Arjen K. Lenstra, Peter L. Montgomery, and Jörg Zayer. A world wide number field sieve factoring record: on to 512 bits. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – Asiacrypt '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 382–394, Springer-Verlag, Berlin, 1996.
12. J.A. Davis, D.B. Holdridge, and G.J. Simmons. Status report on factoring (at the Sandia National Laboratories). In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology, Eurocrypt '84*, volume 209 of *Lecture Notes in Computer Science*, pages 183–215, Springer-Verlag, Berlin, 1985..
13. T. Denny, B. Dodson, A.K. Lenstra, and M.S. Manasse, On the factorization of RSA–120. In D.R. Stinson, editor, *Advances in Cryptology – Crypto '93*, volume 773 of *Lecture Notes in Computer Science*, pages 166–174, Springer-Verlag, Berlin, 1994.
14. B. Dixon and A.K. Lenstra. Factoring using SIMD Sieves. In Tor Helleseth, editor, *Advances in Cryptology, Eurocrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pages 28–39, Springer-Verlag, Berlin, 1994.
15. Marije Elkenbracht-Huizing. *Factoring integers with the number field sieve.* PhD thesis, Leiden University, May 1997.
16. R.-M. Elkenbracht-Huizing. An implementation of the number field sieve. *Experimental Mathematics*, 5:231–253, 1996.
17. *Frequently Asked Questions about today's Cryptography 4.0.* Question 3.1.9, see `http://www.rsa.com/rsalabs/faq/html/3-1-9.html`.
18. R. Golliver, A.K. Lenstra, and K.S. McCurley. Lattice sieving and trial division. In Leonard M. Adleman and Ming-Deh Huang, editors, *Algorithmic Number Theory, (ANTS-I, Ithaca, NY, USA, May 1994)*, volume 877 of *Lecture Notes in Computer Science*, pages 18–27, Springer-Verlag, Berlin, 1994.
19. A.K. Lenstra and H.W. Lenstra, Jr., editors. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993.
20. A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse, and J.M. Pollard. The factorization of the Ninth Fermat number. *Mathematics of Computation*, 61(203):319–349, July 1993.
21. A.K. Lenstra and M.S. Manasse. Factoring by Electronic Mail. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology – Eurocrypt '89*, volume 434 of *Lecture Notes in Computer Science*, pages 355–371, Springer-Verlag, Berlin, 1990.
22. A.K. Lenstra and M.S. Manasse. Factoring with two large primes. In I.B. Dåmgard, editor, *Advances in Cryptology – Eurocrypt '90*, volume 473 of *Lecture Notes in Computer Science*, pages 72–82, Springer-Verlag, Berlin, 1991.

23. Arjen K. Lenstra and Eric R. Verheul. Selecting Cryptographic Key Sizes. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 446–465, Springer-Verlag, Berlin, 2000.
24. Peter L. Montgomery. Square roots of products of algebraic numbers. In Walter Gautschi, editor, *Mathematics of Computation 1943–1993: a Half-Century of Computational Mathematics*, pages 567–571. Proceedings of Symposia in Applied Mathematics, American Mathematical Society, 1994.
25. Peter L. Montgomery. A block Lanczos algorithm for finding dependencies over GF(2). In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology – Eurocrypt '95*, volume 921 of *Lecture Notes in Computer Science*, pages 106–120, Springer-Verlag, Berlin, 1995.
26. Peter L. Montgomery and Brian Murphy. Improved Polynomial Selection for the Number Field Sieve. Extended Abstract for the Conference on the Mathematics of Public-Key Cryptography, June 13–17, 1999, The Fields Institute, Toronto, Ontario, Canada.
27. Michael A. Morrison and John Brillhart. The factorization of F_7. *Bull. Amer. Math. Soc.*, 77(2):264, 1971.
28. Michael A. Morrison and John Brillhart. A method of factoring and the factorization of F_7. *Mathematics of Computation*, 29:183–205, January 1975.
29. B. Murphy. Modelling the Yield of Number Field Sieve Polynomials. J. Buhler, editor, *Algorithmic Number Theory, (Third International Symposium, ANTS-III, Portland, Oregon, USA, June 1998)*, volume 1423 of *Lecture Notes in Computer Science*, pages 137–151, Springer-Verlag, Berlin, 1998.
30. Brian Antony Murphy. Polynomial Selection for the Number Field Sieve Integer Factorisation Algorithm. PhD thesis, The Australian National University, July 1999.
31. J.M. Pollard. The lattice sieve. Pages 43–49 in [19].
32. Herman te Riele, Walter Lioen, and Dik Winter. Factoring with the quadratic sieve on large vector computers. *J. Comp. Appl. Math.*, 27:267–278, 1989.
33. R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21:120–126, 1978.
34. RSA Challenge Administrator. In order to obtain information about the RSA Factoring Challenge, send electronic mail to `challenge-info@rsa.com`. The status of the factored numbers on the RSA Challenge List can be obtained by sending electronic mail to `challenge-honor-rolls@majordomo.rsasecurity.com`. Also visit `http://www.rsa.com/rsalabs/html/factoring.html`.
35. A. Shamir. Factoring large numbers with the TWINKLE device. In C.K. Koc and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES)*, volume 1717 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1999.
36. Robert D. Silverman. The multiple polynomial quadratic sieve. *Mathematics of Computation*, 48:329–339, 1987.
37. Robert D. Silverman. Private communication.
38. URL: `http://www.bxa.doc.gov/Encryption/Default.htm`.

A Details of recent absolute and SNFS factoring records

Table 6. Absolute factoring records

# digits	129	130	140	155
method	QS	GNFS	GNFS	GNFS
code	Gardner	RSA–130	RSA–140	RSA–155
factor date	Apr 2, 1994	Apr 10, 1996	Feb 2, 1999	Aug 22, 1999
size of p, q	64, 65	65, 65	70, 70	78, 78
sieve time (in MIPS years)	5000	1000	2000	8400
total sieve time (in CPU years)	?	?	8.9	35.7
calendar time for sieving (in days)	∼270	120	30	110
matrix size	0.6M	3.5M	4.7M	6.7M
row weight	47	40	32	62
Cray CPU hours	n.a.	67	100	224
group	Internet	Internet	CABAL	CABAL

Table 7. Special Number Field Sieve factoring records

# digits	148[20]	167	180	186	211
code	2,512+	3,349−	12,167+	NEC	10,211−
factor date	Jun 15, 1990	Feb 4, 1997	Sep 3, 1997	Sep 15, 1998	April 8, 1999
size of p, q	49, 99	80, 87	75, 105	71, 73	93, 118
total sieve time (in CPU years)	340[a]	?	1.5	5.1	10.9
calendar time for sieving (in days)	83	?	10	42	64
matrix size	72K	?	1.9M	2.5M	4.8M
row weight	dense	?	29	27	49
Cray CPU hours	3[b]	?	16	25	121
group	Internet	NFSNET	CWI	CWI	CABAL

[a] MIPS years

[b] carried out on a Connection Machine

An Algorithm for Solving the Discrete Log Problem on Hyperelliptic Curves

Pierrick Gaudry*

LIX, École Polytechnique,
91128 Palaiseau Cedex, France
gaudry@lix.polytechnique.fr

Abstract. We present an index-calculus algorithm for the computation of discrete logarithms in the Jacobian of hyperelliptic curves defined over finite fields. The complexity predicts that it is faster than the Rho method for genus greater than 4. To demonstrate the efficiency of our approach, we describe our breaking of a cryptosystem based on a curve of genus 6 recently proposed by Koblitz.

1 Introduction

The use of hyperelliptic curves in public-key cryptography was first proposed by Koblitz in 1989 [24]. It appears as an alternative to the use of elliptic curves [23] [31], with the advantage that it uses a smaller base field for the same level of security. Several authors have given ways to build hyperelliptic cryptosystems efficiently. The security of such systems relies on the difficulty of solving the discrete logarithm problem in the Jacobian of hyperelliptic curves. If an algorithm tries to solve this problem performing "simple" group operations only, it was shown by Shoup [39] that the complexity is at least $\Omega(\sqrt{n})$, where n is the largest prime dividing the order of the group. Algorithms with such a complexity exist for generic groups and can be applied to hyperelliptic curves, but are still exponential. The Pollard Rho method and its parallel variants are the most important examples [34], [46], [17].

For the elliptic curve discrete logarithm problem, there are some particular cases where a solution can be found with a complexity better than $O(\sqrt{n})$. See [30], [38], [40], [37]. Similar cases were discovered for hyperelliptic curves [14], [35]. However they are very particular and can be easily avoided when designing a cryptosystem.

In 1994, Adleman, DeMarrais and Huang [1] published the first algorithm (ADH for short) to compute discrete logs which runs in subexponential time when the genus is sufficiently large compared to the size of the ground field. This algorithm was rather theoretical, and some improvements to it were done. Flassenberg and Paulus [13] implemented a sieve version of this algorithm, but

* This work was supported by Action COURBES of INRIA (action coopérative de la direction scientifique de l'INRIA).

the consequences for cryptographical applications is not clear. Enge [11] improved the original algorithm and gave a precise evaluation of the running time, but did not implement his ideas. Müller, Stein and Thiel [32] extended the results to the real quadratic congruence function fields. Smart and Galbraith [16] also gave some ideas in the context of the Weil descent, following ideas of Frey; they dealt with general curves (not hyperelliptic).

Our purpose is to present a variant of existing index-calculus algorithms like ADH or Hafner-McCurley [19], which allowed us to break a cryptosystem based on a curve of genus 6 recently proposed by Koblitz. The main improvement is due to the fact that the costly HNF computation in classical algorithms is replaced by that of the kernel of a sparse matrix. A drawback is that we have to assume that the order of the group in which we are working is known. This is not a constraint in a cryptographical context, because the knowledge of this order is preferable to build protocols. But from a theoretical point of view it differs from ADH or Hafner-McCurley algorithm where the order of the group was a byproduct of the discrete logarithm computation (in fact the aim of the HNF computation was to find the group structure).

We will analyse our method for small genus and show that it is faster than the Pollard Rho method as soon as the genus is strictly greater than 4. Indeed its complexity is $O(q^2)$ where q is the cardinality of the base field. We will explain below some consequences for the choice of the parameters, curve and base field, when building a cryptosystem.

Moreover, the presence of an automorphism of order m on the curve can be used to speed up the computation, just as in the Rho method [9] [17] [48]. This is the case in almost all the examples in the literature. The gain in the Rho method is a factor $\sqrt{m}$, but the gain obtained here is a factor m^2, which is very significant in practice.

The organization of the paper is as follows: in section 2 after some generalities on hyperelliptic curves, our algorithm is described. It is analyzed in section 3, and in section 4 we explain how the presence of an automorphism can help. Finally the section 5 gives some details on our implementation and the results of our experiments with Koblitz's curve.

2 Description of the Algorithm

2.1 Hyperelliptic Curves

We give an overview of the theory of hyperelliptic curves. More precise statements can be found in [24], [4], [15]. We will restrict ourselves to the so-called imaginary quadratic case.

A *hyperelliptic curve* $\mathcal{C}$ of genus g over a field $\mathbb{K}$ is a smooth plane projective curve which admits an affine equation of the form $y^2 + h(x)y = f(x)$, where f is a polynomial of degree $2g+1$, and h is a polynomial of degree at most g, both with coefficients in $\mathbb{K}$.

A *divisor* on the curve $\mathcal{C}$ is a finite formal sum of points of the curve. The set of all divisors yield an abelian group denoted by $Div(\mathcal{C})$. For each divisor

$D = \sum_i n_i P_i \in Div(\mathcal{C})$, where the P_i are points on the curve, we define the *degree* of D by $\deg(D) = \sum_i n_i$. The set of all divisors of degree zero is a sub-group of $Div(\mathcal{C})$ denoted by $Div^0(\mathcal{C})$.

For each function $\varphi(x, y)$ on the curve, we can define a divisor denoted by $\mathrm{div}(\varphi)$ by assigning at each point P_i of the curve the value n_i equal to the multiplicity of the zero if $\varphi(P_i) = 0$, or the opposite of the multiplicity of the pole if the function is not defined at P_i. It can be shown that the sum is finite, and moreover that the degree of such a divisor is always zero. The set of all divisors built from a function a subgroup of $Div^0(\mathcal{C})$ denoted by $\mathcal{P}(\mathcal{C})$ and we call these divisors *principal*. The *Jacobian* of the curve $\mathcal{C}$ is then defined by the quotient group $Jac(\mathcal{C}) = Div(\mathcal{C})^0/\mathcal{P}(\mathcal{C})$.

If the base field of the curve is a finite field with cardinality q, then the Jacobian of the curve is a finite abelian group of order around q^g. The Hasse-Weil bound gives a precise interval for this order: $(\sqrt{q}-1)^{2g} \leq \#Jac(\mathcal{C}) \leq (\sqrt{q}+1)^{2g}$.

In [4], Cantor gave an efficient algorithm for the computation of the group law. We do not recall his method, but we recall the representation of the elements.

Proposition 1 *In every class of divisors in $Jac(\mathcal{C})$, there exists an unique divisor $D = P_1 + \cdots + P_g - g\infty$, such that for all $i \neq j$, P_i and P_j are not symmetric points. Such a divisor is called* reduced*, and there is a unique representation of D by two polynomials $[u, v]$, such that $\deg v < \deg u \leq g$, and u divides $v^2 + hv - f$.*

In this representation, the roots of the polynomial u are exactly the abscissae of the points which occur in the reduced divisor.

The group $Jac(\mathcal{C})$ can now be used in cryptographical protocols based on the discrete logarithm problem, for example Diffie-Hellman or ElGamal's protocols. The security relies on the difficulty of the following problem.

Definition 1 *The* hyperelliptic discrete logarithm problem *takes on input a hyperelliptic curve of given genus, an element D_1 of the Jacobian, its order n, and another element D_2 in the subgroup generated by D_1. The problem is to find an integer λ modulo n such that $D_2 = \lambda.D_1$.*

2.2 Smooth Divisors

Like any index-calculus method, our algorithm is based on the notions of smoothness, and prime elements. We will recall these notions for divisors on hyperelliptic curves, which were first defined in ADH.

Definition 2 *With the polynomial representation $D = [u, v]$, a divisor will be said to be* prime *if the polynomial u is irreducible over $\mathbb{F}_q$.*

For a prime divisor D, when there is no possible confusion with the degree of D as a divisor (which is always zero), we will talk about the degree of D instead of the degree of u.

Proposition 2 *A divisor D of $Jac(\mathcal{C})$ represented by the polynomials $[u, v]$ is equal to the sum of prime divisors $[u_i, v_i]$, where the u_i are the prime factors of u.*

Now we can give the smoothness definition. Let S be an integer called the smoothness bound.

Definition 3 *A divisor is said to be* S-smooth *if all its prime divisors are of degree at most S. When $S = 1$, a 1-smooth divisor will be a divisor for which the polynomial u splits completely over $\mathbb{F}_q$.*

The case $S = 1$ is the most important for two reasons: the first one is that for a relatively small genus (say at most 9), and a reasonable field size, this choice is the best in practice. The second one is that if we want to analyze our algorithm for a fixed g and a q tending to infinity, this is also the good choice.

The definition of a smooth divisor can be seen directly on the expression of D as a sum of points of the curve. Note that a divisor defined over $\mathbb{F}_q$ is defined by being invariant under the Galois action. But it does not imply that the points occuring in it are defined over $\mathbb{F}_q$; they can be exchanged by Galois. Hence an equivalent definition of smoothness is given by the following proposition.

Proposition 3 *A divisor $D = P_1 + \cdots + P_g - g\infty$ is S-smooth if and only if each point P_i is defined over an extension $\mathbb{F}_{q^k}$ with $k \leq S$.*

We define also a factor basis, similar to the one used for classical discrete log problem over $\mathbb{F}_p^*$.

Definition 4 *The* factor basis*, denoted by G_S, is the set of all the prime divisors of degree at most S. For $S = 1$ we simply write G.*

In the following, we will always take $S = 1$ and we will say 'smooth divisor' for 1-smooth divisor.

2.3 Overview of the Algorithm

For the sake of simplicity, we will suppose that the Jacobian of the curve has an order which is almost prime and that we have to compute a discrete log in the subgroup of large prime order (this is always the case in cryptography). Let $n = \mathrm{ord}(D_1)$ be this prime order, and D_2 be the element for which we search the log.

We introduce a pseudo-random walk (as in [45]) in the subgroup generated by D_1: Let $R_0 = \alpha_0 D_1 + \beta_0 D_2$ be the starting point of the walk, where R_0 is the reduced divisor obtained by Cantor's algorithm, and α_0 and β_0 are random integers. For j from 1 to r, we compute random divisors $T^{(j)} = \alpha^{(j)} D_1 + \beta^{(j)} D_2$. The walk will then be given by $R_{i+1} = R_i + T^{(\mathcal{H}(R_i))}$, where $\mathcal{H}$ is a hash function from the subgroup generated by D_1 to the interval $[1, r]$. This hash function is assumed to have good statistical properties; in practice, it can be given by the last bits in the internal representation of the divisors. Once the initialization is finished, we can compute a new pseudo-random element R_{i+1} at the cost of one addition in the Jacobian. Moreover at each step we get a representation of R_{i+1} as $\alpha_{i+1} D_1 + \beta_{i+1} D_2$, where α_{i+1} and β_{i+1} are integers modulo n.

The classical ρ method is to wait for a collision $R_{i_1} = R_{i_2}$, which will yield the discrete logarithm $\lambda = -(\alpha_{i_1} - \alpha_{i_2})/(\beta_{i_1} - \beta_{i_2}) \bmod n$. We can however make use of the smooth divisors. For each R_i of the random walk, test its smoothness. If it is smooth, express it on the factor basis, else throw it away. Thus we extract a subsequence of the sequence (R_i) where all the divisors are smooth. We denote also by (R_i) this subsequence. Hence we can put the result of this computation in a matrix M, each column representing an element of the factor basis, and each row being a reduced divisor R_i expressed on the basis: for a row i, we have $R_i = \sum_k m_{ik} g_k$, where $M = (m_{ik})$. We collect $w + 1$ rows in order to have a $(w + 1) \times w$ matrix. Thus the kernel of the transpose of M is of dimension at least 1. Using linear algebra, we find a non-zero vector of this kernel, which corresponds to a relation between the R_i's. Then we have a family (γ_i) such that $\sum_i \gamma_i R_i = 0$. Going back to the expression of R_i in function of D_1 and D_2, we get: $\sum_i \gamma_i(\alpha_i D_1 + \beta_i D_2) = 0$, and then

$$\lambda = -\frac{\sum_i \gamma_i \alpha_i}{\sum_i \gamma_i \beta_i}.$$

The discrete logarithm is now found with high probability (the denominator is zero with probability $1/n$).

We summarize this algorithm in the figure 1.

2.4 Details on Critical Phases

In the first step, we have to build the factor basis, and for that, we have to find, if it exists, a polynomial v corresponding to a given irreducible u. This can be rewritten in solving an equation of degree 2 over $\mathbb{F}_q$, which can be done quickly.

The initialization of the random walk is only a matter of operations in the group; after that, computing each random divisor R_i requires a single operation in the group.

One crucial point is to test the smoothness of a divisor, i.e. to decide if a polynomial of degree g (the u of the divisor) splits completely on $\mathbb{F}_q$. A way to do that is to perform the beginning of the factorization of u, which is called DDF (stands for distinct degree factorization). By computing $\gcd(X^q - X, u(X))$, we get the product of all the prime factors of u of degree 1. Thus if the degree of this product is equal to the degree of u, it proves that u splits completely on $\mathbb{F}_q$.

In the case where a smooth divisor is detected, the factorization can be completed, or a trial division with the elements of the basis can be performed.

The linear algebra is the last crucial point. The matrix obtained is sparse, and we have at most g terms in each row. Then sparse technique like Lanczos's [27] or Wiedemann's [47] algorithm can be used, in order to get a solution in time quadratic in the number of rows (instead of cubic by Gaussian elimination).

Some other optimizations can be done to speed up the computation. They will be described in section 5.

Input: A divisor D_1 of a curve of genus g over $\mathbb{F}_q$, of prime order $n = \mathrm{ord}(D_1)$, a divisor $D_2 \in \langle D_1 \rangle$, and a parameter r.
Output: An integer λ such that $D_2 = \lambda D_1$.

1. /* Build the factor basis G */
   For each monic irreducible polynomial u_i over F_q of degree 1, try to find v_i such that $[u_i, v_i]$ is a divisor of the curve. If there is a solution, store $g_i = [u_i, v_i]$ in G (we only put one of the two opposite divisors in the basis).
2. /* Initialization of the random walk */
   For j from 1 to r, select $\alpha^{(j)}$ and $\beta^{(j)}$ at random in $[1..n]$, and compute $T^{(j)} := \alpha^{(j)} D_1 + \beta^{(j)} D_2$.
   Select α_0 and β_0 at random in $[1..n]$ and compute $R_0 := \alpha_0 D_1 + \beta_0 D_2$.
   Set k to 1.
3. /* Main loop */
   (a) /* Look for a smooth divisor */
   Compute $j := \mathcal{H}(R_0)$, $R_0 := R_0 + T^{(j)}$, $\alpha_0 := \alpha_0 + \alpha^{(j)} \bmod n$, and $\beta_0 := \beta_0 + \beta^{(j)} \bmod n$.
   Repeat this step until $R_0 = [u_0(z), v_0(z)]$ is a smooth divisor.
   (b) /* Express R_0 on the basis G */
   Factor $u_0(z)$ over $\mathbb{F}_q$, and determine the positions of the factors in the basis G. Store the result as a row $R_k = \sum m_{ik} g_i$ of a matrix $M = (m_{ik})$.
   Store the coefficients $\alpha_k = \alpha_0$ and $\beta_k = \beta_0$.
   If $k < \#G + 1$, then set $k := k + 1$, and return to step 3.a.
4. /* Linear algebra */
   Find a non zero vector (γ_k) of the kernel of the transpose of the matrix M. The computation can be done in the field $\mathbb{Z}/n\mathbb{Z}$.
5. /* Solution */
   Return $\lambda = -(\sum \alpha_k \gamma_k)/(\sum \beta_k \gamma_k) \bmod n$. (If the denominator is zero, return to step 2.)

Fig. 1. Discrete log algorithm

3 Analysis

3.1 Probability for a Divisor to Be Smooth

The following proposition gives the proportion of smooth divisors and then the probability of smoothness in a random walk. This is a key tool for the complexity analysis.

Proposition 4 *The proportion of smooth divisors in the Jacobian of a curve of genus g over $\mathbb{F}_q$ tends to $1/g!$ when q tends to infinity.*

Proof: This proposition is based on the Hasse-Weil bound for algebraic curves: the number of points of a curve of genus g over a finite field with q elements is

equal to $q+1$ with an error of at most $2g\sqrt{q}$, i.e. for large enough q we can neglect it. Moreover the cardinality of its Jacobian is equal to q^g with an error bounded by approximatively $2gq^{g-\frac{1}{2}}$. Here the approximation holds when q is sufficiently large compared to $4g^2$, which is the case in the applications considered.

To evaluate the proportion of smooth divisors, we consider the number of points of the curve over $\mathbb{F}_q$ which is approximatively q. Now, the smooth divisors of the Jacobian are in bijection with the g-multiset of points of the curve: we have $q^g/g!$ smooth divisors, and the searched proportion is $1/g!$. □

3.2 Complexity

The complexity of the algorithm will be exponential in the size of q, so we will count the number of operations which can be done in polynomial time. These operations are of four types: we denote by c_J the cost of a group operation in the Jacobian, c_q the cost of an operation in the base field, $c_{q,g}$ the cost of an operation on polynomials of degree g over the base field, and c_n the cost of an operation in $\mathbb{Z}/n\mathbb{Z}$, where $n \approx q^g$ is the order of the Jacobian. We consider the enumeration of steps in figure 1.

Step 1. For the building of the factor basis, we have to perform q times (i.e. the number of monic irreducible polynomial of degree 1) a resolution of an equation of degree 2 over $\mathbb{F}_q$. Hence the complexity of this phase is $O(qc_q)$.

Step 2. The initialization of the random walk is only a polynomial number of simple operations. Hence we have $O((\log n)c_J)$ for this step.

Step 3. We have to repeat $\#G = O(q)$ times the steps 3.a. and 3.b.
Step 3.a. The computation of a new element of the random walk costs an addition in the Jacobian and two additions modulo n, and the test for its smoothness costs a first step of DDF. By proposition 4, we have to compute $g!$ divisors on average before getting a smooth one and going away from step 3.a. Hence the cost of this step is $O(g!(c_J + c_n + c_{q,g}))$.
Step 3.b. The final splitting of the polynomial in order to express the divisor on the factor basis can not be proved to be deterministic polynomial (though it is very fast in practice). For the analysis, we can then suppose that we do a trial division with all the elements of the basis. This leads to a complexity of $O(qc_{q,g})$.

Hence the complexity of step 3. is $O(qg!(c_J + c_n + c_{q,g})) + O(q^2 c_{q,g})$.

Step 4. This linear algebra step consists in finding a vector of the kernel in a sparse matrix of size $O(q)$, and of weight $O(gq)$; the coefficient are in $\mathbb{Z}/n\mathbb{Z}$. Hence Lanczos's algorithm provides a solution with cost $O(gq^2 c_n)$.

Step 5. This last step requires only $O(q)$ multiplications modulo n, and one inversion. Hence the complexity is $O(qc_n)$.

Finally, the overall complexity of the algorithm is $O(g!qc_J)+O((g!q+gq^2)(c_n+c_{q,g}))+O(qc_q)$. Now, by Cantor's algorithm c_J is polynomial in $g \log q$, and classical algorithm on finite fields and polynomials give c_n polynomial in $n = g \log q$,

c_q polynomial in $\log q$ and $c_{q,g}$ polynomial in $g \log q$. Hence all these operations can be done in time bounded by a polynomial in $g \log q$.

Theorem 1 *The algorithm requires $O(q^2 + g!q)$ polynomial time operations in $g \log q$ and if one considers a fixed genus g, the algorithm takes time $O(q^2 \log^\gamma q)$.*

4 Using Automorphisms on the Curve

4.1 Curves with Automorphisms in the Literature

When buiding a cryptosystem based on a hyperelliptic curve, it is preferable to know the order of the Jacobian of this curve. Indeed, some protocols use the group order; moreover it is necessary to be sure that it is not smooth. For elliptic curves, the Schoof-Elkies-Atkin algorithm allows to compute quickly this order for random curves (see [29] [28] [22]). For random hyperelliptic curves, a similar polynomial time algorithm exists [33], however it is still unusable in practice (see recent progress on this subject [21] [43]). That is the reason why the curves that we can find in the literature are very particular: they are built in such a way that the order of their Jacobian is easy to compute.

A first way to build such curves is to take a curve defined over a small finite field $\mathbb{F}_q$. It is then possible to deduce the Zeta function (and hence the order) of the Jacobian on the large field $\mathbb{F}_{q^n}$ from the Zeta function of the Jacobian on the small field. This construction provides then the so-called Frobenius automorphism defined by $x \mapsto x^q$, which can be applied to each coordinate of a point of the curve and gives therefore an automorphism of order n.

Another construction, which is a bit harder than the previous (see [42] [7] [3], comes from the theory of complex multiplication. This theory allows to build a curve starting from its ring of endomorphisms. In some cases, this ring contains units of finite order, and then there is an automorphism on the curve corresponding to this unit.

In table 1 we give some examples of curves found in the literature with non trivial automorphisms, and the order obtained by combining them together with the hyperelliptic involution.

Table 1. Examples of curves

Author	Equation of curve	Field	Automorphisms	Order
Koblitz [24], [25]	$Y^2 + Y = X^{2g+1} + X$	$\mathbb{F}_{2^n}$	Frobenius	$2n$
	$Y^2 + Y = X^{2g+1}$	$\mathbb{F}_{2^n}$	Frobenius	$2n$
Buhler Koblitz [3] Chao et al. [7]	$Y^2 + Y = X^{2g+1}$ (and twists)	$\mathbb{F}_p$ with $p \equiv 1\,(2g+1)$	mult by ζ_{2g+1}	$2(2g+1)$
Sakai Sakurai [36] Smart [41]	$Y^2 + Y = X^{13} + X^{11} + X^9 + X^5 + 1$	$\mathbb{F}_{2^{29}}$	Frobenius and $\begin{cases} X \mapsto X+1 \\ Y \mapsto Y + X^6 + X^5 + X^4 + X^3 + X^2 \end{cases}$	4×29
Duursma Sakurai [10]	$Y^2 = X^p - X + 1$	$\mathbb{F}_{p^n}$	Frobenius and $\begin{cases} X \mapsto X+1 \\ Y \mapsto Y \end{cases}$	$2np$

4.2 Reducing the Factor Basis with an Automorphism

In the context of the Pollard's rho algorithm, the existence of an automorphism of order m that can be quickly evaluated can be used to divide the expected running time by a factor $\sqrt{m}$, see [9]. With our algorithm, the automorphism can be used to reduce the basis and leads to a speed-up by a factor m^2, which can be very significant in practice. Moreover, the automorphism does not need to be so quickly evaluated as in the rho method. A polynomial time evaluation is enough.

The idea is to keep in the factor basis one representative for each orbit under the action of the automorphism. Thus the size of the basis is reduced by a factor m, so the necessary number of relations is reduced by the same factor, and the linear algebra phase is speeded up by a factor m^2. Let us explain how it works.

For the moment, assume that the Jacobian is cyclic of prime order $n = \mathrm{ord}(D_1)$, and denote by σ an automorphism of order m on $\mathcal{C}$ extended by linearity to an automorphism of $Jac(\mathcal{C})$. Then $\sigma(D_1)$ belongs to $Jac(\mathcal{C}) = \langle D_1 \rangle$, and there exists an integer θ such that $\sigma(D_1) = \theta D_1$. Moreover, σ being a group automorphism, for all $D \in Jac(\mathcal{C})$, $D = kD_1$ and we have $\sigma(D) = \sigma(kD_1) = k\sigma(D_1) = k\theta D_1 = \theta D$.

Suppose now that we have only kept in the basis one element for each orbit under σ. Let $R = P_1 + P_2 + \cdots + P_k = \alpha D_1 + \beta D_2$ be the decomposition of a smooth divisor into prime divisors of degree 1. For each i, there is a power of σ such that the prime divisor P_i is equal to $\sigma^{l_i}(g_i)$, where g_i is an element of the reduced factor basis. Then we can write $R = \theta^{l_1}(g_1) + \cdots + \theta^{l_k}(g_k)$, and we have a relation in a matrix with m times less columns than the original one.

For the general case where the Jacobian is not cyclic and where we work in a subgroup of prime order n, we have to work a little to justify the computations, but in practice we do essentially the same.

5 Implementation and Results

We have implemented the algorithm in two distinct parts. The first one deals with the building of the matrix and is written in the computer algebra system Magma [2], which is a very good compromise between high level programming and efficiency. The second part is our optimized implementation of the Lanczos algorithm written in C.

5.1 Implementation of the Search for Relations

This part of the implementation was not optimized: it can be done in parallel and it is not the limiting phase. However an interesting optimization suggested by François Morain has been tested. It is based on a paper by Swan [44], where a theorem is given which relates the parity of the number of irreducible factors of a polynomial over a finite field and the fact that its discriminant is a square or not in the corresponding local field. In the context of smoothness testing, a

first computation can be done that tests if the discriminant is a square, and then in half the cases we know that the polynomial cannot split completely and we reject it. If the first test is passed, we do the classical smoothness test by DDF.

This technique provides a gain if and only if Swan's test costs less than half the time of the classical one. In odd characteristic, this is always the case (for large q), but in characteristic 2, the running time estimation is harder because some computations have to be done over an extension of $\mathbb{Z}/8\mathbb{Z}$ and no package exists that provides optimized code for this ring. Note that the complications for the even characteristic is not surprising because in the finite field $\mathbb{F}_{2^n}$ every element is a quadratic residue and it is not simple to have a practical translation of Swan's theorem .

In our implementation, the use of Swan's theorem gave us a speed-up of 30 to 40% for the smoothness test in odd characteristic, but no improvement for characteristic 2.

5.2 Implementation of the Linear Algebra

A critical step in the algorithm is the search of a vector in the kernel of a sparse matrix. We chose Lanczos's algorithm in preference to Wiedemann's, because it needs only $2n$ products of the matrix by a vector, to be compared to $3n$ with Wiedemann's technique. The drawback is a non negligible amount of time spent in computing some scalar products. We refer to [27] for a precise comparison of these two algorithms.

We wrote our program in the C language, using the ZEN library [6] for things which were not critical (i.e. operations that are called a linear number of times), and for others (i.e. operations in the matrix-vector multiplication and scalar products), we used direct calls to some assembly routines taken from the GMP [18] and BigNum [20] packages. Indeed our compact representation of the matrix led to an overcost when using the ZEN functions. We used a classical representation (we could probably obtain a better efficiency with Montgomery representation), with the lazy reduction technique explained in [8].

Before running Lanczos's algorithm, a preprocessing can be done on the matrix (see [8] [5]). This filtering step (also called structured Gaussian elimination) consists in the following tasks:

- Delete the empty columns.
- Delete the columns with exactly one term and the corresponding row.
- If the number of rows is greater than the number of columns plus one, delete one row (randomly chosen, or via an heuristic method).
- Try the beginning of a Gaussian elimination, where the pivot is chosen as to minimize the augmentation of the weight of the matrix, and stopping when it increases the cost of Lanczos's algorithm.

For the examples below, we have run only the first three tasks, our implementation of the last one being unsatisfactory. Therefore there is still some place for further optimizations.

5.3 Timings for Real Life Curves

The first example is a cryptosystem recently proposed by Buhler and Koblitz [3]. We took the values recommended by Koblitz in his book [26], i.e. we have worked on the curve $y^2 + y = x^{13}$, with a prime base field of order p greater than $5,000,000$, with $p \equiv 1 \bmod 13$. This curve has an automorphism of order 13 coming from complex multiplication, which helps in the computation of the order of the Jacobian, but helps also our attack.

The following table gives precise information on that curve.

field	$\mathbb{F}_{5026243}$
equation	$y^2 + y = x^{13}$
genus	6
$\#J$	$13^3 \times 7345240503856807663632202049344834001 \approx 10^{40}$

We give the measured timings for the computation of a discrete logarithm in the following table. These timings are on a Pentium II 450 MHz with 128 Mb. During the Lanczos's step (the most space consuming part of the algorithm), the memory used was around 60Mb.

cardinal of factor basis	$193,485$
time for building the basis	1638 sec
number of random steps	$201,426,284$
number of early abort by Swan	$100,721,873$
number of relations collected	$281,200$
proportion of smooths ($g!$)	716.3 (720)
total time for collecting the relations	$513,870$ sec = **6 days**
time for writing relations on the basis	$8,822$ sec
time for preprocessing the matrix	1218 sec
size of the matrix	$165,778 \times 165,779$
total time for Lanczos	$780,268$ sec = **9 days**

Our algorithm is not dependent on the characteristic of the base field. We have tested our implementation on a genus 6 curve over $\mathbb{F}_{2^{23}}$. This curve was obtained by extending the scalars of a curve defined over $\mathbb{F}_2$. Therefore the Frobenius automorphism can be used for accelerating the attack. The size of the Jacobian is around 10^{41}. Such a curve is not breakable by a parallel collision search based on the birthday paradox (variants of Rho); indeed even using the automorphism, we should compute about 2^{63} operations in the Jacobian.

We give the same indications as for the previous curve.

field	$\mathbb{F}_{2^{23}}$
equation	$y^2 + (x+1)y = x^{13} + x^{11} + x^8 + x^7 + x^5 + x^4 + x + 1$
genus	6
$\#J$	$2^3 \times 7 \times 6225718452117034383550124899048999495177 \approx 10^{41}$

cardinal of factor basis	$182,462$
time for building the basis	6575 sec
number of random steps	$165,732,450$
number of relations collected	$231,000$
proportion of smooths ($g!$)	717.5 (720)
total time for collecting the relations	$797,073$ sec = **9 days**
time for writing relations on the basis	$12,057$ sec
time for preprocessing the matrix	880 sec
size of the matrix	$162,873 \times 162,874$
total time for Lanczos	$1,038,534$ sec = **12 days**

6 Conclusion

We have proposed an algorithm for the hyperelliptic discrete log problem, which is simpler to implement and to analyze than the previous ones. It is specially well suited for practical cryptosystems where the genus is not too large (say less than 9), and the base field is relatively small. Indeed the expected running time is $O(q^2)$ for curves of small genus and therefore it is faster than Pollard Rho as soon as the genus is greater than 4, as explained in the following table:

g	1	2	3	4	5	6	7
Rho	$q^{1/2}$	q	$q^{3/2}$	q^2	$q^{5/2}$	q^3	$q^{7/2}$
Index	q^2	q^2	q^2	q^2	q^2	q^2	q^2

Practical experiments have shown that this algorithm is efficient in practice, and a genus 6 example was broken by this technique. Hence it seems that there is no point in using hyperelliptic cryptosystem with genus other than 2, 3 or 4, because for a higher genus, the size of the key has to be chosen larger in order to guarantee a given level of security. Indeed, assume that we want to have a key of size 2^{160}, i.e. a group of order $\approx 2^{160}$, then we have to choose $g \log q \approx 160$. Increasing g implies decreasing $\log q$ and helps the attack. Hence one of the interests of the use of hyperelliptic curves, which was to decrease the size of q (for example to avoid multiprecision) becomes a weakness.

The special case of genus 4 has to be further studied. In a first approximation the complexity of Rho and our algorithm seem similar, but one trick can be played. We can decide to keep only a fraction of the divisors in the factor basis. Assume that we reduce the basis by a factor n. Then the probability to get a good divisor in the random walk is reduced by a factor n^g, and the cost of the first phase of the algorithm increases by a factor n^{g-1}, whereas the linear algebra is reduced by a factor n^2. In this context, Robert Harley pointed out to us that if we assume that the factorization of polynomials can be done in polynomial time (true in practice), we can balance both phases and choose n in order to get an overall complexity of $O(q^{\frac{2g}{g+1}})$. For $g = 4$, it becomes $O(q^{8/5})$, which is

better than the complexity of the Rho method. We are going to do practical comparisons between the two approaches in a near future.

From a theoretical point of view, we can also analyse our algorithm in the same model as for ADH algorithm, i.e. we assume that the genus grows with q and is always large enough. More precisely, if we have $g > \log q$, we can let vary the smoothness bound S (instead of have it fixed to one), and we obtain a subexponential algorithm with expected running time $L_{q^g}[1/2, \sqrt{2}]$. This result is part of a work with Andreas Enge, where a general framework for this kind of attack is given [12].

Acknowledgements

I am most indebted to François Morain for many fruitful discussions and comments. I would like to thank Emmanuel Thomé, particularly for his help for linear algebra. I am also grateful to Robert Harley and people from the Action Courbes (particularly Daniel Augot) for many discussions concerning this work.

References

1. L. M. Adleman, J. DeMarrais, and M.-D. Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the jacobians of large genus hyperelliptic curves over finite fields. In L. Adleman and M.-D. Huang, editors, *ANTS-I*, volume 877 of *Lecture Notes in Comput. Sci.*, pages 28–40. Springer–Verlag, 1994. 1st Algorithmic Number Theory Symposium - Cornell University, May 6–9, 1994.
2. W. Bosma and J. Cannon. *Handbook of Magma functions*, 1997. Sydney, `http://www.maths.usyd.edu.au:8000/u/magma/`.
3. J. Buhler and N. Koblitz. Lattice basis reduction, Jacobi sums and hyperellitic cryptosystems. *Bull. Austral. Math. Soc.*, 58:147–154, 1998.
4. D. G. Cantor. Computing in the Jacobian of an hyperelliptic curve. *Math. Comp.*, 48(177):95–101, 1987.
5. S. Cavallar. Strategies in filtering in the Number Field Sieve. Extended abstract, conference MPKC, Toronto, June 1999.
6. F. Chabaud and R. Lercier. *ZEN, A new toolbox for computing in finite extensions of finite rings*, February 1998. distributed with the ZEN package at `http://www.dmi.ens.fr/~zen`.
7. J. Chao, N. Matsuda, O. Nakamura, and S. Tsujii. Cryptosystems based on CM abelian variety. In *Proc. Symposium on Cryptography and Information Security*, 1997.
8. T. Denny and D. Weber. The solution of McCurley's discrete log challenge. In H. Krawczyk, editor, *Proc. of CRYPTO'98"*, volume 1462 of *Lecture Notes in Comput. Sci.*, pages 458–471, 1998.
9. I. Duursma, P. Gaudry, and F. Morain. Speeding up the discrete log computation on curves with automorphisms. In K.Y. Lam, E. Okamoto, and C. Xing, editors, *Advances in Cryptology – ASIACRYPT '99*, volume 1716 of *Lecture Notes in Comput. Sci.*, pages 103–121. Springer-Verlag, 1999. International Conference on the Theory and Applications of Cryptology and Information Security, Singapore, November 1999, Proceedings.

10. I. Duursma and K. Sakurai. Efficient algorithms for the jacobian variety of hyperelliptic curves $y^2 = x^p - x + 1$ over a finite field of odd characteristic p. In *Proceedings of the "International Conference on Coding Theory, Cryptography and Related Areas"*, Lecture Notes in Comput. Sci., 1999. Guanajuato, Mexico on April, 1998.
11. A. Enge. Computing discrete logarithms in high-genus hyperelliptic jacobians in provably subexponential time. Preprint; available at `http://www.math.uwaterloo.ca/CandO_ Dept/CORR/corr99.html`, 1999.
12. A. Enge and P. Gaudry. A general framework for subexponential discrete logarithm algorithms. In preparation, 1999.
13. R. Flassenberg and S. Paulus. Sieving in function fields. Preprint; available at `ftp://ftp.informatik.tu-darmstadt.de/pub/TI/TR/TI-97-13.rafla.ps.gz`, 1997.
14. G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. Comp.*, 62(206):865–874, April 1994.
15. W. Fulton. *Algebraic curves.* Math. Lec. Note Series. W. A. Benjamin Inc, 1969.
16. S. D. Galbraith and N. Smart. A cryptographic application of Weil descent. Preprint HP-LABS Technical Report (Number HPL-1999-70)., 1999.
17. R. Gallant, R. Lambert, and S. Vanstone. Improving the parallelized Pollard lambda search on binary anomalous curves. `http://www.certicom.com/chal/download/paper.ps`, 1998.
18. T. Granlund. *The GNU Multiple Precision arithmetic library – 2.0.2.* GNU, 1996. distributed with the gmp package at `ftp://prep.ai.mit.edu/pub/gnu/gmp-M.N.tar.gz`.
19. J. L. Haffner and K. S. McCurley. A rigorous subexponential algorithm for computation of class groups. *J. Amer. Math. Soc.*, 2(4):837–850, 1989.
20. J.-C. Hervé, B. Serpette, and J. Vuillemin. BigNum: A portable and efficient package for arbitrary-precision arithmetic. Technical Report 2, Digital Paris Research Laboratory, May 1989.
21. M.-D. Huang and D. Ierardi. Counting points on curves over finite fields. *J. Symbolic Comput.*, 25:1–21, 1998.
22. T. Izu, J. Kogure, M. Noro, and K. Yokoyama. Efficient implementation of Schoof's algorithm. In K. Ohta and D. Pei, editors, *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Comput. Sci.*, pages 66–79. Springer-Verlag, 1998. International Conference on the theory and application of cryptology and information security, Beijing, China, October 1998.
23. N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48(177):203–209, January 1987.
24. N. Koblitz. Hyperelliptic cryptosystems. *J. of Cryptology*, 1:139–150, 1989.
25. N. Koblitz. A family of jacobians suitable for discrete log cryptosystems. In S. Goldwasser, editor, *Advances in Cryptology – CRYPTO '88*, volume 403 of *Lecture Notes in Comput. Sci.*, pages 94–99. Springer–Verlag, 1990. Proceedings of a conference on the theory and application of cryptography held at the University of California, Santa Barbara, August 21–25, 1988.
26. N. Koblitz. *Algebraic aspects of cryptography*, volume 3 of *Algorithms and Computation in Mathematics.* Springer–Verlag, 1998.
27. B. A. LaMacchia and A. M. Odlyzko. Solving large sparse linear systems over finite fields. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology*, volume 537 of *Lecture Notes in Comput. Sci.*, pages 109–133. Springer–Verlag, 1990. Proc. Crypto '90, Santa Barbara, August 11–15, 1988.

28. R. Lercier. *Algorithmique des courbes elliptiques dans les corps finis.* Thèse, École polytechnique, June 1997.
29. R. Lercier and F. Morain. Counting the number of points on elliptic curves over finite fields: strategies and performances. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology – EUROCRYPT '95*, volume 921 of *Lecture Notes in Comput. Sci.*, pages 79–94, 1995. Saint-Malo, France, May 1995, Proceedings.
30. A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curves logarithms to logarithms in a finite field. *IEEE Trans. Inform. Theory*, 39(5):1639–1646, September 1993.
31. V. Miller. Use of elliptic curves in cryptography. In A. M. Odlyzko, editor, *Advances in Cryptology – CRYPTO '86*, volume 263 of *Lecture Notes in Comput. Sci.*, pages 417–426. Springer-Verlag, 1987. Proceedings, Santa Barbara (USA), August 11–15, 1986.
32. V. Müller, A. Stein, and C. Thiel. Computing discrete logarithms in real quadratic congruence function fields of large genus. *Math. Comp.*, 68(226):807–822, 1999.
33. J. Pila. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Math. Comp.*, 55(192):745–763, October 1990.
34. J. M. Pollard. Monte Carlo methods for index computation mod p. *Math. Comp.*, 32(143):918–924, July 1978.
35. H. G. Rück. On the discrete logarithm in the divisor class group of curves. *Math. Comp.*, 68(226):805–806, 1999.
36. Y. Sakai and K. Sakurai. Design of hyperelliptic cryptosystems in small charatcteristic and a software implementation over $\mathbb{F}_{2^n}$. In K. Ohta and D. Pei, editors, *Advances in Cryptology*, volume 1514 of *Lecture Notes in Comput. Sci.*, pages 80–94. Springer–Verlag, 1998. Proc. Asiacrypt '98, Beijing, October, 1998.
37. T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Comment. Math. Helv.*, 47(1):81–92, 1998.
38. I. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curves in characteristic p. *Math. Comp.*, 67(221):353–356, January 1998.
39. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT '97*, volume 1233 of *Lecture Notes in Comput. Sci.*, pages 256–266. Springer–Verlag, 1997. International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 1997, Proceedings.
40. N. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. of Cryptology*, 12(3):193–196, 1999.
41. N. Smart. On the performance of hyperelliptic cryptosystems. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Comput. Sci.*, pages 165–175. Springer–Verlag, 1999. International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 1999, Proceedings.
42. A.-M. Spallek. *Kurven vom Geschlecht 2 und ihre Anwendung in Public-Key-Kryptosystemen.* PhD thesis, Universität Gesamthochschule Essen, July 1994.
43. A. Stein and E. Teske. Catching kangaroos in function fields. Preprint, March 1999.
44. R. G. Swan. Factorization of polynomials over finite fields. *Pacific J. Math.*, 12:1099–1106, 1962.
45. E. Teske. Speeding up Pollard's rho method for computing discrete logarithms. In J. P. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Comput. Sci.*, pages 541–554. Springer–Verlag, 1998. Third International Symposium, ANTS-III, Portland, Oregon, june 1998, Proceedings.

46. P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *J. of Cryptology*, 12:1–28, 1999.
47. D. H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inform. Theory*, IT–32(1):54–62, 1986.
48. M. J. Wiener and R. J. Zuccherato. Faster attacks on elliptic curve cryptosystems. In S. Tavares and H. Meijer, editors, *Selected Areas in Cryptography '98*, volume 1556 of *Lecture Notes in Comput. Sci.* Springer-Verlag, 1999. 5th Annual International Workshop, SAC'98, Kingston, Ontario, Canada, August 17-18, 1998, Proceedings.

Analysis and Optimization of the TWINKLE Factoring Device

Arjen K. Lenstra[1] and Adi Shamir[2]

[1] Citibank, N.A., 1 North Gate Road, Mendham, NJ 07945-3104, U.S.A.,
arjen.lenstra@citicorp.com
[2] Computer Science Department, The Weizmann Institute, Rehovot 76100, Israel,
shamir@wisdom.weizmann.ac.il

Abstract. We describe an enhanced version of the TWINKLE factoring device and analyse to what extent it can be expected to speed up the sieving step of the Quadratic Sieve and Number Field Sieve factoring algorithms. The bottom line of our analysis is that the TWINKLE-assisted factorization of 768-bit numbers is difficult but doable in about 9 months (including the sieving and matrix parts) by a large organization which can use 80,000 standard Pentium II PC's and 5,000 TWINKLE devices.

1 Introduction

The TWINKLE device is an optoelectronic device which is designed to speed up the sieving operation in the Quadratic Sieve (QS) and Number Field Sieve (NFS) integer factoring algorithms by using arrays of light emitting diodes (LED's) which blink at various rates (cf. [7]). The main purpose of this paper is to carry out a detailed and realistic analysis of the expected behavior of a TWINKLE-assisted factoring attempt on inputs whose binary sizes are 384, 512, and 768 bits. In particular, we describe the optimal choice of the many parameters involved in such factoring attempts, and identify several areas in which the original TWINKLE design leads to computational bottlenecks. We then propose enhanced hardware and algorithmic designs which eliminate these bottlenecks, and make such factorizations more feasible.

This paper is organized as follows. In Section 2 we briefly review the original TWINKLE design from [7]. In Section 3 we discuss the applicability of the original TWINKLE design to 384-bit numbers using the QS algorithm. In the remainder of the paper we concentrate on how TWINKLE may be used for the sieving step of the NFS for the factorization of 512-bit and 768-bit numbers. In Section 4 we briefly sketch the required NFS background, and in Section 5 we discuss the sieving step of the NFS in more detail. In Section 6 we present a number of hardware enhancements of the TWINKLE device. In Section 7 we describe how the NFS sieving step may be carried out on the modified TWINKLE device and we analyse its running time. In Section 8 we address the question what it is about TWINKLE that makes LED's necessary and comment upon the proposals to build a TWINKLE-like device using ordinary electronic circuitry.

All PC running times referred to in this paper are actual measurements rather than asymptotic or conjectured running times. They are based on optimized implementations of the various algorithms on a 450MHz Pentium II PC with 192 megabytes of RAM. However, the TWINKLE device itself has not been built so far, and thus its feasibility, performance, and cost are open to debate.

2 The Original TWINKLE Design

We recall the basics of the TWINKLE device as described in [7]. The most time consuming step of most modern factoring algorithm is the 'sieving step': for many pairs of integers (p, r) in succession add an approximation of $\log(p)$ to location $r + kp$ of a sieve interval (initialized as zeros) for all integers k such that $r + kp$ is in the interval, and report all j for which location j exceeds a certain threshold. Standard implementations use space (memory) to represent the interval, and time (clock cycles) to loop over the (p, r) pairs. The TWINKLE device reverses the roles of space and time: it uses space to represent the (p, r) pairs, and time to loop over the sieve interval. This goes as follows.

The TWINKLE device is a cylinder of 6 inch diameter and 10 inch height. The bottom consists of a single wafer of GaAs, containing one 'cell' for each different p. Each cell contains an LED, a photodetector, an A register representing the value of p, for each r corresponding to that p a B register initially loaded with a representation of r, and wiring. The top of the cylinder contains a summing photodetector that measures the total light intensity emitted by the bottom LED's and a clocking LED that distributes the clock signal by flashing at a fixed clock rate. As clock signal j is received by a cell's photodetector, the values of the B registers are decremented by one, if a resulting value represents zero the cell's LED flashes with intensity proportional to $\log(p)$, and the A register is copied to the B register representing zero. If the total light intensity detected by the top photodetector exceeds a certain threshold the value of j is reported. Further details of the original TWINKLE design, such as how integers are represented and decremented and how the optical delays are handled, can be found in [7].

The bottom wafer as proposed in [7] contains 10^5 cells with one A and two B registers each, and is clocked at 10 GHz. Since it takes a single clock cycle to sum the values corresponding to a sieve location and to report the location if necessary, this would correspond to a QS implementation that takes 10 milliseconds to inspect 10^8 integers for smoothness with respect to the primes up to $3 * 10^6$.

3 Analysis of TWINKLE-Assisted 384-bit QS Factorizations

The original description of the TWINKLE device in [7] is geared towards the QS factoring algorithm (cf. [6]). In this section we analyse the effectiveness of

the original TWINKLE device for the factorization of 384-bit numbers using the QS.

Although 384-bit numbers are unlikely to be the moduli in RSA cryptosystems, their quick factorization may be useful in various number theoretic subroutines (e.g., when we try to complete the factorization of a large number of randomly generated values after we eliminate their smooth parts). The factorization of such numbers is not particularly difficult - it can be carried out in a few months on a single PC. Our goal is simply to find out the improvement ratio between TWINKLE-assisted factorizations and PC-based factorizations. There are two reasons why such an analysis can be interesting:

- There is a great deal of experimental data on the optimal choice of parameters and the actual running time when factoring numbers of this size, and thus the comparison can be based on harder data.
- It enables us to examine the specific issues related to the implementation of the QS algorithm on the TWINKLE device. The QS and NFS algorithms exhibit quite different properties when implemented on the TWINKLE device, but for larger input sizes the QS algorithm is simply not competitive with the NFS algorithm.

We assume that the reader is familiar with the general outline of the QS algorithm (see, e.g. [7]). A sequence of quadratic polynomials $f_1, f_2, \ldots$ is generated that depends on the number to be factored and the length $2 * \mathbf{A}$ of the sieve interval. For $i = 1, 2, \ldots$ in succession the roots of the f_i modulo the primes in the factor base are computed and the values $f_i(x)$ for $-\mathbf{A} \leq x < \mathbf{A}$ are sieved to test them for smoothness. The resulting smooth values have to be post-processed, which consists of trial division possibly followed by the computation of the decomposition of the resulting cofactor. For the QS algorithm the post-processing step is negligible compared to the polynomial generation and root computation. When doing the actual sieving on the TWINKLE device, polynomial generation, root finding, and post-processing have to be carried out by one or more auxiliary PC's. The sequence of polynomials can be generated using the ordinary Multiple Polynomial variant (MPQS) or using the Self Initializing variant (SIQS).

Based on actual data, the factorization of a 384-bit number with the QS algorithm on a single 450 MHz Pentium II requires:

- About 9 months when running the QS algorithm with optimal parameters: 186,000 primes in the factor base and $2 * \mathbf{A}$ ≈1,600,000 (SIQS) or $2 * \mathbf{A}$ ≈16,000,000 (MPQS).
- About 14 months when running the QS algorithm with the suboptimal choices used in the original TWINKLE design (cf. [7]): 100,000 primes in the factor base and a sieving interval of length $2 * \mathbf{A} = 100{,}000{,}000$.

For 384-bit inputs, there is little difference between the running times of MPQS and SIQS, but SIQS can compute the roots of the f_i faster, which is a significant advantage in TWINKLE-assisted factorizations. For the optimal choice of parameters, a PC implementation spends about 25% of the time on polynomial

selection and root finding, but for the original choice (which we shall assume from now on) this fraction drops to 0.6% (about 0.09 seconds per polynomial on a PC). We consider two possible scenarios:

1. A TWINKLE device running at the maximum possible speed of 10 GHz. Each sieving interval of length 100,000,000 can be scanned in 0.01 seconds (cf. [7]). The total running time of the TWINKLE device is about 11 hours, and 9 (= 0.09/0.01) PC's are needed to generate the polynomials and to compute their roots. These 9 PC's can execute a conventional QS factorization with optimal parameters in about a month, and thus the achievable improvement ratio is approximately $30 * 24/11 \approx 65$.
2. A TWINKLE device running at the minimum recommended speed of 1 GHz (cf. 6.1). Scanning a single interval takes 0.1 seconds, and the whole scanning phase takes 110 hours or about 4.5 days. However, in this case we need only one PC to support the TWINKLE device. Thus we have to compare this execution time to the 9 months required by a single PC implementation of QS with optimal parameters. The relevant improvement ratio is thus $9 * 30 * 24/110 \approx 59$.

The surprising conclusion is that we get about the same improvement ratio regardless of whether we run the TWINKLE device at 10 GHz or at 1 GHz, since the computational bottleneck is in the supporting PC's. As described in 6.1, a 1 GHz TWINKLE is much easier to design and operate, and can make the whole idea much more practical.

The improvement ratio of about 60 refers only to application of the QS because a 384-bit number can be factored in about 2 months on a PC using the NFS (this figure is based on extrapolation of the results from [1]).

We next consider the problem of factoring 512-bit numbers, which are typical RSA keys in E-commerce applications. For this size the QS is not competitive with the asymptotically faster NFS so we concentrate on the NFS in the remainder of this article.

4 Number Field Sieve

The Number Field Sieve integer factorization algorithm consists of four main steps:

- Polynomial selection;
- Sieving;
- Matrix processing;
- Algebraic square root computation.

We briefly describe these steps as far as relevant for the description of the TWINKLE device. Let n be the number to be factored. For ease of exposition we assume that n is a 512-bit number.

4.1 Polynomial Selection

In the first step of the NFS factorization of n two polynomials of degrees 5 and 1 with a common root modulo n are selected:

$$f_1(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \in \mathbf{Z}[x]$$

and

$$f_2(x) = x - m \in \mathbf{Z}[x] \ ,$$

where $f_1(m) \equiv 0 \bmod n$. It is advantageous if the a_i and m are small in absolute value and if f_1 has relatively many roots modulo small primes. The best known method to find such polynomials (cf. [5]) produces an m that is of the same order of magnitude as $n^{1/6}$ and a polynomial f_1 that is *skew*, i.e., $|a_5| \ll |a_4| \ll |a_3| \ll |a_2| \ll |a_1| \ll |a_0|$. The *skewness ratio* s of f_1 approximates the average ratio $|a_i|/|a_{i+1}|$. A realistic value for s is 10^4. The bivariate, homogeneous, integral polynomials F_1 and F_2 are defined as

$$F_1(x,y) = y^5 * f_1(x/y) \text{ and } F_2(x,y) = y * f_2(x/y) \ .$$

Everything related to f_1 or F_1 is referred to as the *algebraic side*, as opposed to the *rational side* for f_2 or F_2.

4.2 Sieving

In the second step, the sieving step, *relations* are sought. These are coprime pairs of integers (a,b) such that $b > 0$ and both $F_1(a,b)$ and $F_2(a,b)$ are smooth, where smoothness of $F_1(a,b)$ and $F_2(a,b)$ is defined as follows:

- $F_1(a,b)$ factors over the primes $< 2^{24}$, except for possibly three primes $< 10^9$;
- $F_2(a,b)$ factors over the primes $< 2^{24}$, except for possibly two primes $< 10^9$.

Thus, three large primes are allowed on the algebraic side, but only two large primes are allowed on the rational side. There are about one million primes $< 2^{24}$, more precisely $\pi(2^{24})$ =1,077,871.

Candidate pairs (a,b) are located by twice sieving with the primes $< 2^{24}$ over the rectangular region $-\mathbf{A} \le a < \mathbf{A}, 0 < b \le \mathbf{A}/s$, for a large $\mathbf{A}$ that is specified in 5.5. The sieving region is skew with skewness ratio s. The resulting candidate pairs are trial divided to inspect if they indeed lead to relations. How the sieving and the trial division may be carried out is addressed in the next section. It is the most time consuming step of the NFS, and it is the step for which we want to use the TWINKLE device.

4.3 Matrix Processing

Each relation a, b gives rise to a vector of exponents corresponding to the multiplicities of the primes in the factorizations of $F_1(a,b)$ and $F_2(a,b)$. It may be expected that there are many linear dependencies among these vectors after about 80 to 100 million relations have been found. Dependencies modulo 2 among the vectors are determined in the matrix processing step. How the matrix step is carried out is described in the literature referred to in [1].

4.4 Algebraic Square Root Computation

Each dependency modulo 2 leads with probability at least one half to the factorization of n in the last step of the NFS. References describing how this is done can be found in [1].

5 Sieving

So far two distinct sieving methods have been used in NFS implementations: *line sieving* and *special q sieving*. Line sieving works by sieving over the a-interval $[-\mathbf{A}, \mathbf{A})$ for each $b = 1, 2, 3, \ldots$ consecutively, until enough relations have been found. Special q sieving works by repeatedly picking an appropriate prime q between 2^{24} and, approximately, $5 * 10^8$, and by restricting the sieving to the pairs (a, b) for which q divides $F_1(a, b)$, until enough unique relations have been found. Note that a relation is found at most once by line sieving but that it may be found up to three times by special q sieving because each of the at most three algebraic large primes may be used as special q. Both sieving methods may be used, simultaneously or separately, for a single factorization. We describe both methods in more detail, paying special attention to a property of special q sieving that is not generally appreciated and that turns out to be beneficial for TWINKLE-assisted NFS sieving.

5.1 Factor Bases and Sieving Thresholds

Let for $i = 1, 2$

$$P_i = \{(p, r) : f_i(r) \equiv 0 \bmod p, \;\; p \text{ prime}, \;\; p < 2^{24}, \;\; 0 \le r < p\} \; .$$

The set P_2, the *rational factor base*, consists of the pairs $(p, m \bmod p)$ for all primes $p < 2^{24}$ and is trivial to compute. For the computation of P_1, the *algebraic factor base*, the roots of $f_1 \bmod p$ have to be determined for all primes $p < 2^{24}$. The number of times a particular prime $p < 2^{24}$ occurs in a (p, r) pair in P_1 may be 0, 1, 2, 3, 4, or 5. The sets P_1 and P_2 are computed once. Let T_1 and T_2 be two threshold values.

5.2 Line Sieving

For $b = 1, 2, 3, \ldots$ in succession do the following.

> For $i = 1, 2$ in succession, initialize the sieve locations $S_i(a)$ to zero for $-\mathbf{A} \le a < \mathbf{A}$, and next for all (p, r) in P_i replace $S_i(br + kp)$ by $S_i(br + kp) + \log(p)$ for all integers k such that $br + kp \in [-\mathbf{A}, \mathbf{A})$. Finally, for all a such that $\gcd(a, b) = 1$, $S_1(a) > T_1$, and $S_2(a) > T_2$ inspect if both $F_1(a, b)$ and $F_2(a, b)$ are smooth.

For a fixed b-value the a-interval $-\mathbf{A} \le a < \mathbf{A}$ is referred to as a *line*. Note that line sieving uses $\#P_1 + \#P_2$ arithmetic progressions per line, i.e., per b-value.

5.3 Special Sieving

For pairs (q, r_q) with $f_1(r_q) \equiv 0 \bmod q$, q prime, and $2^{24} < q < 10^9$ do the following.

> Let L_q be the lattice spanned by the two 2-dimensional vectors $(q,0)^T$ and $(r_q,1)^T$. Sieving over $L_q \cap \{(a,b)^T : -\mathbf{A} \le a < \mathbf{A}, 0 < b \le \mathbf{A}/s\}$, i.e., 'small' (a,b) pairs for which q divides $F_1(a,b)$, is approximated as follows. Find a basis $(x_1,y_1)^T$, $(x_2,y_2)^T$ for L_q for which $|x_i| \approx |s * y_i|$. Let V_q be the subset of L_q consisting of the vectors $d*(x_1,y_1)^T + e*(x_2,y_2)^T$ for integers d,e with $-\mathbf{A}/(s*q)^{1/2} \le d < \mathbf{A}/(s*q)^{1/2}$, $0 < e \le \mathbf{A}/(s*q)^{1/2}$ (although in practice one sieves over the same d,e values for all pairs (q,r_q)). For $i = 1,2$ in succession, initialize the sieve locations $S_i(v)$ to zero for all v in V_q, and next for all (p,r) in P_i replace $S_i(v)$ by $S_i(v) + \log(p)$ for all v in V_q that can be written as an integer linear combination of $(p,0)^T$ and $(r,1)^T$. Finally, for all $v = (a,b)^T$ in V_q for which $\gcd(a,b) = 1, S_1(v) > T_1 - \log(q)$, and $S_2(v) > T_2$ inspect if both $F_1(a,b)$ and $F_2(a,b)$ are smooth.

In this case a line is the d-interval for a fixed e-value. It follows from the asymptotic values of $\mathbf{A}$ and p (cf. [3]) that a particular line (e-value) is not hit (in the d-interval) by the majority of pairs (p,r). Using arithmetic progressions for all (p,r) pairs per e-value would therefore increase the asymptotic running time of NFS, i.e., it is too expensive to visit all $\mathbf{A}/(s*q)^{1/2}$ e-values for all $(p,r) \in P_1 \cup P_2$. Instead, per (p,r) only $O(\mathbf{A}^2/(s*q*p))$ steps may be performed for the sieving over V_q. In [2] this problem is adequately solved by *lattice sieving* for each (p,r) pair, as proposed by Pollard (cf. his second article in [3]). Although the TWINKLE device may appear to solve the problem by processing all (p,r) pairs simultaneously for each sieve location, per line the initial B registers still have to be loaded for each (p,r) pair, which is obviously too expensive. This problem and its consequences are discussed in more detail in 7.2.

5.4 Trial Divisions

For reasonable choices of T_1 and T_2 the number of pairs (a,b) for which $F_1(a,b)$ and $F_2(a,b)$ have to be inspected for smoothness is so large that straightforward trial division with the primes in P_1 and P_2 would take considerably more time than the sieving. Trial division is therefore too time consuming. Instead, in PC implementations the primes dividing $F_1(a,b)$ and $F_2(a,b)$ are found by first repeating the sieving step in a somewhat altered fashion, next performing divisions of $F_1(a,b)$ and $F_2(a,b)$ by the primes found by this resieving, and finally factoring the resulting cofactors if they are composite and sufficiently small. For line sieving the F_1-cofactor may have to be factored into three factors, whereas for special q sieving two factors suffice. Cofactor factorization is thus substantially easier for special q sieving than for line sieving. For the line sieving in [1] this problem was avoided by using a substantially larger algebraic factor base and by allowing only two additional (large) primes in the factorization of $F_1(a,b)$.

This is illustrated by the following actual timings. For smoothness as defined here, the divisions and cofactor factorizations for line sieving cost about 0.7 seconds on a PC per resulting relation, including the time spent on the false reports but not including the time spent on the resieving. For special q sieving it goes more than 6 times faster, i.e., about 0.1 seconds per resulting relation, due to the easier factorizations and considerably smaller number of false reports. Between 80 to 100 million relations are needed (cf. 4.3), so that with line sieving only one may expect to spend more than two years on a single PC to process the reports resulting from sieving and resieving. For special q sieving this can be reduced to about 5 months.

Therefore, a line sieving TWINKLE device needs to be supported by about $7 * 10^7$ seconds on a PC to perform the divisions by the primes found by the resieving plus the cofactor factorizations. That is about 12.5% of the total PC sieving time reported in [1]. For a special q sieving TWINKLE device other considerations come into play, as shown in 7.2.

5.5 Sieving Regions

For a 512-bit n sufficiently many good pairs (a, b) can be expected if special q sieving is done for $-2^{12} \leq d < 2^{12}$ and $0 < e \leq$5,000, for all (q, r_q) pairs with $2^{24} < q < 5 * 10^8$. A gross over-estimate for the required line sieving effort follows by taking $q = 5 * 10^8$ and $\mathbf{A} = 2^{12} * (s * q)^{1/2}$. Thus $\mathbf{A} = 9 * 10^9$, i.e., $9 * 10^5$ lines of length $1.8 * 10^{10}$ each, should suffice for line sieving. The number of points sieved would be about $17 * 10^{15}$ for line sieving, but only about 10^{15} for special q sieving (where we use that $\pi(5 * 10^8) =$ 26,355,867). PC implementations of the NFS exclude the 'even, even' locations from the sieve, so on PC's the numbers of points sieved are 25% lower.

6 Hardware Enhancements

In this section we address most of the potential problems in the original TWINKLE paper which were pointed out by hardware designers and factoring experts. The result is a simpler, better, and more practical factoring device. More particularly, we address the following issues:

1. Clock rate;
2. Power consumption;
3. Avoiding trial division;
4. Using separate algebraic and rational LED's;
5. Geometric considerations.

The type of sieving is left unspecified (cf. Section 7). We assume familiarity with the design from [7] as reviewed in Section 2: cells with A registers for the primes, B registers for the counters, a photodetector for the clock signal, and LED's for the flashing.

6.1 Clock Rate

In [7] Shamir assumed that a TWINKLE device can be designed to run at 10 GHz. This speed represents the limit to which currently available GaAs technology can be pushed. However, it is not clear that our ability to produce at great cost a single laser diode which can switch at such speeds can be duplicated in mass produced wafer scale designs. Such a clock rate should thus be viewed as ambitious and speculative, but not as revolutionary as the construction of a quantum factoring computer.

On the other hand, devices made with the slower CMOS technology already run at clock rates exceeding 700 MHz. We can thus reasonably assume that a TWINKLE device built today can run at clock rates exceeding 1 GHz, and that a TWINKLE device built 5 to 10 years from now can run at clock rates approaching 10 GHz. However, as demonstrated in Section 3 the speed issue can be irrelevant since the achievable speedup ratio can be independent of the actual speed of the TWINKLE device.

6.2 Power Consumption

Several experienced hardware designers objected to the original TWINKLE design, claiming that it would consume too much power, which could lead to a wafer meltdown. Since the power consumption grows linearly with the clock rate, a 10-fold reduction of the recommended clock rate can greatly reduce this problem.

Even greater power reduction can be obtained by using a different cell design. The total power consumption of all the LED's is negligible, since at most a few hundred out of the 100,000 LED's can flash at any given time. The total power consumption of all A registers is also negligible, since they change their state only once per sieving interval. Almost all the power consumed by the wafer is used to change the state of the bits in the B registers which count the number of clock cycles. The original TWINKLE design implemented the counters as linear feedback shift registers. Such a counter design eliminates the carry propagation problem and makes the flashes highly synchronized, but it consumes a lot of power since each bit in the counter changes state every second clock cycle on average.

To reduce the power consumption, we now propose a different design. It is based on an asynchronous ripple counter in which the clock signal is fed only to the least significant bit, and the ith bit changes state only once every 2^i clock cycles. As a result, most of the bits in the counter can operate at slow speed, and the average power consumption is a small constant which is independent of the length of the counter.

The LED can be flashed when the most significant bit changes state from 0 to 1. This eliminates the tree of AND's in the original design, but it can take a long time (several clock cycles) for the clock to ripple through the register when state "0111...111" changes to "1000...000". A 10% difference in the switching speeds of two counters can lead to flashes which are almost a full clock cycle

apart, leading to incorrect results. A simple solution to this problem is based on the observation that the timing of the least significant bits is likely to be much more precise than the timing of the most significant bits. Assume that the maximum propagation delay across the register is between 0 and 15 clock cycles. We derive the flashing signal by AND'ing the most significant bit and the fifth least significant bit. Regardless of when the former bit turns to "1", the flash will occur when the latter is turned to "1". Since we reload the B register (in parallel) shortly afterwards, this AND condition will not reoccur until the end of the next cycle.

6.3 Avoiding Trial Division

The analog nature of the TWINKLE device implies that each reported smoothness event has to be confirmed and turned into an actual vector of prime exponents. The original TWINKLE design assumed that such events will be so rare that the host PC will use trial division with 100,000 possible primes to accomplish this. For the QS algorithm this assumption is correct, as mentioned in Section 3. However, as mentioned in 5.4 this is a potential bottleneck for the NFS.

In this section we describe a small modification of the TWINKLE design which can greatly simplify this task. The basic idea is to use the optical photodetector in order to detect that a large number of primes seem to divide the current value, and to use the parallel electronic I/O lines on the wafer to report their identities with a proper encoding technique. The PC only has to perform trial division by about 50 known primes rather than trial division by all primes in the factor bases. The I/O lines are used to load the A and B registers for the new sieving interval, and are idle during the actual sieving. However, these are long high capacitance wires which cannot continuously report the identity of the flashing LED's at each clock cycle. The solution is to make sure that reports will be generated only when the photodetector senses a possible smoothness event, and only by the approximately 50 relevant cells.

To achieve this, we add an optical feedback path from the photodetector to the cells. When the light measured by the photodetector exceeds the threshold, it flashes a query LED placed next to it (and opposite the wafer). Each cell has an additional photodetector for the query LED. When this query LED is sensed, each cell checks whether it flashed its own LED a certain number of clock cycles ago (depending on the total delay along the optical path), and if so, reports its identity on the I/O lines.

The simplest way of implementing this idea is to separate the flashing of the LED and the reloading of the counter in each cell. Assume for example that each B register is a ripple counter which flashes its LED when it reaches state "10...010000" (cf. 6.2). It continues to count upwards, reports its identity if the query LED is sensed AND its state is "10...011000", and reloads itself from register A when its state reaches "10...011001". The value of the A register has to be augmented to compensate for this delay, and different wavelengths have

to be used for the various LED's and photodetectors to avoid confusion between the various optical functions.

6.4 Using Separate Algebraic and Rational LED's

In the QS about half the primes up to a bound **B** do not yield arithmetic progressions, and the other half generate two distinct arithmetic progressions. This implies that in the original TWINKLE design a single cell contains one A register for a prime, two B registers for the arithmetic progressions, and one LED that flashes if either B register enters a special state. For QS with factor base bound **B** one may therefore expect $\pi(\mathbf{B})$ arithmetic progressions generated by $\pi(\mathbf{B})/2$ cells with 3 registers (a single A and two B), and one LED per cell.

NFS requires a different cell design. If distinct cells are feasible we show that the same average number of registers per cell (namely 3) can be achieved as in QS. Let $\mathbf{B} = 2^{24}$ (cf. 4.2). All primes less than **B** are potential divisors of $F_2(a,b)$. Thus, at least $\pi(\mathbf{B})$ different cells, each with at least an A register, a B register, and an LED, are needed for the resulting $\pi(\mathbf{B})$ rational arithmetic progressions. For $F_1(a,b)$ the number of arithmetic progressions required for a certain prime $p < \mathbf{B}$ depends on the number of distinct roots of $f_1 \bmod p$. On average one may expect that for

- 11/30 of the primes $f_1 \bmod p$ does not have a root;
- 3/8 of the primes $f_1 \bmod p$ has a single root;
- 1/6 of the primes $f_1 \bmod p$ has two distinct roots;
- 1/12 of the primes $f_1 \bmod p$ has three distinct roots; and
- 1/120 of the primes $f_1 \bmod p$ has five distinct roots.

(Note that 11/30+3/8+1/6+1/12+1/120 = (44+45+20+10+1)/120 = 1.) Let $p < \mathbf{B}$ be a prime for which $f_1 \bmod p$ has d distinct roots. This p requires d+1 distinct arithmetic progressions which can be taken care of by a single cell with d+2 registers: a single A register for p, a single B register for the rational arithmetic progression, and d different B registers for the d distinct algebraic arithmetic progressions. Here we use that unless n has a small factor, p cannot divide both $F_1(a,b)$ and $F_2(a,b)$, so that the rational arithmetic progression is different from the algebraic ones. This leads to a total of $\pi(\mathbf{B})$ cells: $11 * \pi(\mathbf{B})/30$ with 2 registers, $3 * \pi(\mathbf{B})/8$ with 3 registers, $\pi(\mathbf{B})/6$ with 4 registers, $\pi(\mathbf{B})/12$ with 5 registers, and $\pi(\mathbf{B})/120$ with 7 registers. The total number of registers is $(2 * 11/30 + 3 * 3/8 + 4/6 + 5/12 + 7/120) * \pi(\mathbf{B}) = (88 + 135 + 80 + 50 + 7) * \pi(\mathbf{B})/120 = 3 * \pi(\mathbf{B})$. The expected number of arithmetic progressions equals $\pi(\mathbf{B}) + (3/8 + 2/6 + 3/12 + 5/120) * \pi(\mathbf{B}) = 2 * \pi(\mathbf{B})$. Thus, for NFS with factor base bounds **B** one may expect $2 * \pi(\mathbf{B})$ arithmetic progressions generated by $\pi(\mathbf{B})$ cells with on average 3 registers, which is not much different from QS. The numbers of LED's per cell is discussed below.

The simplest approach to simultaneous algebraic and rational sieving would be to let the rational B register and the algebraic B registers in a particular cell share the same LED. In the terminology of 5.2 and 5.3 this would mean that the

dual condition "$S_1(x) > T_1(-\log(q))$ and $S_2(x) > T_2$" is replaced by the single condition "$S_1(x) + S_2(x) > T_1(-\log(q)) + T_2$". Extensive software experiments using this simplification were not encouraging, as it leads to too many false reports (with the original T_i's) or too many missed pairs (with adapted T_i's). Nevertheless, for TWINKLE it may be worth trying this approach. It would lead to a single LED per cell.

A more promising approach would be to have the algebraic flashes on the odd beat and the rational flashes on the even beat. This can easily be realized by storing $2p$ instead of p in the A registers and by changing the values initially stored in the B registers in the obvious way. If the photodetector detects a pair of consecutive odd and even high intensities a *report* occurs, i.e., a good pair may have been found. This approach still requires a single LED per cell, but it has the disadvantage that it takes two clock cycles to process a single sieve location.

Another approach would be to use LED's of different colours for algebraic and rational flashes. The algebraic LED flashes if either of the algebraic B registers is in a special state, and the rational LED flashes if the rational B register is in a special state. A report occurs if two photodetectors for the two different frequencies simultaneously detect a high intensity. In this approach all cells have a rational LED and 19/30 of the cells have an algebraic LED as well, for a total of $49 * \pi(\mathbf{B})/30$ LED's, which is almost 5/3 LED's per cell on average. The advantage of this approach, which we assume in the sequel, is that processing a single sieve location takes a single clock cycle, as in the original TWINKLE design. Note that it requires yet another different wavelength to avoid confusion with other optical signals.

6.5 Geometric Considerations

The geometry of the TWINKLE design described in [7] was based on the operational requirement that the time delay along the optical paths (from the clocking LED to all the cells on the flat wafer, and from these cells back to the summing photodetector) should be as uniform as possible. The recommended design placed the wafer at one face of a cylindrical tube, the photodetector at the center of the opposite face, and several synchronized clocking LED's around the perimeter of this face. This physical design reduced but did not eliminate the time difference between various optical paths in the tube. As a result, the tube had to be quite long, and thus the LED's on the wafer had to be made stronger, bigger, and more power consuming.

A greatly improved design (which was independently discovered by several researchers in private communication) places both the clocking LED and the photodetector at the center of one face of the cylinder, and at the focal point of a convex lens placed inside the cylinder between its two faces. Since all the relevant light rays (in both directions) between the lens and the wafer are parallel along the cylinder, all the wave fronts (which are perpendicular to the light rays) are flat and parallel to the wafer, and thus the time delay from the clocking LED to any point in the wafer and from there back to the photodetector is exactly the same. In addition, all the light gathered by the lens is concentrated on the

small face of the photodetector, and thus the LED's on the wafer can be made smaller and weaker.

7 Analysis of TWINKLE-Assisted NFS Factorizations

7.1 Line Sieving for 512-bit Numbers

To simplify the analysis, we assume for the moment that the factor base size is irrelevant. Under this assumption, the device as described in [7] and with the modifications from Section 6, can straightforwardly be used to perform line sieving for an NFS 512-bit factorization. The primes p for the (p, r) in P_2 are loaded once in the A registers, with the exact distribution over the different types of cells determined by the number of roots of f_1 mod p, as implied by the description in 6.4. For the first line ($b = 1$) the B register corresponding to a pair $(p, r) \in P_1 \cup P_2$ is initialized as $r + \mathbf{A} - p * [(r + \mathbf{A})/p]$, where $\mathbf{A} = 9 * 10^9$. The initial B-value for the next line follows by adding r to the initial value for the current line and taking the result modulo p. Thus, computation of the two million initial B register values for the next line can be done on an ordinary PC in less time than it takes TWINKLE to sieve the current line (see below). As shown in 5.5, a total of $\mathbf{A}/s = 9 * 10^5$ lines of length $2 * \mathbf{A} = 1.8 * 10^{10}$ each should suffice. As in Section 3 we consider two possible scenarios:

1. A modified TWINKLE device running at the maximum possible speed of 10 GHz. Each sieving interval of length $1.8 * 10^{10}$ can be scanned in 1.8 seconds. Reloading the B registers can be done in 0.02 seconds (cf. [7]) when done sequentially for all registers, or in 0.002 seconds when done in 10-fold parallelism, and can thus be neglected. All $9 * 10^5$ lines can be processed in approximately $1.8 * 9 * 10^5$ seconds, which is less than 3 weeks. A speed-up by 25% can be obtained by excluding 'even, even' locations from the sieve (cf. 5.5). This improvement is not reflected in our TWINKLE running time estimates but is included in the running times from [1]. The 3 week estimate includes the time for reloading and resieving, but does not include the time to do the actual divisions and cofactor factorizations. The latter can be done in $1.8 * 9 * 10^5$ seconds by about 43 loosely coupled PC's, as estimated in 5.4, and one additional PC is needed to compute the root updates. A total of 44 PC's would be able to do the sieving step in about 21 weeks (using special q sieving, cf. [1]). The improvement ratio is about a factor 8.
2. A modified TWINKLE device running at the minimum recommended speed of 1 GHz (cf. 6.1). Each sieving interval of length $1.8 * 10^{10}$ can be scanned in 18 seconds and all $9 * 10^5$ lines can be processed in approximately $18 * 9 * 10^5$ seconds, which is less than 27 weeks. It follows from 5.4 that 5 auxiliary PC's suffice for all auxiliary computations (divisions, cofactor decompositions, and root updates). A total of 5 PC's would be able to do the sieving step in about 186 weeks, and the improvement ratio we obtain is about a factor 7.

Thus, as in Section 3, we get about the same improvement ratio regardless of the clock rate of the TWINKLE device. This is due to the fact that the

computational bottleneck is in the supporting PC's. Note that the improvement ratio is close to the maximum attainable ratio implied by the last paragraph of 5.4.

The factor base sizes specified in 4.2 imply that the TWINKLE device, using the cell design as in 6.4, would contain about 10 wafers of more or less the same size as the wafers described in [7]. For that reason we now concentrate on how special q sieving may be used for TWINKLE-assisted factorizations of 512-bit and 768-bit numbers.

7.2 Special Sieving for 512-bit Numbers

Naïve implementation of special q sieving on a modified TWINKLE device is not promising. Despite the fact that a total of only 10^{15} sieve locations (cf. 5.5) have to be processed (which can, at 10 GHz, be done in less than 28 hours, including the resieving), the B registers have to be reloaded every $2 * 2^{12} =$ 8,192 sieve locations (cf. 5.5). Even with 10-fold parallelized reloading this adds $0.002 * (10^{15}/8{,}192) = 2.4 * 10^8$ seconds, i.e., almost 8 years, to the sieving time, without even considering how a PC is supposed to prepare the required data in 0.8 microseconds (the sieving time per line). As noted in 5.3, this problem is due to the fact that in special q sieving one cannot touch all factor base elements for all lines without violating the NFS running time.

A much better solution is obtained by radically changing the approach, and to make use of the fact that the majority of the factor base elements does not hit a particular line. Of the 2 million (p, r) pairs with $p > 2 * 2^{12}$ on average only about 10^4 hit a particular line, and if it hits, it hits just once. It follows that on average $2 * \pi(8{,}192) + 10^4$ pairs must be considered per line, and that roughly $2 * 10^4$ cells suffice if the same cell is associated with different p's (of approximately the same size) for different lines. Of these cells $2 * \pi(8{,}192)$ are as usual with fixed A registers, variable B registers, and proper arithmetic progressions. The other cells, however, need B registers only, assuming that their (variable) primes can be derived from their location if a report occurs. This follows from the fact that there is just a single hit, which implies that there is no true arithmetic progression to sieve with, and that the step size p is not needed. A clear advantage is that it simplifies the design of the TWINKLE device considerably, because only a single wafer with about $2 * 10^4$ cells would suffice. And most cells are even simpler than usual since they contain just one B register, two photodetectors, and a single rational or algebraic LED (split evenly among the cells). Note that the TWINKLE device would not actually be sieving for the primes $>$ 8,192 but act as an accumulator of logarithms of primes corresponding to identical B values.

We analyze the resulting speed for this modified and simplified TWINKLE device running at the maximum possible speed of 10 GHz. The number of sieve locations per special q is $8,192 * 5,000$ which can be scanned in 4 milliseconds. Per line about $2 * \pi(8{,}192) + 10^4$ values have to be reloaded. This can be done in 0.12 milliseconds. Thus, the auxiliary PC's have $5{,}000 * 0.00012 + 0.004 = 0.6$ seconds to prepare the list of register values ordered according to the lines where they

should be used. Obviously, the PC's should also not touch each line per (p, r) pair, so they will have to use some type of lattice technique to process each (p, r) pair. The lattice siever from [2] that was used in [1] takes about 8.8 seconds to provide the required list. Thus, one may expect that about $8.8/0.6 \approx 15$ PC's are required to prepare the line data for a TWINKLE device. It follows that a single modified and simplified TWINKLE device supported by about 15 PC's can do the special q NFS sieving for a 512-bit number in $(\pi(5*10^8)-\pi(2^{24}))*0.6$ seconds, i.e., about half a year. To keep up with the actual divisions and cofactor factorizations at 0.1 seconds per resulting relation (cf. 5.4) for 100 million relations (cf. 4.3), a single PC suffices. A total of 16 PC's would be able to do the sieving in slightly more than a year (cf. [1]), and the total improvement ratio is about 2.3. But note that the auxiliary PC's require only a modest amount of memory, whereas the PC's running the special q siever from [2] need to be able to allocate 64 megabytes of RAM to run efficiently. The same analysis holds when the TWINKLE device runs at the minimum recommended speed of 1 GHz.

The single wafer required for this modified and simplified TWINKLE device is much smaller than the one proposed in [7]. From our analysis it looks as if loading the new line data is the big bottleneck for special q sieving on the TWINKLE device. If that can be done x times faster, the TWINKLE device will run about x times faster. But the comparison to PC's would not be affected, because also x times more PC's would be needed to prepare the line data. So, from that point of view the data loading time is not a bottleneck, and we conclude that the PC support required for the preparation of the list of line data has a similar (and even stronger) effect on TWINKLE-assisted special q sieving as the post-processing PC-support has for TWINKLE-assisted line sieving.

7.3 Special q Sieving for 768-bit Numbers

Based on the asymptotic running time of the NFS, it may be expected that 768-bit numbers are at most 5,000 times harder to factor than 512-bit numbers. The size of the total sieving region grows proportional to the running time, and the factor base sizes and size of sieving region per special q grow proportional to the squareroot of the running time. Based on the figures from [1] we expect that 90,000 PC's with huge RAM's of about 5 gigabytes per PC can do the special q sieving in about a year. Based on extrapolation of the results from [1] we expect that one terabyte of disk space (about 50 standard PC hard disks costing a total of about $10,000) would suffice to store the data resulting from the sieving step.

Using well known structured Gaussian elimination methods that require only sequential disk-access to the data, a matrix of less than half a billion rows and columns and on average less than 100 entries per row can be built, requiring less than 200 gigabytes of disk space. Extrapolation of existing PC implementations of the block Lanczos algorithm suggests that this still relatively sparse matrix can be processed in less than 4,000 years on a single PC, using a blocking factor of 32. Preliminary results of block Lanczos parallelization seem to indicate that k-fold parallelization leads to a $(k/3)$-fold speed-up, where so far no values of $k > 16$ have been used (cf. [4]). Assuming that this parallelization scales up

to larger k, application of these preliminary results with $k = 5 * 4 * 4,000 =$ 80,000 and a conservative (80,000/5)-fold speed-up leads to the estimate that 80,000 PC's can do the matrix step in 3 months, when they are connected to a sufficiently fast network. Each of the PC clients would need only a few megabytes of RAM to store only a small fraction of the matrix. Unlike other figures in this paper, this estimate has not been confirmed by an actual implementation, and we stress that it is based on the assumption that the parallelization from [4] scales reasonably well. Note that future 64-bit PC's can use a blocking factor of 64, thereby halving the number of Lanczos iterations and substantially reducing the time required. Another way to parallelize block Lanczos that may be worth considering is to replace each PC client by clusters of, say, t PC clients, thereby further reducing the number of Lanczos iterations by a factor t.

We now consider how the simplified design from 7.2 scales up to 768-bit numbers. The total sieving time increases by a factor of about 5,000 to approximately 2,500 years. The factor base sizes increase by a factor 70, but so does the size of the sieving region per special q, so the same number of supporting PC's will be able to prepare the required lists of line data, per TWINKLE device. The wafer size would increase by a factor less than 9, and thus become comparable to the size proposed in [7]. We can thus conclude that about 5,000 modified and simplified TWINKLE devices supported by about 80,000 PC's can do the sieving step for a 768-bit number in about half a year. With the above estimate for the matrix step we arrive at the estimate given in the abstract.

PC's with 5 gigabyte RAM's which are needed to run the special q siever in standard NFS factorizations are highly specialized: Only a negligible number of such machines exist, and they have very few other applications. On the other hand, the auxiliary PC's in TWINKLE-assisted factorizations do not need exceptionally large memories, and thus it is possible to timeshare standard PC's which are used for other purposes in the organization (or over the Internet) during daytime. Large memories are also not needed for parallelized block Lanczos implementations. Since the 80,000 PC's are likely to be more expensive than the 5,000 TWINKLE devices, their free availability can dramatically reduce the cost of the hardware, and make a TWINKLE-assisted attack on a 768-bit RSA modulus much more feasible than a pure PC-based attack that uses dedicated PC's with huge memories.

8 TWINKLE without Optoelectronics

After the publication of the original TWINKLE paper, several alternative implementations were proposed by various researchers. The main theme of the modified designs was to replace the optoelectronic adder by an electronic adder of one of the following types:

1. An analog adder, in which each cell adds some current to a common line. An event is registered whenever the total current is high enough.
2. A digital adder, in which a tree of local 2-way adders adds the binary numbers which represent the contributions of the various cells.

3. A one dimensional systolic array, in which each cell increments one in p numbers passing through it for some p. The sequence of numbers "falling off" the end of the array is scanned for high entries.

The analog adder is likely to be too slow to react to rapidly changing signals due to the high capacitance of the tree of wires. The digital adder tree is faster, but each adder is likely to use a larger area and more power than a single LED which is dark most of the time. In addition, large adder trees are not fault tolerant, since a dead adder can eliminate the contributions of all the cells in its subtree. Similarly, a systolic array requires complex bypass mechanisms to overcome the dead or unreliable cells along it, since each number should pass through all the cells.

A purely electronic design may look more attractive than an optoelectronic design, since it is slightly easier to design and somewhat cheaper to manufacture. However, this is not likely to be a major consideration in large scale factoring efforts by large organizations, and in most respects it makes the design less efficient: Gallium Arsenide technology is faster than silicon technology, LED's are smaller than adders, independent cells are more fault tolerant than interconnected cells, and ultraprecise timing is easier to achieve with optics than with electronics.

Conclusion

From our analysis we conclude that both the original TWINKLE device as proposed in [7] and the variant that runs at one tenth of the speed can be expected to achieve a substantial speed-up over a PC implementation for the QS-factorization of 384-bit numbers. We described a modified version of the TWINKLE device that is better suited for the implementation of the NFS factoring algorithm than the original design. We found that 768-bit RSA moduli are more vulnerable to NFS attacks by our improved TWINKLE design than by current PC implementations.

References

1. S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgomery, B. Murphy, H.J.J. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, P. Zimmermann, Factorization of a 512-bit RSA modulus, these proceedings.
2. R. Golliver, A.K. Lenstra, K.S. McCurley, Lattice sieving and trial division, Proceedings of ANTS-I, Lecture Notes in Computer Science 877, Springer, pp 18-27.
3. A.K. Lenstra, H.W. Lenstra, Jr., The development of the number field sieve, Lecture Notes in Mathematics 1554, Springer, 1993.
4. P.L. Montgomery, Parallel block Lanczos, presentation at the RSA 2000 conference, Jan 17, 2000.
5. P.L. Montgomery, B. Murphy, Improved polynomial selection for the number field sieve. Extended abstract for the Conference on the Mathematics of Public-Key Cryptography, June 13-17, 1999, The Fields Institute, Toronto, Ontario, Canada.

6. C. Pomerance, The quadratic sieve factoring algorithm, Proceedings of Eurocrypt'84, Lecture Notes in Computer Science, Springer, pp 169-182.
7. A. Shamir, Factoring large numbers with the TWINKLE device, Proceedings of the CHES conference, Lecture Notes in Computer Science 1717, Springer, August 1999.

Noisy Polynomial Interpolation and Noisy Chinese Remaindering

Daniel Bleichenbacher[1] and Phong Q. Nguyen[2]

[1] Bell Laboratories, Rm. 2A-366, 700 Mountain Av
Murray Hill, NJ 07974-0636, USA
bleichen@bell-labs.com and http://www.bell-labs.com/user/bleichen/
[2] École Normale Supérieure, Département d'Informatique,
45 rue d'Ulm, 75005 Paris, France
pnguyen@ens.fr and http://www.di.ens.fr/~pnguyen/

Abstract. The noisy polynomial interpolation problem is a new intractability assumption introduced last year in oblivious polynomial evaluation. It also appeared independently in password identification schemes, due to its connection with secret sharing schemes based on Lagrange's polynomial interpolation. This paper presents new algorithms to solve the noisy polynomial interpolation problem. In particular, we prove a reduction from noisy polynomial interpolation to the lattice shortest vector problem, when the parameters satisfy a certain condition that we make explicit. Standard lattice reduction techniques appear to solve many instances of the problem. It follows that noisy polynomial interpolation is much easier than expected. We therefore suggest simple modifications to several cryptographic schemes recently proposed, in order to change the intractability assumption. We also discuss analogous methods for the related noisy Chinese remaindering problem arising from the well-known analogy between polynomials and integers.

1 Introduction

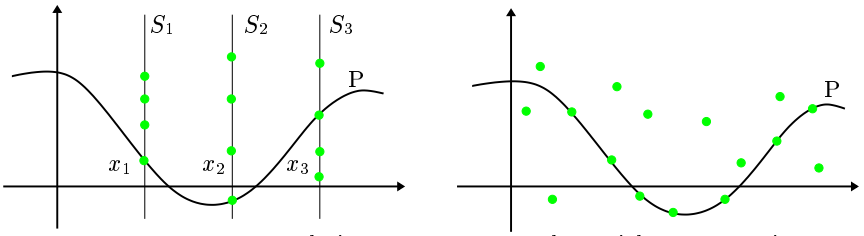
At STOC '99, Naor and Pinkas [26] introduced a new and useful primitive: oblivious evaluation of polynomials, where a polynomial P is known to Bob and he would like to let Alice compute the value $P(x)$ for an input x known to her in such a way that Bob does not learn x and Alice does not gain any additional information about P. The scheme they proposed is quite attractive, as it is much more efficient than traditional oblivious evaluation protocols, which leads to several applications. For instance, Gilboa [14] applied the scheme to two party RSA key generation. Naor and Pinkas mention other interesting applications in their paper [26], such as a method enabling two agencies each having a list of names, to find the common names on the lists without revealing other information.

Perhaps the only problem with the Naor-Pinkas scheme was a security issue, since the scheme used a new intractability assumption. The underlying computational problem, the so-called noisy polynomial interpolation problem, can be stated as follows:

Problem 1 (Noisy polynomial interpolation). Let P be a k-degree polynomial over a finite field $\mathbb{F}$. Given $n > k+1$ sets $S_1, \ldots, S_n$ and n distinct elements $x_1, \ldots, x_n \in \mathbb{F}$ such that each $S_i = \{y_{i,j}\}_{1 \le j \le m}$ contains $m-1$ random elements and $P(x_i)$, recover the polynomial P, provided that the solution is unique.

A simple counting argument suggests that $m^n \ll |\mathbb{F}|^{n-(k+1)}$ should be satisfied to ensure the unicity of the solution. Several generalizations are possible: for instance, one can assume that the sets S_i's have different sizes instead of m. A related problem is the following:

Problem 2 (Polynomial reconstruction). Given as input integers k, t and n points $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{F}^2$, output all univariate polynomials P of degree at most k such that $y_i = P(x_i)$ for at least t values of i.



Noisy polynomial interpolation — Polynomial reconstruction

The polynomial reconstruction problem is well-known because the generalized Reed-Solomon list decoding problem reduces to it. The best algorithm known to solve this problem is the recent algorithm of Guruswami and Sudan [17] (GS), which was inspired by previous work of Ar *et al.* [3] on a related problem. Its running time is polynomial in n, and the algorithm succeeds provided $t > \sqrt{kn}$, for any field $\mathbb{F}$ of cardinality at most 2^n. Naor and Pinkas remarked the existence of a simple reduction from noisy polynomial interpolation to polynomial reconstruction, which led them to conjecture that the noisy polynomial interpolation problem was as hard as the polynomial reconstruction problem.

This paper provides evidence that the conjecture is likely to be false. More precisely, we present new methods to solve noisy polynomial interpolation which (apparently) do not apply to polynomial reconstruction. In particular, we prove that the noisy polynomial interpolation problem can be transformed into a lattice shortest vector problem with high probability, provided that the parameters satisfy a certain condition that we make explicit. This result is qualitatively similar to the well-known lattice-based methods [20, 9] to solve the subset sum problem: the subset sum problem can be transformed into a lattice shortest vector problem with high probability, provided that a so-called low-density condition is satisfied. As with subset sums, experimental evidence suggest that most practical instances of the noisy polynomial interpolation problem with small m can be solved. It follows that noisy polynomial interpolation is much easier than expected (despite known hardness results [2, 24] on the lattice shortest vector problem), and thus, should be used cautiously as an intractability assumption.

Interestingly, the noisy polynomial interpolation and the polynomial reconstruction problems also appeared in password authentication schemes [25, 13]. Both schemes use Shamir's secret sharing scheme based on Lagrange's polynomial interpolation, where the shares are encrypted with low entropy secrets. Shamir's scheme achieves perfect security, but here, additional information is available to the attacker. A closer inspection shows that [13] is based on the noisy polynomial interpolation problem, and is therefore insecure for many choices of the parameters. For instance, the authors propose to use $n = 22, k = 14$ and $m \approx 256$ to protect a 112-bit key. But this configuration can be broken using a meet-in-the-middle attack (see Section 2.3) using $n' = 16$ in time 2^{64}. The solution described in [25] is much better as it is based on the hardness of the discrete log problem and a variant of the polynomial reconstruction problem.

We also discuss analogous methods for a related problem, the so-called noisy Chinese remaindering problem arising from the well-known analogy between polynomials and integers. Curiously, problems such as point counting on elliptic curves over finite fields and integer factorization of the form p^2q, can be viewed as generalized noisy Chinese remaindering problems. We explain why the lattice-based approach does not appear to be as useful in such settings.

The paper is organized as follows. In Section 2, we review simple methods for noisy polynomial interpolation. Section 3 is devoted to lattice-based methods. Cryptographic implications of these results are discussed in Section 4. In Section 5, we study analogous methods for the noisy Chinese remaindering problem. Due to lack of space, some details and proofs are omitted, but those can be found in the full version available on our webpages.

2 Simple methods for noisy polynomial interpolation

2.1 An error-correction method

When the noisy polynomial interpolation problem appeared in [26], the only known algorithm to solve it (apart from exhaustive search) was based on a simple reduction from noisy polynomial interpolation to polynomial reconstruction. More precisely, Naor and Pinkas noticed that by randomly choosing one element $y_{i,j}$ in S_i, one obtains an instance of the polynomial reconstruction problem with the n (randomly chosen) points $(x_i, y_{i,j})$. The solution P is of degree k, and we have $P(x_i) = y_{i,j}$ for approximately n/m values of i. Therefore the solution is expected to be outputted by the GS algorithm, provided that $\frac{n}{m} > \sqrt{kn}$, that is: $m < \sqrt{\frac{n}{k}}$. In fact, one can obtain a better reduction by taking all the points, which was apparently unnoticed. Indeed, if one picks all the nm points $(x_i, y_{i,j})$, then the solution P of degree k satisfies $P(x_i) = y_{i,j}$ for at least n values of (i, j). Hence, the GS algorithm will output P if $n > \sqrt{knm}$, that is: $m < n/k$. It is worth noting that this condition does not depend on the size of the finite field. The previous reductions do not use the specificity of the noisy polynomial interpolation instances. It is not known whether one can improve GS algorithm

when applied to those particular instances, although [6] describes a simple algorithm achieving the same bound $m < n/k$. We now present methods to solve the problem when the condition $m < n/k$ is not satisfied.

2.2 A Gröbner basis method

A natural way to solve the noisy polynomial interpolation problem is reducing the problem to solving a system of polynomial multivariate equations. Write the unknown polynomial P as $P(X) = \sum_{i=0}^{k} a_i X^i$. For all i, there exists j such that $P(x_i) = y_{i,j}$, therefore:

$$\prod_{j=1}^{m} (P(x_i) - y_{i,j}) = 0.$$

One thus obtains n polynomial equations in the $k+1$ unknowns $a_0, \ldots, a_k$, in the field $\mathbb{F}$.

Gröbner basis is the usual way to solve such systems. However, the complexity of such techniques is super-exponential in k: in practice, it is likely that the method would be impractical if k is not very small (for instance, larger than 20). Theoretically, one could also apply the relinearization technique recently introduced by Kipnis and Shamir [19], at least in the case $m = 2$ (that is, a system of quadratic equations). At the moment, the behaviour of this new method is not completely understood, however latest results [10] suggest that the method is impractical for sufficiently large k, such as $k \geq 50$.

2.3 A meet-in-the-middle method

A meet-in-the-middle approach can be used to solve the noisy polynomial interpolation problem. Let $n' \leq n$ be the smallest integer for which we expect the solution to be unique. Define the Lagrange interpolation polynomials in $\mathbb{F}[X]$:

$$L_i(X) = \prod_{\substack{1 \leq j \leq n' \\ j \neq i}} \frac{X - x_j}{x_i - x_j}.$$

The degree of L_i is $n' - 1$. We are looking for coefficients c_i, such that

$$\deg \left(\sum_{i=1}^{n'} y_{i,c_i} L_i(X) \right) \leq k.$$

For all $\mathbf{c} = (c_1, \ldots, c_{\lfloor n'/2 \rfloor}) \in \{1, \ldots, m\}^{\lfloor n'/2 \rfloor}$ and $\tilde{\mathbf{c}} = (c_{\lfloor n'/2 \rfloor + 1}, \ldots, c'_n) \in \{1, \ldots, m\}^{\lceil n'/2 \rceil}$ we compute the polynomials $U_{\mathbf{c}}(X) = \sum_{i=1}^{\lfloor n'/2 \rfloor} y_{i,c_i} L_i(X)$ and $V_{\tilde{\mathbf{c}}}(X) = -\sum_{i=\lfloor n'/2 \rfloor + 1}^{n'} y_{i,c_i} L_i(X)$. We compare the two lists: If some $U_{\mathbf{c}}(X)$ and $V_{\tilde{\mathbf{c}}}(X)$ have identical coefficients for the terms $X^{k+1}, \ldots, X^{n'}$ then $U_{\mathbf{c}}(X) - V_{\tilde{\mathbf{c}}}(X)$ has degree at most k, and therefore, solves the problem.

The method requires the computation of $O(m^{\lceil n'/2 \rceil})$ polynomials $U_{\mathbf{c}}(X)$ and $V_{\tilde{\mathbf{c}}}(X)$. Since the values for $y_{i,j}L_i(X)$ can be precomputed and partial sums can be reused, the time complexity of this attack is $O(c(n'-k)m^{\lceil n'/2 \rceil})$, where c is the time for an addition in $\mathbb{F}$. The memory requirement of this algorithm is $O((\log q)m^{\lfloor n'/2 \rfloor})$, but an improved algorithm needing $O((\log q)m^{\lceil n'/4 \rceil})$ exists.

It is worth noting that the meet-in-the-middle method does not apply to the polynomial reconstruction problem. This is because the Lagrange polynomials $L_i(X)$ in this problem depend on the selection of the values $y_{i,j}$ used for the interpolation. Different $y_{i,j}$'s correspond to different x_i's and therefore different Lagrange polynomials. The meet-in-the-middle method takes advantage of the fact that the x_i's are known in advance.

Note that the meet-in-the-middle method can still be used if we have to compute $g^{f(x_0)}$ for some public x_0 and g, when given the $g^{y_{i,j}}$'s rather than the $y_{i,j}$'s. This is because polynomial interpolation is a linear function of the inputs $y_{i,j}$.

3 Lattice-based methods for noisy polynomial interpolation

We now describe lattice-based methods to solve noisy polynomial interpolation. To simplify the presentation, we assume in the whole section that the finite field $\mathbb{F}$ is a prime field $\mathbb{Z}_q$ (q being a prime number). The results extend to the general case by viewing $\mathbb{F}$ as a finite dimensional vector space over its prime field.

In this paper, we will call *lattice* any integer lattice, that is, any subgroup of $(\mathbb{Z}^n, +)$ for some n. Background on lattice theory can be found in several textbooks, such as [16, 35]. For lattice-based cryptanalysis, we refer to [18].

Our lattice-based methods build in polynomial time a lattice from a given instance of noisy polynomial interpolation. In this lattice, there is a particular lattice point, the so-called *target vector*, which is both unusually short and closely related to the solution of our problem. We will first give heuristic arguments suggesting that the target vector is the lattice shortest vector. Then we will modify our lattice to prove that the target vector is with high probability the shortest vector of the modified lattice, when the parameters satisfy a certain condition that we make explicit. The proofs are somewhat technical, but the underlying idea is similar to the one used to show that the low-density subset sum problem can be reduced with high probability to a lattice shortest vector problem [20, 9]. More precisely, we will estimate the probability that a fixed vector belongs to the lattice built from a randomly chosen instance of the problem. By enumerating all possible short vectors, we can then upper bound the probability that there exists a nonzero lattice point shorter than the target vector for a randomly chosen instance. From a practical point of view, one hopes to solve the problem by using standard lattice reductions algorithms [21, 30–32] as lattice shortest vector oracles.

3.1 Linearization of noisy polynomial interpolation

Let $L_i(X)$ be the Lagrange interpolation polynomial defined as

$$L_i(X) = \prod_{j \neq i} \frac{X - x_j}{x_i - x_j}.$$

The solution P satisfies: $P(X) = \sum_{i=1}^{n} P(x_i) L_i(X)$. We linearize the problem: letting $\delta_{i,j}$ equal to 1 if $P(x_i) = y_{i,j}$, and 0 otherwise, one obtains $P(x_i) = \sum_{j=1}^{m} \delta_{i,j} y_{i,j}$, hence:

$$P(X) = \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{i,j} y_{i,j} L_i(X).$$

Since $P(X)$ has degree k, while L_i has degree $n-1$, we obtain $n-1-k$ linear equations in the nm unknowns $\delta_{i,j}$. As a linear system in the field $\mathbb{F}$, it is underdefined. However, one can also view the problem as a lattice problem for which lattice reduction might apply.[1]

The set L of integer row vectors $(d_{1,1}, d_{1,2}, \ldots, d_{n,m}) \in \mathbb{Z}^{nm}$ such that the polynomial $\sum_{i=1}^{n} \sum_{j=1}^{m} d_{i,j} y_{i,j} L_i(X)$ has degree at most k is clearly a lattice in $\mathbb{Z}^{nm}$. The vector $(\delta_{1,1}, \delta_{1,2}, \ldots, \delta_{n,m})$ belongs to L, we call it the *target vector*. Its Euclidean norm is $\sqrt{n}$. To see how short this vector is compared to other lattice vectors, we need to analyze the lattice L. We wish to obtain results of the flavour of lattice-based algorithms to solve low-density subset sums [20, 9]: with high probability over a certain distribution of the inputs, and under specific conditions on the parameters, the target vector is the lattice shortest vector.

3.2 Volume of the lattice

The previous lattice is related to the lattices used by Ajtai [1] in his celebrated worst-case/average-case equivalence for certain lattice problems. More precisely, let A be a $n \times e$ matrix in $\mathbb{Z}_q$ where q is any integer. Let $L(A)$ be the set of n-dimensional integer row vectors $\mathbf{x}$ such that $\mathbf{x}A \equiv 0 \pmod{q}$. We call $L(A)$ the Ajtai lattice associated to A. It is easy to see that $L(A)$ is a n-dimensional lattice in $\mathbb{Z}^n$, from which one derives:

Lemma 1. *Let $A \in \mathcal{M}_{n,e}(\mathbb{Z}_q)$. Then the volume of $L(A)$ divides q^e. It is exactly q^e if and only if $\{\mathbf{x}A \ : \mathbf{x} \in \mathbb{Z}_q^n\}$ is entirely $\mathbb{Z}_q^e$.*

Proof: By definition, $L(A)$ is the kernel of the group homomorphism ϕ that maps any $\mathbf{x} \in \mathbb{Z}^n$ to $(\mathbf{x}A \bmod q) \in \mathbb{Z}_q^e$. Therefore the group quotient $\mathbb{Z}^n / L(A)$ is isomorphic to the image of ϕ. But since $L(A)$ is a full-dimensional lattice in $\mathbb{Z}^n$, its volume is simply the index $[\mathbb{Z}^n : L(A)]$ of $L(A)$ in $\mathbb{Z}^n$, from which both statements follow. □

[1] If we used the field $\mathrm{GF}(q^a)$ rather than $\mathbb{Z}_q$ we would have $a(n-1-k)$ equations in nm unknowns over $\mathbb{Z}_q$ and the linear system might be solvable directly.

Letting $L_i(x) = \sum_{w=0}^{n-1} \ell_{i,w} x^w$, the lattice L of Section 3.1 is equal to $L(A)$, where $\mathbb{F} = \mathbb{Z}_q$ and A is the following matrix of dimension $nm \times n-1-k$.

$$A = \begin{pmatrix} y_{1,1}\ell_{1,k+1} & \cdots & y_{1,1}\ell_{1,n-1} \\ \vdots & & \vdots \\ y_{i,j}\ell_{i,k+1} & \cdots & y_{i,j}\ell_{i,n-1} \\ \vdots & & \vdots \\ y_{n,m}\ell_{n,k+1} & \cdots & y_{n,m}\ell_{n,n-1} \end{pmatrix}$$

Lemma 2. *Assume that for all* $1 \leq i \leq n$ *there exists* $1 \leq w_i \leq m$, *such that* $y_{i,w_i} \neq 0$. *Then* $\mathrm{rank}(A) = n-1-k$.

Proof: Remember, that for all $c_1, \ldots, c_n \in \mathbb{F}^n$ and $f(x) = \sum_{i=1}^n c_i L_i(x)$ we have $f(x_i) = c_i$. Hence, $\sum_{i=1}^n c_i L_i(x) = 0$ implies $c_1 = \cdots = c_n = 0$. This shows that the $n \times n$ matrix $(\ell_{i,j})_{1 \leq i \leq n; 0 \leq j \leq n-1}$ is nonsingular. In particular, the last $n-1-k$ columns are linearly independent and thus the matrix $(\ell_{i,j})_{1 \leq i \leq n; k+1 \leq j \leq n-1}$ has rank $n-1-k$. We assumed that $y_{i,w_i} \neq 0$ and therefore the matrix $A_0 = (y_{i,w_i}\ell_{i,j})_{1 \leq i \leq n; k+1 \leq j \leq n-1}$ has rank $n-1-k$ too. Since A_0 is a submatrix of A it follows that A has rank $n-1-k$ too. □

A consequence of this lemma is that the set $\{0, \ldots, q-1\}^{nm}$ contains exactly $q^{nm-n+1+k}$ lattice points and hence the volume of $L(A)$ is q^{n-1-k}. Therefore, if γ_d denotes Hermite's constant of order d, we have:

$$\lambda_1(L) \leq \sqrt{\gamma_{nm}}\, q^{\frac{n-1-k}{nm}},$$

where $\lambda_1(L)$ is the first minimum of L (the length of a shortest non-zero lattice point). The best asymptotic estimate known of Hermite's constant is the following (see [7]):

$$\frac{d}{2\pi e} + \frac{\log(\pi d)}{2\pi e} + o(1) \leq \gamma_d \leq \frac{1.744d}{2\pi e}(1 + o(1)).$$

It follows that one expects the target vector to be the shortest lattice vector if

$$\sqrt{n} \ll \sqrt{\frac{nm}{2\pi e}}\, q^{\frac{n-1-k}{nm}}.$$

This condition is very heuristic, as the lattice L cannot be considered as a "random" lattice.

3.3 Structure of the lattice

We now give a different heuristic argument to guess when the target vector is the shortest vector. The argument is inspired by lattice-based attacks against low-density subset sums (see [20, 9]). If we denote by $N(n, r)$ the number of integer points in the n-dimensional sphere of radius $\sqrt{r}$ centered at the origin, we have the following elementary result :

Lemma 3. *Let A be a $nm \times e$ matrix in $\mathbb{Z}_q$ (q prime) chosen at random with uniform distribution. Then:*

$$\Pr\left(\lambda_1(L(A)) < \sqrt{n}\right) \leq \frac{N(nm,n)}{q^e}.$$

Proof: Let $\mathbf{x} = (x_1, \ldots, x_{nm}) \in \mathbb{Z}_q^{nm}$ be a non-zero vector. The probability that $\mathbf{x}A \equiv 0 \pmod q$ for a uniformly chosen matrix $A = (a_{i,j})_{1 \leq i \leq nm, 1 \leq j \leq e}$ is q^{-e}. Indeed, there exists $i_0 \in \{1, \ldots, nm\}$ such that $x_{i_0} \neq 0$. Then, for any choice of $(a_{i,j})_{i \neq i_0, 1 \leq j \leq e}$, there exists a unique choice of $(a_{i_0,j})_{1 \leq j \leq e}$ such that $\mathbf{x}A \equiv 0 \pmod q$, which gives the expected probability. Since the number of possible $\mathbf{x}$ is less than $N(nm,n)$, the result follows. □

It follows that one expects the target vector to be the shortest lattice vector when $N(nm,n) \ll q^{n-1-k}$. Numerical values of $N(nm,m)$ can be computed by recursion. And sharp theoretical estimates of $N(nm,m)$ can be obtained using the power series $h(x) = 1 + 2\sum_{k=1}^{\infty} x^{k^2}$ (see [23, Lemma 1]). However, the condition is still heuristic, since in our case, the matrix A cannot be considered as uniformly distributed. In particular, it does not seem easy to compute the probability that a fixed vector belongs to the lattice $L(A)$ for a randomly chosen instance of noisy polynomial interpolation.

3.4 Reduction by lattice improvement

To achieve a reduction from noisy polynomial interpolation to the lattice shortest vector problem, we consider a certain sublattice. The improvement is based on a property of the target vector which has not been used so far: for all i_1 and i_2, $\sum_{j=1}^{m} \delta_{i_1,j} = \sum_{j=1}^{m} \delta_{i_2,j} = 1$. This leads us to define the lattice Λ as the set of lattice points $(d_{1,1}, d_{1,2}, \cdots, d_{n,m}) \in L$ such that for all i_1 and i_2:

$$\sum_{j=1}^{m} d_{i_1,j} = \sum_{j=1}^{m} d_{i_2,j}. \tag{1}$$

Since Λ is the intersection of the full-dimensional lattice L (in $\mathbb{Z}^{nm}$) with a $(nm-n+1)$-dimensional vector subspace, Λ is a $(nm-n+1)$-dimensional lattice in $\mathbb{Z}^{nm}$, which can be computed in polynomial time.

We will be able to compute the probability that a (fixed) short vector satisfying (1) belongs to Λ, which was apparently not possible for L. The probability is with respect to the natural distribution induced by the definition of noisy polynomial interpolation, which is the following:

- Let $x_1, \ldots, x_n$ be distinct elements of $\mathbb{F} = \mathbb{Z}_q$, and g be a function from $\{1, \ldots, n\}$ to $\{1, \ldots, m\}$.
- Choose uniformly at random a k-degree polynomial P in $\mathbb{F}[X]$.
- For all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\} \setminus g(i)$, choose uniformly at random an element $y_{i,j}$ in $\mathbb{F}$, and let $y_{i,g(i)} = P(x_i)$.

Recall that the noisy polynomial interpolation problem is to recover either g or P, given k, the $y_{i,j}$'s and the x_i's. The (secret) function g indicates which $y_{i,j}$ is equal to $P(x_i)$.

Let $\mathbf{d} = (d_{1,1}, \ldots, d_{n,m}) \in \mathbb{Z}^{nm}$ be a vector satisfying (1). We define $p(\mathbf{d})$ as the probability that $\mathbf{d}$ belongs to the lattice Λ, that is, the probability that $\deg(\sum_{i=1}^{n} \sum_{j=1}^{m} d_{i,j} y_{i,j} L_i(X)) \leq k$, with respect to the previous distribution. Let $t(\mathbf{d})$ be the number of indices i for which there exists at least one nonzero $d_{i,j}$ modulo q with $j \neq g(i)$:

$$t(\mathbf{d}) = \Big| \{1 \leq i \leq n : \exists j \in \{1, \ldots, m\} \setminus g(i) \text{ such that } d_{i,j} \not\equiv 0 \bmod q\} \Big|.$$

The following technical lemma gives a formula for $p(\mathbf{d})$. It shows that the heuristic assumptions made in Section 3.2 and Section 3.3 are correct for all vectors $\mathbf{d}$ where $t(\mathbf{d}) \geq n-k-1$, but $p(\mathbf{d})$ is larger than expected when $t(\mathbf{d}) < n-k-1$. As we will see later the effect of those vectors is often negligible. A proof can be found in the full version of the paper.

Lemma 4. *Let $\mathbf{d} \in \mathbb{Z}^{nm}$ satisfying (1). Then:*

$$p(\mathbf{d}) = q^{-\min(t(\mathbf{d}), n-k-1)}.$$

It follows that $p(\mathbf{d}) > \frac{1}{q}$ if and only if $t(\mathbf{d}) = 0$ (recall that $n > k+1$). But if $\mathbf{d}$ satisfies (1) and $t(\mathbf{d}) = 0$, then either $\mathbf{d}$ is a multiple (possibly zero) of the target vector, or at least one of $\mathbf{d}$'s entries is a nonzero multiple of q, implying $\|\mathbf{d}\| \geq q$. By enumerating all possible $\mathbf{d}$'s, we finally obtain a reduction:

Theorem 1. *Let $\sqrt{r} < q$. Let a noisy polynomial interpolation instance be chosen uniformly at random as described above and let Λ be the sublattice built from the instance. Then the expected number of nonzero vectors $E(r, n, m)$ contained in Λ not equal to the target vector or a multiple of it with norm $\leq \sqrt{r}$ is:*

$$E(r, n, m) = \sum_{\lambda=-\lfloor r/n \rfloor}^{\lfloor r/n \rfloor} \sum_{w=1}^{n} R(w, r, \lambda, n, m)\ q^{-\min(w, n-k-1)},$$

where $R(w, r, \lambda, n, m)$ denotes the number of vectors $\mathbf{d} = (d_{1,1}, \ldots, d_{n,m}) \in \mathbb{Z}^{mn}$ such that $t(\mathbf{d}) = w$, $\|\mathbf{d}\| \leq \sqrt{r}$ and $\sum_{j=1}^{m} d_{i,j} = \lambda$.

If $E(n, n, m) < 1$ then $E(n, n, m)$ is a nontrivial upper bound on the probability that Λ contains a nonzero vector shorter than the target vector. The proof of Theorem 1 and numerical methods to compute $E(r, n, m)$ are given in the full version of the paper. The results are more complicated than low-density subset sum attacks for the following reasons. In low-density subset sum attacks, one can compute fairly easily an upper bound of the probability that a fixed nonzero short vector (different from the target vector) belongs to a certain lattice built from the subset sum instance (see [20, 9]). And the bound obtained is independent of the vector. It then remains to estimate the number of possible short vectors, by bounding the number of integer points in high-dimensional spheres (using

techniques of [23]). Here, we have an exact formula for the probability instead of an upper bound, but the formula depends on the vector, for it involves $t(\mathbf{d})$. This leads to more complicated enumerations and asymptotic formulas. Hence, we cannot give a criterion as "simple" as the low-density criterion for subset sum, to indicate when the reduction is expected to hold. However, for some special cases we have some preliminary results:

Lemma 5. *Let $n \geq 2$, $m \geq 2$ and $n^2 < q$. Let $0 < x < 1$ and $h(x) = 1 + 2\sum_{k=1}^{\infty} x^{k^2}$. Then:*

$$\frac{N(n, \lfloor n/2 \rfloor) + 2^{n+1} - 3}{q^{n-1-k}} \leq E(n, n, 2) \leq \frac{N(n, \lfloor n/2 \rfloor) + 2^{n+1}}{q^{n-1-k}} + 2n^2/q + 4n/q$$

$$E(n, n, m) \leq \frac{N(nm, n)}{q^{n-1-k}} + 3x^{-n}\left(\left(1 + \frac{h(x)^m}{q}\right)^n - 1\right)$$

The proof of Lemma 5 can be found in the full version of the paper. Note that $h(x)$ can be approximated numerically. The result for the case $m = 2$ are much stronger than the result for a general m. From a practical point of view, we can alternatively compute the upper bound $E(r, n, m)$ numerically for any given choice of the parameters. And the bound seems to be sharp in practice.

The following table shows for some values of m, n, q the largest k, such that the expected number of vectors with norm shorter or equal to $\sqrt{n}$ is smaller than 1. We compare this to the largest $\tilde{k}$ for which we would expect the target vector to be the shortest vector in the original lattice without improvement.

A missing entry in the column k says that for this particular choice of m and n the problem is very likely not solvable with the lattice based method for any k. We have chosen m and n such that the meet-in-the middle method has a time complexity of 2^{80}. We have chosen $q > 2^{80}$, so that elements of $\mathbb{Z}_q$ can be used to represent 80 bit keys for symmetric ciphers.

m	n	$\log_2(q)$	k	$\tilde{k}$
2	160	80	155	152
3	115	80	110	108
4	105	80	100	98
16	44	80	40	39
256	20	80	–	–

3.5 Non-prime fields

When $\mathbb{F}$ is a field of the form $\mathrm{GF}(q^a)$ with $a > 1$, Lemma 4 still holds if one replaces q by q^a, with the same definition of $t(\mathbf{d})$ (that is, the number of indices i for which there exists at least one nonzero $d_{i,j}$ modulo q with $j \neq g(i)$), so that $p(\mathbf{d}) = q^{-a \min(t(\mathbf{d}), n-k-1)}$. Theorem 1 and Lemma 5 need to be modified accordingly. It follows that the lattice-based approach is useful only when the characteristic of $\mathbb{F}$ is sufficiently high ($q > \sqrt{n}$), so that any vector $\mathbf{d}$ satisfying $t(\mathbf{d}) = 0$ is strictly longer than the target vector.

3.6 Experiments

We implemented the improved lattice-based method on a 500 MHz 64-bit DEC Alpha using Victor Shoup's NTL library [34]. For a randomly chosen instance, we built the corresponding sublattice Λ. For lattice reduction, we successively applied three different types of reduction : plain LLL [21], Schnorr's BKZ reduction [30, 31] with block size 20, and when necessary, Schnorr-Hörner's pruned BKZ reduction [32] with block size 54 and pruning factor 14. We stopped the reduction as soon as the reduced basis contained the target vector.

To fix ideas on the efficiency of the lattice-based method, we chose $n = 160$ and $m = 2$, with a prime field of size 80 bits. The error-correction method succeeds only if $k \leq 80$. The meet-in-the-middle method requires at least $2^{k/2}$ operations. And the Gröbner basis approaches are very unlikely to be practical. Numerical values given by theorem 1 (see Section 3.4) suggest that the noisy polynomial interpolation problem can be reduced to a lattice shortest vector problem, as while as $k \leq 155$. The lattice dimension is then 160. Our implementation was able to solve noisy polynomial interpolation up to $k = 154$. For $k \leq 152$, only BKZ-20 reduction was necessary, and the total running time was less than 4 hours. For $153 \leq k \leq 154$, an additional Schnorr-Hörner pruned BKZ reduction was necessary: 1 hour for $k = 153$, and 8 hours for $k = 154$. We do not know if the theoretical value of $k = 155$ can be reached in practice: the corresponding lattice problem is hard because there are many lattice points almost as short as the target vector. The situation might be similar to lattice-based subset sum attacks: when the subset sum density is very close to the critical density, and the lattice dimension is large, the lattice problem is hard. It is worth noting that to ensure the unicity of the solution, one should have $k \leq 156$. This suggests that the lattice-based method is likely to solve most instances of practical interest for small m. We also made a few experiments with $m > 2$. A BKZ-20 reduction can solve in one day the problem with $n = 115$, $k = 101$, $m = 3$ and $n = 105$, $k = 80$, $m = 4$. For such parameters, the meet-in-the-middle method requires at least 2^{80} operations.

4 Cryptographic implications

We showed that when the parameters satisfy a certain relation, there exists a provable reduction from noisy polynomial interpolation to the lattice shortest vector problem. This results in an attack which is much more effective than previously known methods based on list decoding algorithms, due to the strength of current lattice reduction algorithms. We could not apply the same method to the polynomial reconstruction problem. This suggests (but does not prove) that the polynomial reconstruction problem is harder than the noisy polynomial interpolation problem, so that Conjecture 3.1 in [26] about the hardness equivalence[2] of the two problems does not hold.

[2] In fact, Conjecture 3.1 relates the hardness of polynomial reconstruction and an easier version of noisy polynomial interpolation.

It follows that cryptographic protocols should – if possible – be based on the polynomial reconstruction problem rather than the noisy polynomial interpolation problem. Such a change is possible for the oblivious polynomial evaluation of Naor and Pinkas [26]. There are two players Alice and Bob. Bob's secret input is a polynomial $P(x)$, which he hides in a bivariate polynomial $Q(x, y)$, such that $Q(0, y) = P(y)$. Alice has a secret value α and would like to learn $P(\alpha)$. Alice chooses a polynomial $S(x)$ with $S(0) = \alpha$. In a crucial step of the protocol Alice would like to learn $Q(x_i, S(x_i))$ without revealing $S(x)$. This is done by sending x_i and a list of random values $y_{i,j}$, except that one value $S(x_i)$. Bob computes $Q(x_i, y_{i,j})$ for all these values and A retrieves the answer she is interested in using a 1-out-of-m oblivious transfer. The privacy of Alice depends on the difficulty to find $S(x)$ given x_i and $y_{i,j}$, *i.e.* the noisy polynomial interpolation problem. However, the protocol can be changed by using the values $Q(x_{i,j}, y_{i,j})$ for distinct $x_{i,j}$'s rather than $Q(x_i, y_{i,j})$ [29].

Another way to prevent lattice-based attacks is to use a field where computing discrete logarithms is intractable, and to publish the powers $g^{y_{i,j}}$ rather than the values $y_{i,j}$. It is then still possible to perform a polynomial interpolation, that is to compute $g^{f(x_0)}$, given sufficiently many values $g^{f(x_i)}$. In fact, the meet-in-the middle method is the only algorithm known to us that is applicable in this case and it can only be used for the noisy polynomial interpolation problem but not for the polynomial reconstruction problem. A protocol using the polynomial interpolation problem combined with the discrete logarithm problem is described in [25].

5 Noisy Chinese remaindering

There is a well-known analogy between polynomials and integers: the polynomial degree corresponds to the integer size; Lagrange's interpolation corresponds to Chinese remainders; and polynomial evaluation corresponds to the modulo operation (in fact, a polynomial P evaluated at x_0 can also be viewed as the remainder of $P(x)$ modulo the linear polynomial $x - x_0$). We refer to [15] for some examples. The noisy polynomial interpolation and polynomial reconstruction problems then become the following ones:

Problem 3 (Noisy Chinese remaindering). Let $0 \leq N \leq B$, and $p_1, \ldots, p_n$ be coprime integers. Given n sets $S_1, \ldots, S_n$ where each $S_i = \{r_{i,j}\}_{1 \leq j \leq m}$ contains $m - 1$ random elements in $\mathbb{Z}_{p_i}$ and $N \bmod p_i$, recover the integer N, provided that the solution is unique (*e.g.*, $m^n B \ll \prod_{i=1}^{n} p_i$).

Problem 4 (Chinese remaindering with errors). Given as input integers t, B and n points $(r_1, p_1), \ldots, (r_n, p_n) \in \mathbb{N}^2$ where the p_i's are coprime, output all numbers $0 \leq N < B$ such that $N \equiv r_i \pmod{p_i}$ for at least t values of i.

We refer to [15] for a history of the latter problem, which is beyond the scope of this article. We will only mention that the best decoding algorithm known for the problem is the recent lattice-based work of Boneh [6], which improves

previous work of Goldreich *et al.* [15]. The algorithm works in polynomial time and solves the problem provided that a certain condition is satisfied. The exact condition is analogous to the bound obtained by GS algorithm for polynomial reconstruction.

We note that there are two well-known problems for which the general noisy Chinese remaindering problem (in which one allows different sizes for the sets S_i's) arises. The first problem is point counting on elliptic curves over finite fields. The best general algorithm for this problem is the Schoof-Elkies-Atkin (SEA) algorithm [33, 12, 4, 5] (see [22] for implementation issues). Let E be an elliptic curve over a finite field of cardinality q. Hasse's theorem states that the cardinality of E is of the form $q+1-t$ where $|t| \leq 2\sqrt{q}$. The SEA algorithm tries to determine this t, using Chinese remainders. However, in practice, it turns out to be too expensive to compute the exact value of t modulo sufficiently many coprime numbers. Therefore, one actually determines many coprime numbers of two kinds: for the first kind of numbers, t modulo such numbers is exactly known; for the second kind of numbers, the value of t modulo such numbers is constrained to a small number of values. This is exactly a noisy Chinese remaindering problem. To solve this problem, current versions of SEA apply a meet-in-the-middle strategy. The second problem is integer factorization of numbers of the form $N = p^2q$. It has been noticed for some time (see for instance [28]) that for any number r, the Jacobi symbol $\left(\frac{r}{N}\right)$ is equal to the Legendre symbol $\left(\frac{r}{q}\right)$. It follows that for any number r, $q \bmod r$ is limited to half of $\mathbb{Z}_r$, and such a half can be determined. The problem of computing q can thus be viewed as a noisy Chinese remaindering problem. However, the S_i's are so dense that this formulation is likely to be useless.

We briefly review methods for noisy Chinese remaindering, analogous to the ones we described for noisy polynomial interpolation. One can first use the analog of the meet-in-the-middle method of Section 2.3. One can also use the reduction to Chinese remaindering with errors and the algorithm of [6], in a way analogous to Section 2.1. But the following simpler method achieves the same results.

5.1 Coppersmith's method

We obtain an analogous method to the Gröbner basis approach by translating the problem in terms of polynomial equations. The solution N satisfies for each i the following equation:

$$\prod_{j=1}^{m} (N - r_{i,j}) \equiv 0 \pmod{p_i}.$$

Using Chinese remainders and collecting all equations, one obtains a univariate polynomial equation of degree m in the unknown N modulo $\prod_{i=1}^{n} p_i$. We then apply the following lattice-based result by Coppersmith [8] :

Theorem 2. *Let $P(x)$ be a polynomial of degree δ in one variable modulo an integer M of possibly unknown factorization. In time polynomial in $(\log M, 2^{\delta})$, one can find all integers x_0 such that $P(x_0) \equiv 0 \ (mod M)$ and $|x_0| \leq M^{1/\delta}$.*

In time polynomial in $(\sum_{i=1}^{n} \log p_i, 2^m)$, we can thus find the solution N to noisy Chinese remaindering, provided that : $B^m \leq \prod_{i=1}^{n} p_i$. This condition is analogous to the condition $m < n/k$ we obtained by applying GS algorithm to the noisy polynomial interpolation problem. The method is mentioned in [6].

5.2 Lattice-based methods

Let $P = \prod_{i=1}^{n} p_i$. By analogy to the lattice-based method of section 3, we define interpolation numbers L_i in $\{0, \ldots, P-1\}$ by : $L_i \equiv 1 \pmod{p_i}$ and $L_i \equiv 0 \pmod{\prod_{j \neq i} p_j}$. The solution N of noisy Chinese remaindering satisfies:

$$N \equiv \sum_{i=1}^{n} (N \bmod p_i) L_i \pmod{P}.$$

We linearize the problem: letting $\delta_{i,j}$ equal to 1 if $N \equiv r_i \pmod{p_i}$, and 0 otherwise, one obtains

$$N \equiv \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{i,j} r_{i,j} L_i \pmod{P}.$$

This equation basically says that N is a small subset sum of the $r_{i,j}L_i$'s modulo P. It is thus natural to consider the $(nm+1)$-dimensional lattice L spanned by the rows of the following matrix:

$$\begin{pmatrix} P & 0 & \ldots & \ldots & 0 \\ r_{1,1}L_1 & B & 0 & \ldots & 0 \\ r_{1,2}L_1 & 0 & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ r_{n,m}L_n & 0 & \ldots & 0 & B \end{pmatrix}$$

The lattice L is the set of integer row vectors $(M, d_{1,1}B, d_{1,2}B, \ldots, d_{n,m}B) \in \mathbb{Z}^{nm+1}$ such that $M \equiv \sum_{i=1}^{n} \sum_{j=1}^{m} d_{i,j} r_{i,j} L_i \pmod{P}$. It contains the *target vector* $(N, \delta_{1,1}B, \delta_{1,2}B, \ldots, \delta_{n,m}B)$, which has norm $\sqrt{N^2 + nB^2} \leq B\sqrt{n+1}$. Since the previous matrix is triangular, the volume of L is simply $P \times B^{nm}$. It follows that the target vector is expected to be the shortest vector of L when

$$B\sqrt{n+1} \ll (PB^{nm})^{1/(nm+1)} \approx P^{1/(nm+1)} B,$$

that is $\sqrt{n} \ll P^{1/(nm+1)}$. The condition should however be taken with care, as the lattice L cannot be considered as random. For instance, note that any sufficiently short linear relation between $r_{i,1}, r_{i,2}, \ldots, r_{i,m}$ gives rise to a shorter lattice point. It can be proved that such a case occurs when one of the p_i's is small or one of the $|S_i|$'s is big (using the notion of orthogonal lattice [27], see full version). As with noisy polynomial interpolation, one can improve the lattice L by considering the sublattice Λ of points $(M, d_{1,1}B, d_{1,2}B, \ldots, d_{n,m}B) \in L$ such

that, for all i_1 and i_2, $\sum_{j=1}^{m} d_{i_1,j} = \sum_{j=1}^{m} d_{i_2,j}$. However, the previous obstruction still holds (see full version of the paper). Thus, the lattice-based approach is unlikely to be useful for elliptic curve point counting or integer factorization. Still, the reduction can be proved for certain choices of the parameters, for we have the following analog of lemma 4.

Lemma 6. *Let* $\mathbf{d} = (M, d_{1,1}B, d_{1,2}B, \ldots, d_{n,m}B) \in \mathbb{Z}^{nm+1}$ *satisfying* (1) *and shorter than the target vector. Assume that* $B(m+1)\sqrt{n+1} < P/2$. *Then:*

$$p(\mathbf{d}) \leq q^{-\min(t(\mathbf{d}), n-k)},$$

where $q = \min p_i$, *k is the least positive integer such that* $B(m+1)\sqrt{n+1} < \frac{q^k}{2}$, *and* $t(\mathbf{d}) = \left| \{1 \leq i \leq n : \exists j \in \{1, \ldots, m\} \setminus g(i) \text{ such that } d_{i,j} \not\equiv 0 \bmod p_i\} \right|$.

This lemma is useful, when none of the $|S_i|$'s are big and none of the p_i's are small (which is not the case arising in elliptic curve point counting or integer factorization) in which case one can obtain a provable reduction to the lattice shortest vector problem roughly similar to Theorem 1 since one can upper bound the probability that there exists a nonzero vector strictly shorter than the target vector. In particular, by taking all the p_i's of the same size (such as 32 bits), it is easy to build instances for which the lattice-based approach can experimentally solve noisy Chinese remaindering with a bound B much larger than with Coppersmith's method.

6 Conclusion

We presented various methods to solve the noisy polynomial interpolation problem. In particular, we proved the existence of a reduction from the noisy polynomial interpolation problem to the lattice shortest vector problem, for many choices of the parameters. This reduction appears to be very efficient in practice: experimental evidence suggest that many instances can be solved using standard lattice reduction algorithms. We therefore suggested simple modifications to several cryptographic schemes for which the security assumption relied on the computational hardness of noisy polynomial interpolation. We also briefly discussed analogous methods to solve the related noisy Chinese remaindering problem. The lattice-based approach is the best known method for certain choices of the parameters, but unfortunately not in applications such as elliptic curve point counting or integer factorization. There are several open problems, such as:

- Is there a better[3] reduction from noisy polynomial interpolation or Chinese remaindering to the lattice shortest vector problem ?
- Is there a lattice-based method to solve the polynomial reconstruction problem ?

[3] holding for more or all choices of the parameters.

Acknowledgments

We thank Amin Shokrollahi and Karin Melnick for their help and ideas with error correcting algorithms. We are grateful to Dan Boneh for several enlightening discussions, and for informing us on references [3, 15, 6]. We also thank Louis Granboulan and Andrew Odlyzko for their comments.

References

1. M. Ajtai. Generating hard instances of lattice problems. In *Proc. of 28th STOC*, pages 99–108. ACM, 1996. Available at [11] as TR96-007 report.
2. M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Proc. 30th ACM STOC*, 1998. Available at [11] as TR97-047.
3. S. Ar, R. Lipton, R. Rubinfeld, and M. Sudan. Reconstructing algebraic functions from mixed data. *SIAM J. of Computing*, 28(2):488–511, 1999.
4. A. O. L. Atkin. The number of points on an elliptic curve modulo a prime. Email on the Number Theory mailing list, 1988.
5. A. O. L. Atkin. The number of points on an elliptic curve modulo a prime. Email on the Number Theory mailing list, 1991.
6. D. Boneh. Finding smooth integers in short intervals using CRT decoding. In *Proc. of 32nd STOC*. ACM, 2000.
7. J.H. Conway and N.J.A. Sloane. *Sphere Packings, Lattices and Groups*. Springer-Verlag, 1998. Third edition.
8. D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. of Cryptology*, 10(4):233–260, 1997.
9. M.J. Coster, A. Joux, B.A. LaMacchia, A.M. Odlyzko, C.-P. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Comput. Complexity*, 2:111–128, 1992.
10. N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Proc. of Eurocrypt 2000*, volume 1807 of *LNCS*, pages 398–413. Springer-Verlag, 2000.
11. ECCC. `http://www.eccc.uni-trier.de/eccc/`. The Electronic Colloquium on Computational Complexity.
12. N. D. Elkies. Explicit isogenies. Draft, 1991.
13. C. Ellison, C. Hall, R. Milbert, and B. Schneier. Protecting secret keys with personal entropy. *Future Generation Computer Systems*, 16(4):311–318, 2000. Available at `http://www.counterpane.com/`.
14. N. Gilboa. Two party RSA key generation. In *Proc. of Crypto '99*, volume 1666 of *LNCS*, pages 116–129. Springer-Verlag, 1999.
15. O. Goldreich, D. Ron, and M. Sudan. Chinese remaindering with errors. In *Proc. of 31st STOC*. ACM, 1999. Also available at [11].
16. M. Gruber and C. G. Lekkerkerker. *Geometry of Numbers*. North-Holland, 1987.
17. V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Trans. on Information Theory*, 45(6):1757–1767, 1999. An extended abstract appeared in the Proc. of IEEE FOCS '98.
18. A. Joux and J. Stern. Lattice reduction: A toolbox for the cryptanalyst. *J. of Cryptology*, 11:161–185, 1998.
19. A. Kipnis and A. Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In *Proc. of Crypto '99*, volume 1666 of *LNCS*, pages 19–30. Springer-Verlag, 1999.

20. J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. *Journal of the Association for Computing Machinery*, January 1985.
21. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.
22. R. Lercier and F. Morain. Counting the number of points on elliptic curves over finite fields: strategies and performances. In *Proc. of Eurocrypt '95*, volume 921 of *LNCS*, pages 79–94. Springer-Verlag, 1995.
23. J. E. Mazo and A. M. Odlyzko. Lattice points in high-dimensional spheres. *Monatsh. Math.*, 110:47–61, 1990.
24. D. Micciancio. The shortest vector problem is NP-hard to approximate within some constant. In *Proc. 39th IEEE FOCS*, 1998. Available at [11] as TR98-016.
25. F. Monrose, M. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. In *Proc. of 6th Conf. on Computer and Communications Security*. ACM, 1999.
26. M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. of 31st STOC*, pages 245–254. ACM, 1999.
27. P. Nguyen and J. Stern. Merkle-Hellman revisited: a cryptanalysis of the Qu-Vanstone cryptosystem based on group factorizations. In *Proc. of Crypto '97*, volume 1294 of *LNCS*, pages 198–212. Springer-Verlag, 1997.
28. R. Peralta and E. Okamoto. Faster factoring of integers of a special form. *IEICE Trans. Fund. of Electronics, Communications, and Computer Sciences*, 79(4), 1996.
29. B. Pinkas, 1999. Private communication.
30. C.-P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.
31. C. P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Math. Programming*, 66:181–199, 1994.
32. C.P. Schnorr and H.H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Proc. of Eurocrypt'95*, volume 921 of *LNCS*, pages 1–12. Springer-Verlag, 1995.
33. R. Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7:219–254, 1995.
34. V. Shoup. Number Theory Library (NTL) version 3.7a. Can be obtained at `http://www.shoup.net/ntl/`.
35. C. L. Siegel. *Lectures on the Geometry of Numbers*. Springer-Verlag, 1989.

A Chosen Messages Attack on the ISO/IEC 9796–1 Signature Scheme

François Grieu

Innovatron, 1 rue Danton, 75006 Paris, France
fgrieu@innovatron.fr

Abstract. We introduce an attack against the ISO/IEC 9796–1 digital signature scheme using redundancy, taking advantage of the multiplicative property of the RSA and Rabin cryptosystems. The forged signature of 1 message is obtained from the signature of 3 others for any public exponent v. For even v, the modulus is factored from the signature of 4 messages, or just 2 for $v = 2$. The attacker must select the above messages from a particular message subset, which size grows exponentialy with the public modulus bit size. The attack is computationally inexpensive, and works for any modulus of $16z$, $16z \pm 1$, or $16z \pm 2$ bits. This prompts the need to revise ISO/IEC 9796–1, or avoid its use in situations where an adversary could obtain the signature of even a few mostly chosen messages.

1 Introduction

ISO/IEC 9796–1 [1] [2] is an international standard specifying a digital signature scheme giving message recovery, designed primarily for the RSA and Rabin public key cryptosystems.

To sign a message M, it is first transformed by inserting redundant information obtained by simple transformations of individual bytes of M, producing the expanded message $\tilde{M}$; then the private key function $\mathcal{S}$ of the cryptosystem is applied, producing the signature $\ddot{M} = \mathcal{S}(\tilde{M})$.

To verify an alleged signature $\ddot{M}'$, the public key function $\mathcal{V}$ of the cryptosystem is applied, producing an alleged expanded message $\tilde{M}' = \mathcal{V}(\ddot{M}')$; then the alleged message M' is recovered from $\tilde{M}'$ by straightforward extraction, and it is checked $\tilde{M}'$ is what it should be under the signature production process.

ISO/IEC 9796–1 expansion makes it highly improbable that a randomly generated value is an acceptable signature. It meets precise design criterias in order to guard against a variety of other attacks, see [3] and [2].

The recently introduced Coron–Naccache–Stern forgery strategy of [4] is effective on a slightly simplified variant of ISO/IEC 9796–1. Motivated by this breakthrough and unaware of an extension to the full standard in [6], the author made an independent effort to attack ISO/IEC 9796–1 and discovered a new, simple and effective method.

In a nutshell, we efficiently construct many message pairs A, B with $\tilde{A}/\tilde{B}$ equal to a common ratio. Forgery follows from the multiplicative property of the cryptosystem used: $\mathcal{S}(xy) = \mathcal{S}(x)\mathcal{S}(y)$.

2 Definitions

When there is no ambiguity, we assimilate a bit string of fixed length and the integer having this binary representation. Following ISO/IEC 9796–1 unless stated otherwise, we use the notations

$x \parallel y$	Concatenation of bitstrings x and y.
$x \oplus y$	Bitwise exclusive OR of bitstrings x and y.
$[x]_i$	The bitstring of exactly i bits with $[x]_i \equiv x \bmod 2^i$.
$\mathrm{lcm}(x, y)$	Least Common Multiple of x and y.
$\gcd(x, y)$	Greatest Common Divisor of x and y.
v	Public verification exponent.
k	Number of bits in public modulus.
	NB: the standard [1] often use $k_s = k - 1$.
n	Public modulus of k bits, thus with $2^{k-1} \leq n < 2^k$.
p, q	Secret factors of n, with $n = p\,q$.
	if v is odd, $p - 1$ and $q - 1$ are prime with v.
	if v is even, $(p-1)/2$ and $(q-1)/2$ are prime with v,
	$p \equiv 3 \bmod 4$ and $q \equiv p + 4 \bmod 8$.
$(x\|n)$	Jacobi symbol of x with respect to n, used for even v only.
	$(x\|n) = (x\|p)(x\|q) = (x^{(p-1)/2} \bmod p)(x^{(q-1)/2} \bmod q)$.
	For even v the construction of p and q is such that $(2\|n) = -1$.
	$(x\|n)$ can be efficiently computed without knowledge of p and q.
s	Secret signing exponent.
	if v is odd, $s\,v \equiv 1 \bmod \mathrm{lcm}(p-1, q-1)$,
	and as a consequence $(x^s)^v \equiv x \bmod n$ for any x.
	if v is even, $s\,v \equiv 1 \bmod \mathrm{lcm}(p-1, q-1)/2$,
	and as a consequence $(x^s)^v \equiv x \bmod n$ if $(x\|n) = +1$.
z	Number of bytes a message fits in; $z \leq \lfloor (k+2)/16 \rfloor$.
M	Message to sign, which breaks up into the z bytes string
	$m_z \parallel m_{z-1} \parallel .. \parallel m_2 \parallel m_1$
$\tilde{M}$	Message as expanded according to ISO/IEC 9796–1 (see below).
	NB: $\tilde{M}$ is noted Ir in [1] and also Sr in [2].
$\ddot{M}$	The signature of M. NB: $\ddot{M}$ is noted $\Sigma(M)$ in [1] and [2].
	if v is odd, $\ddot{M} = \min(\tilde{M}^s \bmod n,\ n - \tilde{M}^s \bmod n)$
	if v is even, assuming $\gcd(\tilde{M}, n) = 1$ which is highly probable,
	$\ddot{M} = \min\left(\left(\frac{\tilde{M}}{2^{(1-(\tilde{M}\|n))/2}}\right)^s \bmod n,\ n - \left(\frac{\tilde{M}}{2^{(1-(\tilde{M}\|n))/2}}\right)^s \bmod n\right)$

We restrict our attack and our description of ISO/IEC 9796–1 to the cases $k \equiv 0, \pm 1$, or $\pm 2 \bmod 16$, which covers many common choices of moduli, and to messages of $z = \lfloor (k+2)/16 \rfloor$ bytes, the maximum allowed message size. With these restrictions, the construction of the redundant message amounts to the local transformation of each byte m_i of the message by an injection F_i, yielding the redundant message

$$\tilde{M} = F_z(m_z) \parallel F_{z-1}(m_{z-1}) \parallel .. \parallel F_2(m_2) \parallel F_1(m_1)$$

with the injections F_i transforming an individual byte m_i of two 4 bit digits $x \parallel y$ as defined by

$$\begin{array}{l} F_1(x \parallel y) = \Pi(x) \parallel \Pi(y) \parallel y \parallel [6]_4 \\ F_i(x \parallel y) = \Pi(x) \parallel \Pi(y) \parallel x \parallel y \quad \text{for } 1 < i < z \\ F_z(x \parallel y) = [1]_1 \parallel [\Pi(x)]_{k+2 \bmod 16} \parallel \Pi(y) \parallel x \parallel (y \oplus 1) \end{array} \tag{1}$$

and where Π is the permutation on the set of 4 bit nibbles given by

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\Pi(x)$	E	3	5	8	9	4	2	F	0	D	B	6	7	A	C	1

or as an equivalent definition, if the nibble x consists of the bits $x_4 \parallel x_3 \parallel x_2 \parallel x_1$, $\Pi(x) = (x_4 \oplus x_2 \oplus x_1 \oplus 1) \parallel (x_4 \oplus x_3 \oplus x_1 \oplus 1) \parallel (x_4 \oplus x_3 \oplus x_2 \oplus 1) \parallel (x_3 \oplus x_2 \oplus x_1)$.

3 The new attack

We essentialy select a pair of small positive integers a, b and search all the message pairs A, B that yield redundant messages verifying

$$\frac{\tilde{A}}{\tilde{B}} = \frac{a}{b} \tag{2}$$

3.1 Choice of ratio a/b

Since the ratios a/b and b/a will uncover the same messages, we can restrict our choice of a, b to $a < b$ without missing any message pairs satisfying (2). Similarly, we can restrict ourselves to relatively prime a, b. Since $\tilde{A}$ and $\tilde{B}$ are strings of equal length with a 1 bit on the left, we must have $b < 2a$. We transform equation (2) into $\tilde{B}a = \tilde{A}b$, reduce $\bmod 16$, observe $[\tilde{A}]_4 = [\tilde{B}]_4 = 6$, get $6a \equiv 6b \bmod 16$, so we restrict ourselves to $a \equiv b \bmod 8$.

Thus in the following we restrict our choice for the ratio a/b to relatively prime integers a, b with $9 \leq a < b < 2a$ and $a \equiv b \bmod 8$.

3.2 Making the search manageable

Since the fraction a/b is chosen irreducible, for an hypothetical message pair A, B verifying (2), we can uniquely define the integer W such that

$$\tilde{A} = a\,W \quad \text{and} \quad \tilde{B} = b\,W \tag{3}$$

We break up A, B into z bytes, and, noticing that $9 \leq a < b$ implies $W < 2^{16z}$ for our choice of k, we break up W into z 16 bits strings

$$\begin{array}{l} A = a_z \parallel a_{z-1} \parallel .. \parallel a_2 \parallel a_1 \\ B = b_z \parallel b_{z-1} \parallel .. \parallel b_2 \parallel b_1 \\ W = w_z \parallel w_{z-1} \parallel .. \parallel w_2 \parallel w_1 \end{array}$$

We break up each of the two multiplications appearing in (3) into z multiply and add steps operating on each of the w_i, performed from right to left, with $z-1$ steps generating an overflow to the next step, and a last step producing the remaining left $(k+2 \bmod 16)+13$ bits. We define the *overflows*

$$\begin{array}{ll} \bar{a}_0 = \bar{a}_z = 0 & \bar{b}_0 = \bar{b}_z = 0 \\ \bar{a}_i = \lfloor (a\, w_i + \bar{a}_{i-1})/2^{16} \rfloor & \bar{b}_i = \lfloor (b\, w_i + \bar{b}_{i-1})/2^{16} \rfloor \qquad \text{for } 1 \le i < z \end{array} \tag{4}$$

so we can transform (3) into the equivalent

$$\begin{array}{ll} F_i(a_i) = aw_i + \bar{a}_{i-1} \bmod 2^{16} & F_i(b_i) = bw_i + \bar{b}_{i-1} \bmod 2^{16} \quad \text{for } 1 \le i < z \\ F_i(a_z) = aw_z + \bar{a}_{z-1} & F_z(b_z) = bw_z + \bar{b}_{z-1} \end{array} \tag{5}$$

The search for message pairs A, B satisfying (2) is equivalent to the search of w_i, a_i, b_i, $\bar{a}_i$, $\bar{b}_i$ satisfying (4)(5). This is z smaller problems, linked together by the overflows $\bar{a}_i, \bar{b}_i$.

3.3 Reducing overflows $\bar{a}_i, \bar{b}_i$ to one link l_i

Definition (4) of the overflows $\bar{a}_i, \bar{b}_i$ implies, by induction

$$\bar{a}_i = \left\lfloor \frac{a\,[W]_{16i}}{2^{16i}} \right\rfloor \quad \text{and} \quad \bar{b}_i = \left\lfloor \frac{b\,[W]_{16i}}{2^{16i}} \right\rfloor \quad \text{for } 1 \le i < z \tag{6}$$

Since $0 \le [W]_{16i} < 2^{16i}$ we have

$$0 \le \bar{a}_i < a \quad \text{and} \quad 0 \le \bar{b}_i < b \tag{7}$$

We also observe that $\bar{a}_i$ and $\bar{b}_i$ are roughly in the ratio a/b, more precisely equation (6) implies sucessively

$$a\frac{[W]_{16i}}{2^{16i}} - 1 < \bar{a}_i \le a\frac{[W]_{16i}}{2^{16i}} \quad \text{and} \quad b\frac{[W]_{16i}}{2^{16i}} - 1 < \bar{b}_i \le b\frac{[W]_{16i}}{2^{16i}}$$

$$\frac{\bar{a}_i}{a} \le \frac{[W]_{16i}}{2^{16i}} < \frac{\bar{a}_i + 1}{a} \quad \text{and} \quad \frac{\bar{b}_i}{b} \le \frac{[W]_{16i}}{2^{16i}} < \frac{\bar{b}_i + 1}{b}$$

$$a\frac{\bar{b}_i}{b} - 1 < \bar{a}_i < a\frac{\bar{b}_i + 1}{b} \quad \text{and} \quad b\frac{\bar{a}_i}{a} - 1 < \bar{b}_i < b\frac{\bar{a}_i + 1}{a}$$

so, as consequence of their definition, the $\bar{a}_i$, $\bar{b}_i$ must verify

$$-a < a\bar{b}_i - b\bar{a}_i < b \tag{8}$$

For a given $\bar{b}_i$ with $0 \le \bar{b}_i < b$, one or two $\bar{a}_i$ are solution of (8): $\lfloor a\,\bar{b}_i/b \rfloor$, and $\lfloor a\,\bar{b}_i/b \rfloor + 1$ if and only if $a\,\bar{b}_{i-1} \bmod b > b - a$.

It is handy to group $\bar{a}_i$, $\bar{b}_i$ into a single *link* defined as

$$l_i = \bar{a}_i + \bar{b}_i + 1 \qquad \text{with} \quad 1 \le l_i < a + b \tag{9}$$

so we can rearrange (8) into

$$\bar{a}_i = \left\lfloor \frac{a\,l_i}{a+b} \right\rfloor \quad \text{and} \quad \bar{b}_i = \left\lfloor \frac{b\,l_i}{a+b} \right\rfloor \tag{10}$$

3.4 Turning the problem into a graph traversal

For $1 \le i \le z$ we define the z sets of triples

$$T_i = \{(l_i, w_i, l_{i-1}) \quad \exists (a_i, b_i, \bar{a}_i, \bar{b}_i, \bar{a}_{i-1}, \bar{b}_{i-1}) \quad \text{verifying (4)(5)(7)(9)(10)}\}$$

and we define that $(l_i, w_i, l_{i-1}) \in T_i$ connects to $(l'_j, w'_j, l'_{j-1}) \in T_j$ when $j = i-1$ and $l_{i-1} = l'_j$. Solving (2) is equivalent to finding a connected path from an element of T_z to an element of T_1. If this can be achieved, a suitable W is obtained by concatenating the w_i in the path, and $\tilde{A}$, $\tilde{B}$ follow from (3).

3.5 Building and traversing the graph

The graph can be explored in either direction with about equal ease, we describe the right to left procedure.

Initialy we start with the only link $l_0 = 1$. At step $i = 1$ and growing, for each of the link at the previous step, we vary b_i in range $[0..2^8 - 1]$ and directly compute

$$w_i = \left(F_i(b_i) - \left\lfloor \frac{b\, l_{i-1}}{a+b} \right\rfloor \right) b^{-1} \bmod 2^{16} \tag{11}$$

Using an inverted table of F_i we can determine in one lookup if there exist an a_i such that

$$F_i(a_i) = a\, w_i + \left\lfloor \frac{a\, l_{i-1}}{a+b} \right\rfloor \bmod 2^{16} \tag{12}$$

and in that case we remember the new triple (l_i, w_i, l_{i-1}) with the new link

$$l_i = \left\lfloor \frac{a\, w_i + \left\lfloor \frac{a\, l_{i-1}}{a+b} \right\rfloor}{2^{16}} \right\rfloor + \left\lfloor \frac{b\, w_i + \left\lfloor \frac{b\, l_{i-1}}{a+b} \right\rfloor}{2^{16}} \right\rfloor + 1 \tag{13}$$

We repeat this process until a step has failed to produce any link, or we reach $i = z$ where we need to modify (11)(12)(13) by replacing the term 2^{16} by $2^{(k+2 \bmod 16)+13}$, and reject nodes where $l_z \neq 1$.

If we produce a link in the last step $i = z$, we can obtain a solution to (2) by backtracking any path followed, and the resulting graph covers every solutions.

Exploration for the simplest ratio 9/17 stops on the first step, but 11/19 is more fruitfull. For $k = 256$, and restricting to nodes belonging to a solution, we can draw the graph in figure 1.

Using this graph to produce solutions to (2) is childishly simple: message pairs are obtained by choosing a path between terminals nodes, and collecting the message bytes a_i (resp. b_i) shown above (resp. below) the nodes[1].

[1] As a convenience we have shown the bytes a_i, b_i of messages A, B instead of the triples (l_i, w_i, l_{i-1}).

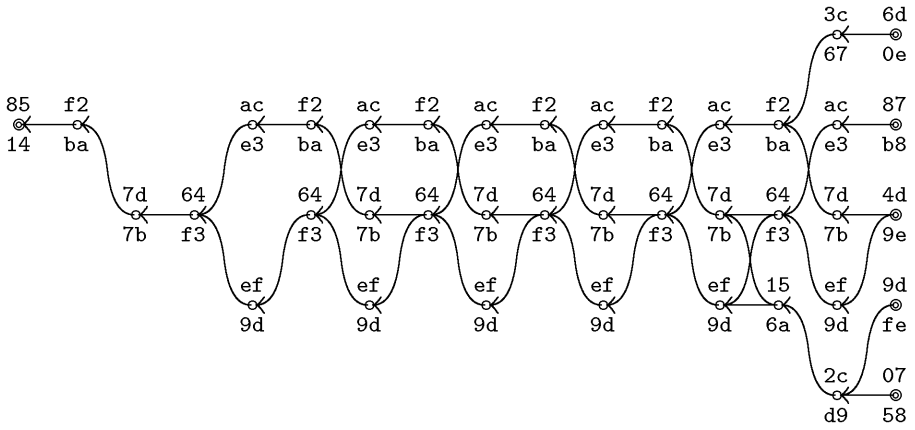


Fig. 1. Graph of solutions of (2) for $k = 256$ and $a/b = 11/19$

For example, if we follow the bottom link, the graph gives messages
A=85f27d64ef..64ef..64ef152c07
B=14ba7bf39d..f39d..f39d6ad958 thus
$\tilde{A}$=458515f2fa7d2964c1ef..2964c1ef..2964c1ef3415572cef76
$\tilde{B}$=78146bbaf67b18f3da9d..18f3da9d..18f3da9d2b6aadd94086
with indeed $\tilde{A}/\tilde{B} = 11/19$.

3.6 Counting solutions

It is easy to count the solutions: assign the count 1 to right nodes, and to all others the sum of the count of their right-linked nodes. The number of solutions to (2) is the sum of the count of the left nodes. This gives 42 for the graph above, which Douglas Adams fans will appreciate.

Since the center part of the graph has a period of two steps, it is trivial to extend it for higher k with $k \equiv 0 \bmod 32$. Asymptoticaly, this count grows by a factor $\frac{\sqrt{5}+1}{2}$ when the modulus is increased by 32 bits.

If we take $k = 1024$ bits and restrict to $b < 2^{10}$, there are 13264 ratios worth to explore. About 40% are eliminated on the first step, 9% have at least one solution to (2), 7% have at least two solutions. There are about $5.7\,10^{14}$ usable message pairs, among which 98% come from the ratio 389/525 which yields 2^{49} solutions. The code computing the above statistics runs in a minute on a personal computer, and can output thousands of messages per second.

Lower bounds on the number of pairs of solutions to (2) are derived by counting solutions for a good ratio

$2^{(k-164.7..)/16}$	solutions for $k \equiv -2 \bmod 16$	using ratio 389/525
$1.62177..^{(k-148.3..)/16}$	solutions for $k \equiv -1 \bmod 16$	using ratio 511/775
$2^{(k-240)/16}$	solutions for $k \equiv 0 \bmod 16$	using ratio 389/525
$1.62177..^{(k-227.6..)/16}$	solutions for $k \equiv 1 \bmod 16$	using ratio 511/775
$2^{(k-226)/16}$	solutions for $k \equiv 2 \bmod 16$	using ratio 389/525

3.7 Existential forgery from the signature of 3 chosen messages

By selecting a ratio a/b and finding two messages pairs A, B and C, D solutions of (2), we can now construct 4 messages A, B, C, D such that

$$\tilde{A}\,\tilde{D} = \tilde{B}\,\tilde{C} \tag{14}$$

With high probability, $\tilde{A}$ and n are relatively prime[2], so that

$$\tilde{D}^s \equiv \tilde{A}^{-s}\tilde{B}^s\tilde{C}^s \bmod n \tag{15}$$

and therefore, for odd v,

$$\ddot{D} = \min(\ddot{A}^{-1}\ddot{B}\,\ddot{C} \bmod n,\ n - \ddot{A}^{-1}\ddot{B}\,\ddot{C} \bmod n) \tag{16}$$

If we can obtain the three signatures $\ddot{A}$, $\ddot{B}$, $\ddot{C}$, it is now straightforward to compute $\ddot{D}$, using the extended Euclidian algorithm for the modular inversion of $\ddot{A} \bmod n$.

For even v, equation (15) implies

$$\ddot{D} = \min(2^{js}\ddot{A}^{-1}\ddot{B}\,\ddot{C} \bmod n,\ n - 2^{js}\ddot{A}^{-1}\ddot{B}\,\ddot{C} \bmod n) \tag{17}$$

$$\text{with } j = \frac{(\tilde{A}|n)-1}{2} + \frac{1-(\tilde{B}|n)}{2} + \frac{1-(\tilde{C}|n)}{2} + \frac{(\tilde{D}|n)-1}{2}$$

If $(a|n) = (b|n)$ then $(\tilde{A}|n) = (\tilde{B}|n)$ and $(\tilde{C}|n) = (\tilde{D}|n)$ thus j is always 0. If $(a|n) = -(b|n)$ then j is -2, 0 or 2, the case $j = 0$ has probability about $1/2$, and it is necessary to examine at most three message pairs before finding two such that $j = 0$. When $j = 0$, equation (17) reduces to (16) and again we obtain a forgery from three signatures.

In summary we have one forgery from three signatures for any public exponent. Using the terminology in [8], it is a chosen messages existential forgery, in that the adversary is bound to pick from a predefined subset the messages submited for signature and the bogus message. More generaly, f forgeries can be obtained from $f + 2$ signatures.

3.8 Total break from the signature of 4 chosen messages for even v

As pointed out in [7], for even public exponents v, finding a multiplicative relation among expanded messages can lead to factorisation of the public modulus n.

We select a ratio a/b such that $(a|n) = -(b|n)$, which for a given n occurs for about half the ratios. We then test solutions of (2) until we find two messages pairs A, B and C, D solutions of (2) verifying $(\tilde{A}|n) = 1$ and $(\tilde{C}|n) = -1$, with the probability of not finding a solution about halved after each trial. For even v, equation (15) implies

$$2^{2s} = u \quad \text{with } u = \pm\ddot{A}\ddot{B}^{-1}\ddot{C}^{-1}\ddot{D} \bmod n \tag{18}$$

[2] else we would get a prime factor of n by computing $\gcd(\tilde{A}, n)$

where the term u is known[3]. Taking the above to the known power $v/2$ and reducing $\bmod p$ gives

$$u^{v/2} \equiv 2^{vs} \equiv 2^{\frac{p-1}{2}\frac{q-1}{2}+1} \equiv (2|p)^{\frac{q-1}{2}}\ 2 \equiv (2|p)\ 2 \bmod p$$

and similarly

$$u^{v/2} \equiv (2|q)\ 2 \bmod q.$$

Noticing that one of p or q is 3 mod 8 and the other is 7 mod 8, we have $(2|p) = -(2|q)$. We deduce that $(u^{v/2}+2) \bmod n$ is a multiple of only one of p or q.

Therefore a prime factor of n is $\gcd(\ddot{A}^{v/2}\ddot{B}^{-v/2}\ddot{C}^{-v/2}\ddot{D}^{v/2}+2 \bmod n,\ n)$.

If we can obtain the four signatures $\ddot{A}$, $\ddot{B}$, $\ddot{C}$, $\ddot{D}$ we can thus factor the modulus n. Of course this let us compute a valid signing exponent s then sign any message just as easily as the legitimate signer, a total break using the terminology in [8].

3.9 Reducing the number of required signatures for small v

Assume we can find two messages A, B solution of

$$\frac{\tilde{A}}{\tilde{B}} = \frac{c^v}{d^v} \quad \text{with } c \neq d \tag{19}$$

This implies

$$\tilde{A}^s d^{vs} \equiv \tilde{B}^s c^{vs} \bmod n \tag{20}$$

For odd v, it follows that

$$\ddot{B} = \min(c^{-1}d\ \ddot{A} \bmod n,\ n - c^{-1}d\ \ddot{A} \bmod n) \tag{21}$$

and we obtain one forgery from a single signature.

For even v, we can similarly obtain forgery from a single signature if $(c|n) = (d|n)$, or factor the modulus from two signature if $(c|n) = -(d|n)$.

Solutions to (19) can be found for $v = 2$. For example with $v = 2$ and $k = 1024$, 21 among the 1933 irreducible ratios with $d < 2^{16}$ give 22645 message pairs, among which 16059 for the ratio $19^2/25^2$. An example for $k = 512$ is:
`ECE8F706C09CA276A3FC8F00803C821D90A3C03222C37DE26F5C3FD37A886FE4`
`CA969C94FA0B801DDEEA0C22932D80570F95A9C767D27FA8F06A56E7371B16DF`

For $v = 3$ the search becomes more difficult, with only 7 ratios and message pairs for $d < 2^{16}$ and $510 \leq k \leq 2050$, and many values of k without a solution. An example is $k = 510$ and ratio $49^3/57^3$ which gives the message pair:
`C6C058A3239EE6D5ED2C4D17588B02B884A30D92B5D414DDB4B5A6DA58B6901B`
`20768B854644F693DB1508DE0124B4457CD7261DF699F422D9634D5E4D5781A4`

[3] within sign; we could recover the sign, but it is not needed.

3.10 Constraints on signed messages

We have seen the number of usable message pairs is huge, and grows exponentialy with the modulus bit size, by a factor of 2 or 1.62177.. for every 16 bit of modulus. The attacker can, with remarkable ease, select among this mass those messages obeying a particular constraint, simply by restricting the range of bytes allowed at some stage in the graph construction.

For example with $k = 0 \bmod 16$ the ratio 389/525 generates many mostly ASCII message pairs, like

`2B0D59B00D060D8FF65300B56A3A3D3D3D3D3D3D3D3D3D3D3D3D3D..3D3D3D37`
`A50F7D50962A02BDE981A4B28D9F5A5A5A5A5A5A5A5A5A5A5A5A..5A5A5A26`

If we restrict all message bytes to $[32..126] \cup [160..255]$, a subset of what Windows considers displayable, it is still easy to generate messages pairs; for example with $k = 512$ the ratio 169/217 gives 682 message pairs like

`5374FC56DEA856DEA856DEA856DEA856DEA856DEA856DEA856DEA856439F22CF`
`27D36E26425A26425A26425A26425A26425A26425A26425A26425A26CD1EB6F1`

and

`53A856DEA856DEA856DEA856DEA856DEA856DEA856DEA856DE74FCA3C7711BAF`
`275A26425A26425A26425A26425A26425A26425A26425A2642D36E0D81C70B21`

3.11 Generality of the attack

The idea of searching solutions to (2) could apply to other redundancy schemes, though no actually used system comes to mind. The same search principle efficiently finds the solutions, *if they exist*, for any redundancy scheme that operate on independant segments of the message, regardless of the tranformations used, as long as the width of the individual segments do no prevent exhaustive search. The search can be conducted sequentialy in either direction, and works equally well if the redundancy added is dependant on the whole portion of the message on some fixed side of the segment, rather than on the segment alone.

Experimentally, the existence of solutions to (2) appears independant of the particular permutation $\Pi(x)$. It does depends to some degree on the repeated use of the same local injection, because that makes the center of the graph more regular. It does depend heavily on an amount of redundancy not markedly exceeding the message itself.

4 Future work

4.1 Other parameters

We have restricted our attack to $k \equiv 0, \pm 1$, or $\pm 2 \bmod 16$ and to messages of $z = \lfloor (k+2)/16 \rfloor$ bytes, the maximum allowed message size. The difficulty appears to increase quickly as the message gets shorter than half the modulus. The attack does works without modification for messages a few bits shorter, and maybe could be extended to any value of k.

4.2 Attack of 'massive mask changes' variants

As a countermeasure against the attack of [4], it has been envisioned in [5] to use not only three injections like in the original standard, but z injections F_i depending on i. Although the above search method applies, the author did not yet establish if (2) has solutions for some ratios a/b with the particular variants[4] proposed.

4.3 Combination with other attacks

Other attacks against ISO/IEC 9796–1 introduced in [4] then perfected in [6] construct messages M which expanded form $\tilde{M}$ is the product of a common constant Γ and small prime factors, then by gaussian elimination find a multiplicative relation similar to 14, although among thousands messages.

The technique we describe can be used to efficently find messages satisfying (2) where a and b only have small prime factors. This gives a relation readily usable in the gaussian elimination process. The combined attack can operate on a wider range of messages, yet still has modest computing requirements.

5 Conclusion

Our attack applies to the full ISO/IEC 9796–1 standard, with common parameters: public modulus of $16z$, $16z \pm 1$, or $16z \pm 2$ bits, and messages of $8z$ bits. Using an inexpensive graph traversal, we constructs 2 messages pairs which expansion are in a common ratio, giving 4 messages which signatures are in a simple multiplicative relation.

For any public exponent v, the attack obtains the forged signature of 1 such message from the legitimate signature of 3 chosen others, or asymptotically nearly one forgery per legitimate signature; it is a major concern for example if obtaining a signature is possible for a price, and forged signatures have a value for messages the attack applies to.

For even v, the attack is a total break in situations where an attacker can obtain the signature of 4 chosen messages (or just 2 for $v = 2$). It is a major concern for example if the attacker can gain limited access to a signing device accepting arbitrary messages, as likely with an off-the-shelf Smart Card implementation of ISO/IEC 9796–1.

The messages the attack can use are computationally easy to generate. Their number grows exponentialy with the modulus size. Messages can efficiently be found including with a small degree of constraint on the message structure.

This prompts the need to revise ISO/IEC 9796–1, or avoid its use in situations where an adversary could obtain the signature of even a few mostly chosen messages.

[4] Remarking that $\Pi(x \oplus y) = \Pi(x) \oplus \Pi(y) \oplus \Pi(0)$, two of the three variants differ only by the choice of arbitrary constants.

References

1. ISO/IEC 9796:1991. Information technology – Security techniques – Digital signature scheme giving message recovery, 1991.
   See also <http://www.iso.ch/jtc1/sc27/27sd799a.htm#9796>.
2. ISO/IEC 9796–1 Second edition Final Committee Draft. Information technology – Security techniques – Digital signature scheme giving message recovery – Part 1: Mechanisms using redundancy.
   Circulated as ISO/IEC JTC1/SC27 N2175 (1998).
3. Guillou, L. C. and Quisquater, J. J. and Walker, M. and Landrock, P. and Shaer, C.: Precautions taken against various potential attacks in ISO/IEC DIS 9796. Advances in Cryptology - EuroCrypt '90 (1990) 465–473.
4. Coron, J. S. and Naccache, D. and Stern, J. P.: A new signature forgery strategy applicable to ISO 9796–1/2, ECASH™, PKCS#1 V2.0, ANSI X9.31, SSL-3.02.
   Circulated as ISO/IEC JTC1/SC27 N2329 alias WG2 N429 (1999).
5. Coppersmith, D. and Halevi, S. and Jutla, C.: Some countermeasures against the new forgery strategy (Working Draft).
   Circulated as ISO/IEC JTC1/SC27 N2362 (1999).
6. Coppersmith, D. and Halevi, S. and Jutla, C.: ISO 9796–1 and the new forgery strategy (Working Draft) (1999).
   See <http://grouper.ieee.org/groups/1363/contrib.html>.
7. Joye, M. and Quisquater, J.J.: On Rabin-type Signatures (Working Draft)
   Circulated as ISO/IEC JTC1/SC27/WG2 N449 (1999).
8. Menezes, A. and van Oorschot, P. and Vanstone, S.: Handbook of Applied Cryptography (1997). CRC Press, ed.
   See <http://cacr.math.uwaterloo.ca/hac/>.

Cryptanalysis of Countermeasures Proposed for Repairing ISO 9796-1

Marc Girault and Jean-François Misarsky

France Télécom - CNET
42 rue des Coutures, B.P. 6243
14066 CAEN CEDEX 4, FRANCE
{marc.girault, jeanfrancois.misarsky}@cnet.francetelecom.fr

Abstract. ISO 9796-1, published in 1991, was the first standard specifying a digital signature scheme with message recovery. In [4], Coron, Naccache and Stern described an attack on a slight modification of ISO 9796-1. Then, Coppersmith, Halevi and Jutla turned it into an attack against the standard in full [2]. They also proposed five countermeasures for repairing it. In this paper, we show that all these countermeasures can be attacked, either by using already existing techniques (including a very recent one), or by introducing new techniques, one of them based on the decomposition of an integer into sums of two squares.

1 Introduction: ISO 9796-1 and Forgery

The first standard on digital signature scheme with message recovery is ISO 9796-1 [10]. At the end of 80's, no hash-function standard was available. Consequently, ISO 9796-1 used only redundancy function to resist attacks that exploit the multiplicative property of the RSA cryptosystem. The precautions taken in this standard are described in [8]. Until the *rump session* of Crypto '99, no known attack [13] was able to forge a signature complied with the ISO 9796-1 standard.

1.1 The ISO 9796-1 Standard

This standard specifies how a message m is encoded to a valid message $\mu_{iso}(m)$ before applying the RSA signature function. Only redundancy is used, no hash-function. Notations used in this paper to describe encoded functions are the same as in [2]:

- $s(x)$: the function mapping 4 bits of message to 4 bits of redundancy. It is an Hamming code (8, 8, 4).
- $\bar{s}(x)$: the result of setting the most significant bit of $s(x)$ to 1:

$$\bar{s}(x) = s(x) \text{ OR } 1000 \ . \tag{1}$$

- $\tilde{s}(x)$: the result of flipping the least significant bit of $s(x)$:

$$\tilde{s}(x) = s(x) \oplus 0001 \ . \tag{2}$$

When the length of the modulus is $16z+1$ bits and the length of the message is $8z+1$ bits, the encoding function, or redundancy function μ_{iso}, is defined as follows:

$$\begin{aligned}\mu_{iso}(m) = \; & \bar{s}(m_{l-1})\tilde{s}(m_{l-2})m_{l-1}m_{l-2} \\ & s(m_{l-3})s(m_{l-4})m_{l-3}m_{l-4} \\ & \ldots \\ & s(m_3)s(m_2)m_3m_2 \\ & s(m_1)s(m_0)m_0 6 \ .\end{aligned} \tag{3}$$

1.2 Attack Against a Slight Modification of ISO 9796-1

At first, a new strategy of forgery was presented at Crypto '99 by Coron, Naccache and Stern in their paper [4]. They described an attack against a slight modification of ISO 9796-1. Their forgery is possible when the length of the modulus is $16z+1$ bits, the length of the message is $8z+1$ bits, and the valid message $\mu(m)$ is defined as follows:

$$\begin{aligned}\mu(m) = \; & \bar{s}(m_{l-1})s(m_{l-2})m_{l-1}m_{l-2} \\ & s(m_{l-3})s(m_{l-4})m_{l-3}m_{l-4} \\ & \ldots \\ & s(m_3)s(m_2)m_3m_2 \\ & s(m_1)s(m_0)m_0 6 \ .\end{aligned} \tag{4}$$

Remark 1. $\mu(m) = \mu_{iso}(m)$ except that $\tilde{s}(m_{l-2})$ is replaced by $s(m_{l-2})$.

1.3 Attack Against ISO 9796-1 and Countermeasures

At the *rump session* of Crypto '99, Coppersmith, Halevi and Jutla described a modified version of the attack of Coron, Naccache and Stern to forge a signature of a chosen message when the encoding function μ_{iso} of ISO standard is used, i.e. (3). After Crypto conference, they submitted a contribution [2] to "IEEE P1363 research contributions". In their paper, they proposed five possible countermeasures to avoid forgeries. Their solutions avoid Coron-Naccache-Stern-like forgeries, but not all forgeries as we show now.

More precisely, we present various chosen messages attacks against all the five countermeasures, in which the signatures of two (or three) messages chosen by the enemy allow him to forge the signature of another one.

2 Massive Mask Changes

Coppersmith, Halevi and Jutla propose three solutions based on the massive mask change technique. In their propositions, they use the same principle of dispersion as in ISO 9796-1.

Remark 2. These three propositions allow message recovery, but nothing notifies the length of the message. In ISO 9796-1 [10], a nibble was modified in order to mark the length of the message.

2.1 μ_1: Fixed Redundancy

In the first proposition, μ_1, only fixed redundancy is used. The i'th nibble, π_i in the hexadecimal expansion of the irrational number $\pi = 3.14159...$, is used to obtain redundancy. Note that the number of bits of redundancy is half the number of bits of the RSA modulus n.

$$\begin{aligned}\mu_1(m) = {} & \pi_{l-1}\pi_{l-2}m_{l-1}m_{l-2} \\ & \pi_{l-3}\pi_{l-4}m_{l-3}m_{l-4} \\ & \dots \\ & \pi_1\pi_0 m_1 m_0 \ .\end{aligned} \tag{5}$$

The Coron-Naccache-Stern-like forgeries are avoided. But we are at the limit of the efficiency of the forgery described in [12], which allows to find three messages m_1, m_2, m_3 such that $\mu_1(m_1)\mu_1(m_2) = \mu_1(m_3) \pmod n$ and therefore, given signatures of m_1 and m_2, forge the signature of m_3. Moreover, the limit of this attack is heuristic. Consequently, the forgery in [12] may be used.

2.2 μ_2 and μ_3: Irrational Numbers and Exclusive-OR

With μ_2 and μ_3, the attacks based on the Coron-Naccache-Stern forgery [4], [2], are also avoided. In these cases, the i'th nibbles, π_i and e_i in the hexadecimal expansion of the irrational numbers $\pi = 3.14159...$ and $e = 2.71828...$ respectively, are used. Moreover, the native redundancy of ISO 9796-1 is present and plays its role to defeat the other forgeries [13].

$$\begin{aligned}\mu_2(m) = {} & (\pi_{l-1} \oplus s(m_{l-1}))(\pi_{l-2} \oplus s(m_{l-2}))m_{l-1}m_{l-2} \\ & (\pi_{l-3} \oplus s(m_{l-3}))(\pi_{l-4} \oplus s(m_{l-4}))m_{l-3}m_{l-4} \\ & \dots \\ & (\pi_1 \oplus s(m_1))(\pi_0 \oplus s(m_0))m_1 m_0 \ .\end{aligned} \tag{6}$$

$$\begin{aligned}\mu_3(m) = {} & (\pi_{l-1} \oplus s(m_{l-1} \oplus e_{l-1}))(\pi_{l-2} \oplus s(m_{l-2} \oplus e_{l-2}))m_{l-1}m_{l-2} \\ & (\pi_{l-3} \oplus s(m_{l-3} \oplus e_{l-3}))(\pi_{l-4} \oplus s(m_{l-4} \oplus e_{l-4}))m_{l-3}m_{l-4} \\ & \dots \\ & (\pi_1 \oplus s(m_1 \oplus e_1))(\pi_0 \oplus s(m_0 \oplus e_0))m_1 m_0 \ .\end{aligned} \tag{7}$$

Nevertheless, a new attack by Grieu [6], disclosed in October 1999, can be applied to these functions of redundancy. This attack is originally against the ISO 9796-1 [10], but the principle of this attack can be used to forge a signature when μ_2 or μ_3 is the redundancy function in a signature scheme. This forgery is based on the multiplicative property of the RSA cryptosystem and, for any public exponent, the forged signature of a message is obtained from the signature of three other messages. This attack is computationally inexpensive and works for modulus of $16z$, $16z \pm 1$, or $16z \pm 2$ bits.

3 Length Expanding Encoding: μ_4

The encoded function μ_4 involves encoding the message m into a string longer than the modulus n. This solution does not have the property of message recovery. Two constants c_0 and c_1 are fixed, each half the length of the modulus n. The message m is also half the length of the modulus. The redundancy function μ_4 is defined[1] as follows :

$$\mu_4(m) = (m + c_0)||(m + c_1)||m \ . \tag{8}$$

We can easily write μ_4 as an affine function:

$$\begin{aligned} \mu_4(m) &= (m + c_0)||(m + c_1)||m \\ &= (m + c_0)2^\alpha + (m + c_1)2^\beta + m \\ &= m \underbrace{(2^\alpha + 2^\beta + 1)}_{\omega} + \underbrace{c_0 2^\alpha + c_1 2^\beta}_{a} \\ &= m\omega + a \ . \end{aligned} \tag{9}$$

We are at the limit of the efficiency of the forgery described in [5] against signature scheme with an affine function of redundancy. This forgery allows to find three messages m_1, m_2, m_3 such that $\mu_4(m_1)\mu_4(m_2) = \mu_4(m_3) \pmod n$ and therefore, given signatures of m_1 and m_2, forge the signature of m_3. Moreover, the limit of this attack is heuristic. Consequently, the forgery in [5] may be used.

4 Encoding via Squaring: μ_5

The redundancy function μ_5 is defined as follows :

$$\mu_5(m) = m^2 + \delta \ . \tag{10}$$

where δ is a fixed random constant of about the same size as the RSA modulus n and the message m is less than the square root of the modulus n. We present two forgeries when μ_5 is used.

First forgery[2]: Forges the signature of the message $(m_1 m_2 + \delta \pmod n)$ with the signatures of m_1 and m_2 such that:

$$m_2 = m_1 + 1 \ . \tag{11}$$

Second forgery: Forges the signature of a message in the set $\{x, y, z, t\}$ when we can write $A = 2(n - \delta)$ as at least two different sums of two squares:

$$A = x^2 + y^2 = z^2 + t^2 \quad (x, y) \neq (z, t) \text{ and } (y, x) \neq (z, t) \ . \tag{12}$$

[1] The symbol || denotes the concatenation of two strings.
[2] Discovered independently by D. Naccache.

4.1 First Forgery

Let m_1 and m_2 be two messages such that :

$$m_2 = m_1 + 1 \ . \tag{13}$$

Then we have :

$$\begin{aligned}
\mu_5(m_1)\mu_5(m_2) &= (m_1^2 + \delta)(m_2^2 + \delta) \\
&= (m_1 m_2)^2 + \delta(m_1^2 + m_2^2) + \delta^2 \\
&= (m_1 m_2 + \delta)^2 - 2m_1 m_2 \delta + \delta(m_1^2 + m_2^2) \\
&= (m_1 m_2 + \delta)^2 + \delta(m_1^2 - 2m_1 m_2 + m_2^2) \\
&= (m_1 m_2 + \delta)^2 + \delta \underbrace{(m_1 - m_2)^2}_{1} \\
&= \mu_5(m_1 m_2 + \delta) \\
&= \mu_5(m_1 m_2 + \delta \pmod n) \pmod n \ .
\end{aligned} \tag{14}$$

Now, we can find m_1 and m_2 s.t. $m_1 m_2 + \delta \pmod n$ is less than $\sqrt{n}$ by choosing m_1 close enough to $\sqrt{n-\delta}$. More precisely, let $m_1 = \sqrt{n-\delta} + \theta$ such that $\theta \in \left[-\frac{1}{2}, \frac{1}{2}\right]$. Then :

$$\begin{aligned}
m_1 m_2 + \delta &= m_1(m_1 + 1) + \delta \\
&= m_1^2 + m_1 + \delta \\
&= (\sqrt{n-\delta} + \theta)^2 + (\sqrt{n-\delta} + \theta) + \delta \\
&= (2\theta + 1)\sqrt{n-\delta} + \theta(\theta + 1) \pmod n \ .
\end{aligned} \tag{15}$$

and will be certainly (resp. possibly) smaller than $\sqrt{n}$ if $\theta \in \left[-\frac{1}{2}, 0\right]$ (resp. if $\theta \in \left]0, \frac{1}{2}\right]$). Of course, other values of m_1 and m_2 can be suitable, depending on the value of $\sqrt{n-\delta}$. Moreover, one can choose a large value for θ as long as $m_1 m_2 + \delta \pmod n$ is less than $\sqrt{n}$.

4.2 Second Forgery

The second forgery uses the fact that many integers can be written as sums of two squares in (at least) two different ways. This will be applied to various values of $A = 2(n-\delta)$, where n is a RSA modulus. Roughly speaking, if we can write:

$$A = x^2 + y^2 = z^2 + t^2, \quad (x, y) \text{ and } (y, x) \neq (z, t) \ . \tag{16}$$

then it comes (see (25)):

$$\mu_5(x)\mu_5(z) = \mu_5(y)\mu_5(t) \pmod n \ . \tag{17}$$

and the signature of any message in the set $\{x, y, z, t\}$ can be deduced from the signatures of the three other ones. To do that, we first need to recall some basic results from (computational) number theory.

The sum of two squares in two ways. In 17th century, Fermat proved that every prime p such that $p = 1 \pmod 4$ has a unique decomposition as a sum of two squares and, more generally, that an integer n has such a decomposition if and only if all its prime factors such that $p = 3 \pmod 4$ have even exponents in the factorization of n. In the latter case, the number of essentially different decompositions[3] is 2^{k-1}, where k is the number of primes such that $p = 1 \pmod 4$ [9]. Here, we will be specially interested in the case $k \geq 2$.

Remark 3. (Gauss) If a number n can be written as a sum of squares then n has $\left\lceil \frac{\prod_i (e_i+1)}{2} \right\rceil$ representations[4] [7, section 182] where the e_i are the powers of the prime factors p_i of n such that $p_i = 1 \pmod 4$.

Diophante's identities are crucial in the proof of these theorems. We recall them:

$$\begin{aligned}(a^2+b^2)(c^2+d^2) &= (ac-bd)^2+(bc+ad)^2 = e_1^2+f_1^2 \\ &= (ac+bd)^2+(bc-ad)^2 = e_2^2+f_2^2 \ .\end{aligned} \tag{18}$$

They show that the product of two sums of two squares is still the sum of two squares, and in two different ways (see example 1). There is an exception to the latter statement: if one of the initial sums is equal to 2 $(= 1^2+1^2)$, then the two identities become only one, and the decomposition remains the same (see example 2).

Example 1.

$$\begin{aligned}13.17 &= (2^2+3^2)(4^2+1^2) \\ &= (2.4-3.1)^2+(3.4+2.1)^2 = 5^2+14^2 \\ &= (2.4+3.1)^2+(3.4-2.1)^2 = 11^2+10^2 \ .\end{aligned} \tag{19}$$

Example 2.

$$\begin{aligned}2.13 &= (1^2+1^2)(2^2+3^2) \\ &= (1.2-1.3)^2+(1.2+1.3)^2 = \ 1^2+5^2 \\ &= (1.2+1.3)^2+(1.2-1.3)^2 = 5^2+1^2 \ .\end{aligned} \tag{20}$$

Now, the point is to make, when existing, these decompositions efficient. In 1908, Cornacchia [3] showed how to use Euclid's algorithm to find the decomposition of a prime p equal to 1 modulo 4 [1, pages 34-35], [15]. It can be briefly described as follows: find a square root z of -1 modulo p, then apply Euclid's algorithm to p and z, until the remainder x is smaller than $\sqrt{p}$. Then it can be proven that $p - x^2$ is a square y^2 and we have: $p = x^2 + y^2$.

Finally, it is trivial to remark that the product of a square and of a sum of two squares is still a sum of two squares:

$$C^2(x^2+y^2) = (Cx)^2+(Cy)^2 \ . \tag{21}$$

[3] $n = a^2+b^2$ with $\gcd(a, b) = 1$ and $(a, b) \in \mathbb{N}^* \times \mathbb{N}^*$.
[4] $n = a^2+b^2$ with $(a, b) \in \mathbb{N} \times \mathbb{N}$.

As a consequence of all these facts, if we can write $A = 2(n-\delta)$ as a product in the form:

$$C^2 \prod_{i=1}^{k} p_i \quad \text{or} \quad 2C^2 \prod_{i=1}^{k} p_i \quad . \tag{22}$$

where the p_i are equal to 1 modulo 4 and $k \geq 2$, then, by applying Cornacchia's algorithm to the p_i and applying Diophante's identities to its outputs, we will obtain at least 2^{k-1} different decompositions of A in sums of two squares.

Example 3. $n = 493 = 17.29$ and $\delta = 272$.
Then $A = 2(n-\delta) = 2.13.17$.
We have $13 = 2^2 + 3^2$, $17 = 4^2 + 1^2$.
And, by applying Diophante's identities:

$$\begin{aligned} A = 2(5^2 + 14^2) &= (1^2 + 1^2)(5^2 + 14^2) &= 9^2 + 19^2 \\ = 2(11^2 + 10^2) &= (1^2 + 1^2)(11^2 + 10^2) &= 1^2 + 21^2 \quad . \end{aligned} \tag{23}$$

If we cannot write A as such a product, either because its factorization reveals a prime equal to 3 mod 4 with an odd exponent, or reveals only one prime equal to 1 modulo 4, or simply because we failed in factorizing $n - \delta$, then we have to try again with another value of n. This leads to the following forgery method.

Forgery.

Step 1: Try different moduli n until obtaining:

$$\begin{aligned} A &= x^2 + y^2 & x, y, z, t < \sqrt{n} \\ &= z^2 + t^2 & (x, y) \neq (z, t) \text{ and } (y, x) \neq (z, t) \quad . \end{aligned} \tag{24}$$

Step 2: Obtain the signature of 3 messages in the set $\{x, y, z, t\}$.
Step 3: Use the following relation to compute the signature of the remaining message:

$$\begin{aligned} \mu_5(x)\mu_5(z) &= (x^2 + \delta)(z^2 + \delta) \\ &= (A - y^2 + \delta)(A - t^2 + \delta) \\ &= (-y^2 - \delta)(-t^2 - \delta) \\ &= (y^2 + \delta)(t^2 + \delta) \\ &= \mu_5(y)\mu_5(t) \qquad (\text{mod } n) \quad . \end{aligned} \tag{25}$$

Example 4. $n = 493 = 17.29$ and $\delta = 272$.
Then $A = 9^2 + 19^2 = 1^2 + 21^2$ (see example 3).

$$\begin{aligned} \mu_5(9)\mu_5(1) &= (9^2 + 272)(1^2 + 272) \\ &= 234 \qquad (\text{mod } 493) \quad . \end{aligned} \tag{26}$$

And,

$$\begin{aligned} \mu_5(19)\mu_5(21) &= (19^2 + 272)(21^2 + 272) \\ &= 234 \qquad (\text{mod } 493) \quad . \end{aligned} \tag{27}$$

Remark 4. The attack can be extended to $A = 3n - 2\delta$, if $\delta > \frac{n}{2}$ (if not, A will be too large and some elements in the set $\{x, y, z, t\}$ will be greater than $\sqrt{n}$).

Example 5. We try our attack on the signature scheme where the RSA-modulus is the modulus specified in the Annex A of ISO 9796-1 [10]. All values in this example are in hexadecimal.

$$\begin{aligned} p = \quad & \texttt{BA09106C 754EB6FE BBC21479 9FF1B8DE} \\ & \texttt{1B4CBB7A 7A782B15 7C1BC152 90A1A3AB} \end{aligned} \tag{28}$$

$$\begin{aligned} q = \quad & \texttt{1 6046EB39 E03BEAB6 21D03C08 B8AE6B66} \\ & \texttt{CFF955B6 4B4F48B7 EE152A32 6BF8CB25} \end{aligned} \tag{29}$$

$$\begin{aligned} n = \quad & \texttt{1 00000000 00000000 00000000 00000000} \\ & \texttt{BBA2D15D BB303C8A 21C5EBBC BAE52B71} \\ & \texttt{25087920 DD7CDF35 8EA119FD 66FB0640} \\ & \texttt{12EC8CE6 92F0A0B8 E8321B04 1ACD40B7} \end{aligned} \tag{30}$$

Let δ a random constant of about the same size as the modulus n:

$$\begin{aligned} \delta = \quad & \texttt{FFE3B564 A0CB8C6C 6585C9CF A1CFC64B} \\ & \texttt{64B0C0F9 6CE980F5 ACC276C1 13045D1D} \\ & \texttt{05B1D218 D58C7D32 2387A305 9547EC31} \\ & \texttt{CF62CA5D 8C316E99 24B7F2C1 8A873FAE} \end{aligned} \tag{31}$$

Compute the factorization of A:

$$\begin{aligned} A &= 2(n - \delta) \\ &= \texttt{2.2F9.2F9D10D.} \\ &\quad \texttt{200000011}^2\texttt{.3FE9820B7AE6D}^5\texttt{.} \\ &\quad \texttt{3385F065A24DB4467E066FBBD577A0C6F6D119} \end{aligned} \tag{32}$$

With the Cornacchia algorithm and by applying the Diophante's identities we obtain 72 couples of values (a_i, b_i) such that $a_i^2 + b_i^2 = A$. And all these values are less than $\sqrt{n}$. We give 4 couples as examples:

$$\begin{aligned} a_1 = \quad & \texttt{10F26AC8 379A5197 8F6D6E3E 17461ED9} \\ & \texttt{1642DE79 C90D14D5 923190C6 D0A0EB} \\ b_1 = \quad & \texttt{78599149 C677F865 48F58E83 DA99C194} \\ & \texttt{9F653DBD FAEA8B8C 02BCDD8D 04F7F5B} \end{aligned} \tag{33}$$

$$\begin{aligned} a_2 = \quad & \texttt{15CCECF3 6BC80743 296A7F88 78FFC0E2} \\ & \texttt{D509B3C9 B1EA0B53 8FE5036E B23E93} \\ b_2 = \quad & \texttt{7858C944 CDCA3E18 0B0477F2 C6728C54} \\ & \texttt{BC4ADCD1 17361A46 2C0D7267 8661173} \end{aligned} \tag{34}$$

$$\begin{aligned} a_3 &= \texttt{274AEA5B 8289F65F 2C849CA7 DA69F691} \\ &\quad\ \texttt{15430C53 4EA3101F ACF6B8A8 673DDF} \\ b_3 &= \texttt{78545894 164142A8 FC5E800A 3DAC3705} \\ &\quad\ \texttt{BBAD4B7C 46AE5A24 1B4D5830 E9FC137} \end{aligned} \tag{35}$$

$$\begin{aligned} a_4 &= \texttt{4CE8CD96 B9920AB2 075E197C 564950E1} \\ &\quad\ \texttt{18BA416D 9FEC2BDF 5BE6BBEF C18F45} \\ b_4 &= \texttt{78422D6B ED414DAD 9BE47D08 F2CF8EF8} \\ &\quad\ \texttt{D742C8E5 C0440C45 F2B3300E B3E4A75} \end{aligned} \tag{36}$$

5 Conclusion

We have shown that all the countermeasures described in "ISO 9796 and the new forgery strategy (Working Draft)" [2] by Coppersmith, Halevi and Jutla can be attacked. For two propositions, we use previous forgeries presented at Eurocrypt '97 and Crypto '97. For the propositions two and three, μ_2 and μ_3, a recent attack is used. Moreover, we present two new ways to forge a signature when the last proposition is used.

Our contribution on the cryptanalysis of signature schemes with redundancy, after De Jonge-Chaum [11], Girault-Misarsky [5], Misarsky [12], Coron-Naccache-Stern [4] and Coppersmith-Halevi-Jutla [2] shows that is very difficult to define this kind of scheme. But, perhaps it is a good challenge for a year with a high level of redundancy (three zeroes) such as the year 2000.

6 Acknowledgements

We are grateful to Jean-Louis Nicolas for pointing out Cornacchia's algorithm to us and to Yves Hellegouarch for results picked out from his nice *Invitation aux mathématiques de Fermat-Wiles* [9].

References

1. Cohen, H.: A Course in Computational Algebraic Number Theory. Graduate Texts in Mathematics **138** Springer-Verlag (1995)
2. Coppersmith, D., Halevi, S., Jutla, C.: ISO 9796-1 and the new forgery strategy (Working Draft). Research Contribution to P1363 (1999) http://grouper.ieee.org/groups/1363/contrib.html
3. Cornacchia, G.: Su di un metodo per la risoluzione in numeri unteri dell' equazione $\sum_{h=0}^{n} C_h x^{n-h} y^h = P$. Giornale di Matematiche di Battaglini **46** (1908) 33–90
4. Coron, J.-S., Naccache, D., Stern, J. P.: On the Security of RSA Padding. Advances in Cryptology - Crypto '99 - Lecture Notes in Computer Science **1666** Springer-Verlag (1999) 1–18

5. Girault, M., Misarsky, J.-F.: Selective Forgery of RSA Signatures Using Redundancy. Advances in Cryptology - Eurocrypt '97 - Lecture Notes in Computer Science **1233** Springer-Verlag (1997) 495–507
6. Grieu, F.: A Chosen Messages Attack on ISO/IEC 9796-1 Signature Scheme. Advances in Cryptology - Eurocrypt 2000 - Lecture Notes in Computer Science **1807** Springer-Verlag (2000) 70–80 (this volume)
7. Gauss, C.F.: Disquisitiones Arithmeticae. Reissue Edition Springer-Verlag (1986)
8. Guillou, L.C., Quisquater, J.J., Walker, M., Landrock, P., Shaer, C.: Precautions Taken Against Various Potential Attacks in ISO/IEC DIS 9796, Digital Signature Scheme Giving Message Recovery. Advances in Cryptology - Eurocrypt '90 - Lecture Notes in Computer Science **473** Springer-Verlag (1991) 465–473
9. Hellegouarch, Y.: Invitation aux Mathématiques de Fermat-Wiles. Masson (1997)
10. ISO: ISO/IEC 9796-1, Information technology – Security techniques – Digital signature schemes giving message recovery – Part 1: Mechanisms using redundancy. (1991)
11. De Jonge, W., Chaum, D.: Attacks on Some RSA Signatures. Advances in Cryptology - Crypto '85 - Lecture Notes in Computer Science **218** Springer-Verlag (1986) 18–27
12. Misarsky, J.-F.: A Multiplicative Attack using LLL Algorithm on RSA Signatures with Redundancy. Advances in Cryptology - Crypto '97 - Lecture Notes in Computer Science **1294** Springer-Verlag (1997) 221–234
13. Misarsky, J.-F.: How (not) to Design RSA Signature Schemes. Public Key Cryptography - First International Workshop on Pratice and Theory in Public Key Cryptography - PKC'98 Lecture Notes in Computer Science **1431** Springer-Verlag (1998) 14–28
14. Morain, F.: Courbes elliptiques et tests de primalité. Thesis (1990)
15. Morain, F., Nicolas, J.-L.: On Cornacchia's algorithm for solving the diophantine equation $u^2 + dv^2 = m$. Chapter 2 of [14] ftp://ftp.inria.fr/INRIA/publication/Theses/TU-0144/ch2.ps

Security Analysis of the Gennaro-Halevi-Rabin Signature Scheme

Jean-Sébastien Coron
Ecole Normale Supérieure
45 rue d'Ulm
Paris, F-75230, France
coron@clipper.ens.fr

David Naccache
Gemplus Card International
34 rue Guynemer
Issy-les-Moulineaux, F-92447, France
david.naccache@gemplus.com

Abstract. We exhibit an attack against a signature scheme recently proposed by Gennaro, Halevi and Rabin [9]. The scheme's security is based on two assumptions namely the strong RSA assumption and the existence of a division-intractable hash-function. For the latter, the authors conjectured a security level exponential in the hash-function's digest size whereas our attack is sub-exponential with respect to the digest size. Moreover, since the new attack is optimal, the length of the hash function can now be rigorously fixed. In particular, to get a security level equivalent to 1024-bit RSA, one should use a digest size of approximately 1024 bits instead of the 512 bits suggested in [9].

1 Introduction

This paper analyses the security of a signature scheme presented by Gennaro, Halevi and Rabin at Eurocrypt'99 [9]. The concerned scheme (hereafter GHR) uses a standard (public) RSA modulus n and a random public base s. To sign a message m, the signer computes the e-th root modulo n of s with $e = H(m)$ where H is a hash function. A signature σ is verified with $\sigma^{H(m)} = s \bmod n$.

The scheme is proven to be existentially unforgeable under chosen message attacks under two assumptions : the strong RSA assumption and the existence of division-intractable hash-functions. The originality of the construction lies in the fact that security can be proven without using the random oracle model [3].

In this paper we focus on the second assumption, *i.e.* the existence of division-intractable hash-functions. Briefly, a hash function is division-intractable if it is computationally infeasible to exhibit a hash value that divides the product of other hash values. Assimilating the hash function to a random oracle, it is conjectured [9] based on numerical experiments that the number of k-bits digests needed to find one that divides the product of the others is approximately $2^{k/8}$. Here we show that the number of necessary hash-values is actually subexponential in k, namely $\exp((\sqrt{2\log 2}/2 + \circ(1))\sqrt{k \log k})$.

The paper is organised as follows. We briefly start by recalling the GHR scheme and its related security assumptions. Then we describe our attack, evaluate its asymptotical complexity and, by extrapolating from running times observed for small digest sizes, estimate the practical complexity of our attack. We also show that the attack is asymptotically optimal and estimate from a simple heuristic model the minimal complexity of finding a hash value that divides the product of the others.

2 The Gennaro-Halevi-Rabin Signature Scheme

2.1 Construction

The GHR scheme is a hash-and-sign scheme that shares some similarities with the standard RSA signature scheme :

Key generation : Generate a RSA modulus $n = p \cdot q$, product of two primes p and q of about the same length and a random element $s \in \mathbb{Z}_n^*$. The public key is (n, s) and the private key is (p, q).

Signature generation : To sign a message m, compute an odd exponent $e = H(m)$. The signature σ is :

$$\sigma = s^{e^{-1} \bmod \phi(n)} \bmod n$$

where $\phi(n) = (p-1)(q-1)$ is Euler's function.

Signature verification : Check that :

$$\sigma^{H(m)} = s \bmod n$$

2.2 GHR's Security Proof

The originality of the GHR signature scheme lies in the fact that its security can be proven without using the random oracle model. In the random oracle model, the hash function is seen as an oracle which outputs a random value for each new query. Instead, the hash function must satisfy some well defined computational assumptions [9]. In particular, it is assumed that the hash function family is division-intractable.

Definition 1 (Division intractability [9]). *A hashing family $\mathcal{H}$ is division intractable if finding $h \in \mathcal{H}$ and distinct inputs $X_1, \ldots X_n$,Y such that $h(Y)$ divides the product of the $h(X_i)$ values is computationally infeasible.*

The GHR signature scheme is proven to be existentially unforgeable under an adaptive chosen message attack, assuming the strong RSA conjecture.

Conjecture 1 (Strong-RSA [2]) *Given a randomly chosen RSA modulus n and a random $s \in \mathbb{Z}_n^*$, it is infeasible to find a pair (e, r) with $e > 1$ such that $r^e = s \bmod n$.*

An opponent willing to forge a signature without solving the strong-RSA problem can try to find messages $m, m_1, \ldots, m_r$ such that $H(m)$ divides the least common multiple of $H(m_1), \ldots, H(m_r)$. In this case, we say that a *division-collision* for H was exhibited. Using Euclid's algorithm the opponent can obtain $a_1, \ldots, a_r, k$ such that :

$$\frac{a_1}{H(m_1)} + \ldots + \frac{a_r}{H(m_r)} = \frac{1}{\mathrm{lcm}(H(m_1), \ldots, H(m_r))} = \frac{1}{k \cdot H(m)}$$

and forge the signature σ of m from the signatures σ_i of messages m_i by :

$$\sigma = \Big(\prod_{i=1}^{r} \sigma_i^{a_i}\Big)^k \bmod n$$

If $\mathcal{H}$ is division-intractable then it is infeasible for a polynomially bounded attacker to find a division collision for a hash function in $\mathcal{H}$. In particular, a random oracle is shown to be division-intractable in [9].

A natural question that arises is the complexity of finding a division collision, if one assumes that the hash function behaves as a random oracle, *i.e.* outputs a random integer for each new query. This question will condition the choice of the signature scheme's parameters. [9] conjectures (based on numerical experiments) a security level exponential in the length of the hash function, namely that the number of hash calls necessary to obtain a division-collision is asymptotically $2^{k/8}$ where k is the digest size. To get equivalent security to a 1024-bit RSA, [9] suggests to use 512-bit digests. In the next section, we exhibit a sub-exponential forgery and study its consequences for the recommanded digest size.

3 A Sub-exponential Attack

The outline of our attack is the following : we first look among many digests to find a smooth one, *i.e.* a hash value that factors into moderate-size primes p_i. Then for each of the p_i we look for a hash value divisible by p_i, so that the smooth hash value divides the least common multiple of the other hash values.

3.1 Background on Smooth Numbers

Let y be a positive integer. We say that an integer z is y-smooth if each prime dividing z is $\leq y$. An integer z is y-powersmooth if all primes powers dividing z are $\leq y$. Letting $\psi(x, y)$ denote the number of integers $1 \leq z \leq x$ such that z is y-smooth, the following theorem gives an estimate of the density of smooth numbers [5] :

Theorem 1. *If ϵ is an arbitrary positive constant, then uniformly for $x \geq 10$ and $y \geq (\log x)^{1+\epsilon}$,*

$$\psi(x,y) = xu^{-u+\circ(u)} \quad \text{as } x \to \infty$$

where $u = (\log x)/(\log y)$.

In particular, setting $y = L_x[\beta] = \exp\left((\beta + \circ(1))\sqrt{\log x \log\log x}\right)$, the probability that a random integer between one and x is $L_x[\beta]$-smooth is :

$$\frac{\psi(x,y)}{x} = L_x[-\frac{1}{2\beta}]$$

The proportion of squarefree integers is asymptotically $6/\pi^2$ [10]. Letting $\psi_1(x,y)$ denote the number of *squarefree* integers $1 \leq z \leq x$ such that z is y-smooth, theorem 3 in [10] implies that the same proportion holds for y-smooth numbers :

$$\psi_1(x,y) \sim \frac{6}{\pi^2}\psi(x,y) \tag{1}$$

under the growing condition :

$$\frac{\log y}{\log\log x} \to \infty, \quad (x \to \infty)$$

A squarefree y-smooth integer is y-powersmooth, so letting $\psi'(x,y)$ denote the number of integers $1 \leq z \leq x$ such that z is y-powersmooth, we have for all $x, y > 0$:

$$\psi_1(x,y) \leq \psi'(x,y) \leq \psi(x,y)$$

which using (1) shows that for $y = L_x[\beta]$, the probability that a random integer between one and x is y-powersmooth is :

$$\frac{\psi'(x,y)}{x} = L_x[-\frac{1}{2\beta}]$$

3.2 The Attack

In the following we assimilate the hash function to a random oracle which outputs random integers between one and x. Given a set $\mathcal{S}$ of random integers, we say that $(e, e_1, \ldots, e_r)$ is a *division-collision* for $\mathcal{S}$ if $e, e_1, \ldots, e_r \in \mathcal{S}$ and e divides the least common multiple of $e_1, \ldots, e_r$.

Theorem 2. *Let $\mathcal{S} = \{e_1, \ldots, e_v\}$ be a set of v random integers uniformly distributed between one and x. If $v = L_x[\sqrt{2}/2]$ then there exist a probabilistic Turing machine which outputs a division-collision for $\mathcal{S}$ in time $L_x[\sqrt{2}/2]$ with non-negligible probability.*

Proof: Using the following algorithm with $\beta = \sqrt{2}/2$, a division-collision is found in time $L_x[\sqrt{2}/2]$ with non-negligible probability.

An algorithm finding a division-collision :

Input : a set $\mathcal{S} = \{e_1, \ldots, e_v\}$ of $v = L_x[\sqrt{2}/2]$ random integers between one and x.

Output : a division-collision for $\mathcal{S}$.

Step 1 : Look for a powersmooth $e_k \in \mathcal{S}$ with respect to $y = L_x[\beta]$, using Pollard-Brent's Method [4] or Lenstra's Elliptic Curve Method (ECM) [11] to obtain :

$$e_k = \prod_{i=1}^{r} p_i^{\alpha_i} \quad \text{with} \quad {p_i}^{\alpha_i} \leq y \text{ for } 1 \leq i \leq r \tag{2}$$

Step 2 : For each prime factor p_i look for $e_{j_i} \in \mathcal{S}$ with $j_i \neq k$ such that $e_{j_i} = 0 \bmod p_i^{\alpha_i}$, whereby :

$$e_k \mid \operatorname{lcm}(e_{j_1}, \ldots, e_{j_r})$$

Pollard-Brent's method finds a factor p of n in $\mathcal{O}(\sqrt{p})$ expected running time, whereas the ECM extracts a factor p of n in $L_p[\sqrt{2}]$ expected running time. Using Pollard-Brent's method at step 1, an $L_x[\beta]$-powersmooth $H(m)$ is found in expected $L_x[1/(2\beta)] \cdot L_x[\beta/2] = L_x[1/(2\beta) + \beta/2]$ time. Using the ECM an $L_x[\beta]$-powersmooth $H(m)$ is found in $L_x[1/(2\beta)] \cdot L_x[\circ(1)] = L_x[1/(2\beta)]$ operations. Since $p_i^{\alpha_i} \leq y$, the second stage requires less than $y = L_x[\beta]$ operations.

The overall complexity of the algorithm is thus minimal for $\beta = 1$ when using Pollard-Brent's method, resulting in a time complexity of $L_x[1]$. The ECM's minimum complexity occurs for $\beta = \sqrt{2}/2$ giving a time complexity of $L_x[\sqrt{2}/2]$. □

Moreover, the following theorem shows that the previous algorithm is optimal.

Theorem 3. *Let $\mathcal{S} = \{e_1, \ldots, e_v\}$ be a set of v random integers uniformly distributed between one and x. If $v = L_x[\alpha]$ with $\alpha < \sqrt{2}/2$, then the probability that one integer in $\mathcal{S}$ divides the least common multiple of the others is negligible.*

Proof: See appendix A.

□

Consequently, assuming that the hash function behaves as a random oracle, the number of hash values necessary to exhibit a division-collision with non-negligible probability is asymptotically $L_x[\sqrt{2}/2]$ and this can be done in time $L_x[\sqrt{2}/2]$.

3.3 The Attack's Practical Running Time

Using the ECM, the attack has an expected time complexity of :

$$L_x[\sqrt{2}/2] = \exp\left((\frac{\sqrt{2}}{2} + \circ(1))\sqrt{\log x \log\log x}\right) \quad (3)$$

It appears difficult to give an accurate formula for the attack's practical running time since one would have to know the precise value of the term $\circ(1)$ in equation (3). However, extrapolating from (3) and the running times observed for small hash sizes, we can estimate the time complexity for larger hash sizes.

We have experimented the attack on a Pentium 200 MHz for hash sizes of 128, 160, and 192 bits, using the MIRACL library [12]. In Table 1 we summarize the observed running time in seconds and the logarithm in base 2 of the number of operations (assuming that the Pentium 200 MHz performs $200 \cdot 10^6$ operations per second).

Table 1. Experimental running times in seconds and $\log_2$ complexity (number of operations) of the attack for various digest sizes

digest size in bits	time complexity in seconds	$\log_2$ complexity
128	$3.5 \cdot 10^2$	36
160	$3.6 \cdot 10^3$	39
192	$2.1 \cdot 10^4$	42

Assuming that the complexity of the attack (number of operations) can be expressed as $C \cdot \exp(\sqrt{2}/2\sqrt{\log x \log\log x})$, the experimental complexity for a 192-bits hash size gives $C = 6.1 \cdot 10^4$, from which we derive in Table 2 the estimated complexity for larger hash sizes. The estimate may be rather imprecise and only provides an order of magnitude of the attack's complexity. However, the results summarized in Table 2 suggest that in order to reach a security level equivalent to 1024-bit RSA, digests should also be approximately 1024-bit long. Finally, we describe in the full version of the paper [6] a slightly better attack for the particular hash function suggested in [9].

4 Minimal Number of Hash Calls Necessary to Obtain a Division-collision

In the previous section we have estimated the time complexity of the attack using the ECM, from its asymptotic running time (3) and the observed running times for small hash sizes. Consequently, our estimate depends on the practical implementations of the hash function and the ECM. However theorem 3 shows that there is a lower bound on the number of hash calls necessary to mount the attack : asymptotically the number of hash calls must be at least $L_x[\sqrt{2}/2]$ so

Table 2. Estimated $\log_2$ complexity (number of operations) of the attack for various digest sizes

digest size	$\log_2$ complexity (number of operations)
256	47
512	62
640	69
768	75
1024	86

that with non-negligible probability there exist a division-collision (*i.e.* one hash value divides the least common multiple of the others). In this section we obtain heuristically a more precise estimate of the minimal number of hash calls necessary to have a division-collision with given probability. As in the previous section we assume that the hash function behaves as a random oracle, *i.e.* it outputs a random integer for each new query. Consequently the problem is the following : given a set $\mathcal{S}$ of v random integers in $\{1, \ldots, x\}$, what is the probability $P(x, v)$ that one integer in $\mathcal{S}$ divides the least common multiple of the others ?

4.1 A Heuristic Model

The probability $P(x, v)$ can be derived from a simple heuristic model called *random bisection*. In this model, the relative length of the first prime factor of a random number is obtained asymptotically by choosing a random λ uniformly in $[0, 1]$, and then proceeding recursively with a random integer of relative size $1 - \lambda$. This model is used in [1] to compute a recurrence for $F(\alpha) = \rho(1/\alpha)$, the asymptotic probability that all prime factors of a random x are smaller than x^α. In the above formula ρ is *Dickman's rho function* defined for real $t \geq 0$ by the relation [7] :

$$\rho(t) = \begin{cases} 1 & \text{if } \ 0 \leq t \leq 1 \\ \rho(n) - \displaystyle\int_n^t \frac{\rho(w-1)}{w} dw & \text{if } \ n \leq t \leq n+1 \text{ for } n \in \mathbb{N} \end{cases} \tag{4}$$

For an x^α-smooth integer x, the relative length λ chosen by random bisection is smaller than α, and the remaining integer of relative size $1 - \lambda$ is also x^α-smooth. Consequently, we obtain equation (5) from which we derive (4).

$$F(\alpha) = \int_0^\alpha F(\frac{\alpha}{1-\lambda}) d\lambda \tag{5}$$

Let $Q(x, v)$ denote the probability that a random integer z comprised between one and x divides the least common multiple of v other random integers in $\{1, \ldots, x\}$. Let $X = \log_2 x$ and $V = \log_2 v$. Let p be a prime factor of z of

relative size λ (*i.e.* $p = x^\lambda$). The probability that p divides a random integer in $\{1, \ldots, x\}$ is roughly $1/p$. Consequently, the probability P that p divides the least common multiple of v random integers in $\{1, \ldots, x\}$ is roughly :

$$P = 1 - (1 - \frac{1}{p})^v \simeq 1 - \exp(\frac{-v}{p}) \text{ for large } p$$

If $\lambda \leq V/X$, then $p \leq v$ and we take $P = 1$. Otherwise if $\lambda \geq V/X$ then $p \geq v$ and we take $P = v/p$. Consequently, we obtain :

$$Q(x,v) = \begin{cases} 1 & \text{if } x \leq v \\ \int_0^{\frac{V}{X}} Q(x^{1-\lambda}, v)d\lambda + \int_{\frac{V}{X}}^1 Q(x^{1-\lambda}, v)\frac{v}{x^\lambda} & \text{if } x > v \end{cases}$$

Letting $S(\alpha, V) = Q(v^\alpha, v)$, we have :

$$S(\alpha,V) = \begin{cases} 1 & \text{if } \alpha \leq 1 \\ \frac{1}{\alpha}\int_0^1 S(\alpha - s, V)ds + \frac{1}{\alpha}\int_1^\alpha S(\alpha - s, V)2^{V(1-s)}ds & \text{if } \alpha > 1 \end{cases}$$

We obtain :

$$\frac{\partial^2 S}{\partial \alpha^2}(\alpha, V) = -\frac{V \log 2}{\alpha} S(\alpha - 1, V) - (\frac{1}{\alpha} + V \log 2)\frac{\partial S}{\partial \alpha}(\alpha, V) \quad (6)$$

$S(\alpha, V)$ for $\alpha \geq 0$ is thus defined as the solution with continuous derivative of the delay differential equation (6) with initial condition $S(\alpha, V) = 1$ for $0 \leq \alpha \leq 1$.

A division-collision occurs if at least one integer divides the least common multiple of the others. We assume those events to be statistically independent. Consequently, we obtain :

$$P(x,v) \simeq 1 - \left(1 - S(\frac{X}{V}, V)\right)^v \quad (7)$$

4.2 Numerical Experiments

Table 3. Number of random integers required to obtain a division-collision with probability 1% as a function of their size (numerical experiments and heuristic model)

integer size	16	32	48	64	80	96
number of integers (experiments)	4	25	119	611	1673	7823
$\log_2$ number of integers (experiments)	2.0	4.6	6.9	9.3	10.7	12.9
$\log_2$ number of integers (model)	2.0	4.7	7.0	9.1	10.9	12.6

We performed numerical experiments to estimate the number of k-bit integers required so that a division-collision appears with good probability. We considered bit-lengths between $k = 16$ to $k = 96$ in increments of 16, and as in [9] for each bit length we performed 200 experiments in which we counted how many random integers were chosen until one divides the least common multiple of the others. As in [9], we took the second smallest result of the 200 experiments as an estimate of the number of integers required so that a division-collision appears with probability 1%. The results are summarized in Table 3.

Table 4. $\log_2$ number of random integers required to obtain a division-collision with probability 1% as a function of their size

integer size in bits	$\log_2$ number of integers
128	15.6
256	25.6
512	40.6
640	46.8
768	52.4
1024	63.2
1280	72.1

The function $S(\alpha, V)$ can be computed by numerical integration from (6) and $S(\alpha, V) = 1$ for $0 \leq \alpha \leq 1$. We used Runge-Kutta method of order 4 to solve the differential equation (6). We summarize in Table 3 the $\log_2$ number of k-bit integers required to obtain a division-collision with probability 1% for $k = 16$ to $k = 96$, from the heuristic model. We see that the values predicted by the model are close to the experimental values. In Table 4 we use the model to estimate the number of k-bit integers required to obtain a division-collision with probability 1% for large values of k. As in section 3.3 we see that in order to get a security level of a 1024-bits RSA, one should use a hash function of size approximately 1024 bits.

5 Conclusion

We have analysed the security of the Gennaro-Halevi-Rabin signature scheme of Eurocrypt'99. In particular, we exhibited a sub-exponential attack that forces to increase the security parameters beyond 512 or 642 bits up to approximately 1024 bits in order to get a security level equivalent to 1024-bits RSA. Another variant of the scheme described in [9] consists in generating prime digests only, by performing primality tests on the digests until a prime is obtained. In this case, a division-collision is equivalent to a collision in the hash function, but the signature scheme becomes less attractive from a computational standpoint.

References

1. E. Bach and R. Peralta, *Asymptotic semismoothness probabilities*, Mathematics of computation, vol. 65, no. 216, pp. 1701–1715, 1996.
2. N. Barić and B. Pfitzmann, *Collision-free accumulators and fail-stop signature scheme without trees*, proceedings of Eurocrypt'97, LNCS vol. 1233, Springer-Verlag, 1997, pp. 480-494.
3. M. Bellare and P. Rogaway, *Random oracles are practical : a paradigm for designing efficient protocols*. Proceedings of the First Annual Conference on Computer and Commmunications Security, ACM, 1993.
4. R. Brent, *An improved Monte Carlo factorization algorithm*, Nordisk Tidskrift för Informationsbehandling (BIT) 20 (1980) pp. 176-184.
5. E. Canfield, P. Erdös and C. Pomerance, *On a problem of Oppenheim concerning 'Factorisatio Numerorum'*, J. Number Theory, vol. 17, 1983, PP. 1-28.
6. J.S. Coron and D. Naccache, *Security analysis of the Gennaro-Halevi-Rabin signature scheme*, full version of this paper, available at `http://www.eleves.ens.fr:8080/home/coron`, 2000.
7. K. Dickman, *On the frequency of numbers containing prime factors of a certain relative magnitude*, Arkiv för matematik, astronomi och fysik, vol. 22A, no. 10, pp. 1–14, 1930.
8. G. Hardy and E. Wright, *An introduction to the theory of numbers*, Fifth edition, Oxford, 1979, pp. 354-359, 368-370.
9. R. Gennaro, S. Halevi and T. Rabin, *Secure hash-and-sign signatures without the random oracle*, proceedings of Eurocrypt'99, LNCS vol. 1592, Springer-Verlag, 1999, pp. 123-139.
10. A. Ivić and G. Tenenbaum, *Local densities over integers free of large prime factors*, Quart. J. Math. Oxford (2), 37 (1986), pp. 401-417.
11. H. W. Lenstra, Jr., *Factoring integers with elliptic curves*, Ann. of Math. (2) 126 (1987) pp. 649-673.
12. M.I.R.A.C.L. library, Shamus Software Ltd., 94 Shangan Road, Ballymun, Dublin, Ireland.

A Proof of Theorem 3

Proof: Let $\mathcal{S} = \{e_1, \ldots, e_v\}$ with $v = L_x[\alpha]$ and $\alpha < \sqrt{2}/2$ be a set of v random integers uniformly distributed between one and x. Denote by $P(x, v)$ the probability that one integer in $\mathcal{S}$ divides the least common multiple of the others and by B the event in which e_1 divides the least common multiple of $\{e_2, \ldots, e_v\}$. The proof's outline is the following : we consider the possible smoothness degrees of e_1 and compute the probability of B for each smoothness degree. Then we show that $\Pr[B]$ is smaller than $L_x[-\sqrt{2}/2 + \epsilon]$ for $\epsilon > 0$ and conclude that $P(x, v)$ is negligible.

The possible smoothness degrees of e_1 are denoted :

• Sm : e_1 is $L_x[\sqrt{2}/2]$-smooth. This happens with probability

$$\Pr[\mathrm{Sm}] = L_x[-\sqrt{2}/2]$$

and consequently :

$$\Pr[B \wedge \mathrm{Sm}] = \mathcal{O}(L_x[-\sqrt{2}/2]) \tag{8}$$

• $\mathrm{Sm}(\gamma, \epsilon)$: e_1 is $L_x[\gamma + \epsilon]$-smooth without being $L_x[\gamma]$ smooth, for $\sqrt{2}/2 < \gamma < \sqrt{2}$ and $\epsilon > 0$. This happens with probability :

$$\Pr[\mathrm{Sm}(\gamma, \epsilon)] = L_x[\frac{-1}{2 \cdot (\gamma + \epsilon)}] - L_x[\frac{-1}{2 \cdot \gamma}] = L_x[\frac{-1}{2 \cdot (\gamma + \epsilon)}] \tag{9}$$

In this case, e_1 contains a prime factor greater than $L_x[\gamma]$, which appears in the factorization of another e_i with probability $\mathcal{O}(L_x[-\gamma])$. Consequently e_1 divides the least common multiple of $\{e_2, \ldots, e_v\}$ with probability :

$$\Pr[B|\mathrm{Sm}(\gamma, \epsilon)] = \mathcal{O}(L_x[\alpha - \gamma])$$

With (9) and $\gamma + \frac{1}{2(\gamma+\epsilon)} \geq \sqrt{2} - \epsilon$ for all $\gamma > 0$, we get :

$$\Pr[B \wedge \mathrm{Sm}(\gamma, \epsilon)] = \mathcal{O}(L_x[-\frac{\sqrt{2}}{2} + \epsilon]) \tag{10}$$

• $\neg\mathrm{Sm}$: e_1 is not $L_x[\sqrt{2}]$-smooth. Consequently e_1 contains a factor greater than $L_x[\sqrt{2}]$ and thus :

$$\Pr[B \wedge \neg\mathrm{Sm}] = \mathcal{O}(L_x[\alpha - \sqrt{2}]) = \mathcal{O}(L_x[-\frac{\sqrt{2}}{2}]) \tag{11}$$

Partitioning the segment $[\sqrt{2}/2, \sqrt{2}]$ into segments $[\gamma, \gamma + \epsilon]$ and using equations (8), (10) and (11), we get :

$$\Pr[B] = \mathcal{O}(L_x[-\frac{\sqrt{2}}{2} + \epsilon])$$

Since $\alpha < \sqrt{2}/2$ we can choose $\epsilon > 0$ such that $\sqrt{2}/2 - \alpha - \epsilon = \delta > 0$ and obtain :

$$P(x, v) = \mathcal{O}(L_x[\alpha - \sqrt{2}/2 + \epsilon]) = \mathcal{O}(L_x[-\delta])$$

which shows that $P(x, v)$ is negligible. □

On the Security of 3GPP Networks

Michael Walker

Vodafone Airtouch and Royal Holloway College, University of London

(Extended Abstract)

Later this year we shall see the release of the Third Generation Partnership Project (3GPP) specifications for WCDMA – the first third generation standard for mobile communications. This 3G system combines elements of both a radical departure and a timid evolution from the 2G system known as GSM. It is radically different from GSM in having a wide-band CDMA system for its air-interface, but it hangs on to the GSM/GPRS core switching network with its MAP based signalling system. In this paper we consider the security features in WCDMA, taking a critical look at where they depart from those in GSM, where they are still very much the same and how they may develop as the core switching network is replaced by an IP based infrastructure.

Three principles underpinned the approach adopted for security in WCDMA: build on 2G by retaining security features from GSM that have proved to be needed and robust; address the weaknesses in 2G, both the real and the perceived ones; introduce new features where new 3G architectures and services demand them. In addition there was the desire to retain as much compatibility with GSM as possible in recognition of the fact that many WCDMA networks would be rolled out alongside GSM networks, with them sharing a core switching network, and with handover of calls between the two.

The problems with GSM security derive not so much from intrinsic problems with the mechanisms (although we will consider the algorithms separately) but rather from deliberate restrictions on the design. The most significant restriction was that GSM only needed to be as secure as the fixed networks. This was interpreted to mean that wherever fixed network technology was used cryptographic features were not needed. After all, they were not, and still are not, used by fixed carriers to protect consumer services. Fixed links in a mobile network were excluded from consideration, as was mobile signalling data when transferred over fixed networks. Protection against attacks involving impersonating a network element was not addressed. All this has led to three real security concerns for GSM: the use of false base stations to intercept mobile originated calls, interception of microwave links between base stations and the core network, and the vulnerability of signalling to interception and impersonation. We will consider each of these concerns and explain how they have been addressed in WCDMA.

The GSM algorithms were designed at a time when the political climate was very different from what it is today. It was radical to launch a public access

telecommunications system that automatically provided encryption – open evaluation and publication of the algorithm design criteria was just not an option. But the system was designed so that operators could use the best authentication algorithms available – so why was one used that is so obviously flawed? We look at these problems, and the rather different approach taken for the WCDMA algorithms.

All these considerations have led to the following set of security features in the first release of the WCDMA standard. Encryption of user traffic and signalling data on the air-interface, with the encryption terminated in the network at the RNC (radio network controller). This is further into the network than with GSM, where termination is at the base station. In addition to encryption, there is an integrity check on the air-interface signalling data. Authentication uses the same challenge-response technique as in GSM, except that it is enhanced to allow the mobile to verify the origin and freshness of the challenge. The basic key management is unchanged from GSM. The SIM still features as the security processor in the mobile terminal, and it shares an authentication key with its home network. This key is used to generate authentication data and encryption and integrity keys used to protect traffic in the access network. The security protocol is still executed in the local access network, but the network signalling is now protected. Thus user authentication data and ciphering keys can be encrypted when they are transferred between or within networks on signalling links.

The cryptographic keys for encryption and integrity are longer than those used in GSM, and a more open approach has been adopted for the design and evaluation of the air-interface algorithm. At the time of writing the algorithm has not been published, but it is hoped that it will be available on the ETSI web site shortly. As we shall see, the algorithm is very different from that used in GSM.

So for the first release of the WCDMA standards, the so-called release 99 or R99, the security features are more-or-less an upgraded version of those used in GSM. In particular, we still have a set of security features for an access network. This was to be expected, since the focus to date of 3GPP standardisation has been to define WCDMA as a new radio access to the GSM/GPRS switching network. The emphasis for R00 is now shifting to an IP based core network. We shall see that this is resulting in a set of additional security features.

One-Way Trapdoor Permutations Are Sufficient for Non-trivial Single-Server Private Information Retrieval

Eyal Kushilevitz[1] and Rafail Ostrovsky[2]

[1] IBM T.J. Watson Research Center, New-York and Computer Science Department, Technion, Haifa 32000, Israel.
eyalk@cs.technion.ac.il, eyalk@watson.ibm.com

[2] Telcordia Technologies Inc., 445 South Street, Morristown, New Jersey 07960-6438, USA
rafail@research.telcordia.com

Abstract. We show that general one-way trapdoor permutations are sufficient to privately retrieve an entry from a database of size n with total communication complexity strictly less than n. More specifically, we present a protocol in which the user sends $O(K^2)$ bits and the server sends $n - \frac{cn}{K}$ bits (for any constant c), where K is the security parameter of the trapdoor permutations. Thus, for sufficiently large databases (e.g., when $K = n^\epsilon$ for some small ϵ) our construction breaks the information-theoretic lower-bound (of at least n bits). This demonstrates the feasibility of basing single-server private information retrieval on general complexity assumptions.

An important implication of our result is that we can implement a 1-out-of-n Oblivious Transfer protocol with communication complexity strictly less than n based on any one-way trapdoor permutation.

1 Introduction

Private information retrieval (PIR, for short) is a communication protocol between a user and a server. In this protocol the user wishes to retrieve an item from a database stored in the server without revealing to the server which item is being retrieved. For concreteness, the database is viewed as an n-bit string x and the entry to be retrieved is the i-th bit of x. This problem was introduced by Chor et al. [9] and various aspects of it were further studied in [1, 8, 32, 27, 11, 15, 16, 28, 39, 12, 2, 7, 24, 30]. A naive solution for hiding which particular item is being retrieved (i.e., the index i) is to retrieve the entire database x. The communication complexity of this solution is n bits. Solutions that are more efficient than the naive one, in a setting where there are identical copies of the database stored in several servers, were found by [9] and later in [1, 24]. In this setting, the user can make queries to different servers and use the answers to reconstruct the bit x_i. Assuming that the servers do not communicate with each other, then privacy can be achieved with a cost which is much less than n (e.g., $O(n^{1/3})$ when

two such servers are available). Moreover, [9] have shown that if there is only a single server, then getting information-theoretic privacy with communication of less than n bits is impossible, hence motivating the use of replication.

Kushilevitz and Ostrovsky [27] have shown a way to get around this impossibility results. Namely they show that, assuming the hardness of some number-theoretic problem (specifically, the quadratic residuosity problem), it is possible to design a private information retrieval protocol with a single server and communication complexity of $O(n^\epsilon)$ (for any constant $\epsilon > 0$). [1] Their result strongly relies on the algebraic properties of the quadratic residuosity problem. Other single-server PIR protocols which are based on specific (number-theoretic and/or algebraic) intractability assumptions were subsequently presented in [28, 39, 7]. In particular, Cachin, Micali and Stadler [7] have shown that under the so-called ϕ-hiding (number-theoretic) assumption one can achieve even more efficient poly-logarithmic (in n) communication with a single server. (This is almost optimal since even without the privacy requirement the communication complexity must be at least $\log n$.) All these PIR protocols exploit specific algebraic structures related to the specific intractability assumption in use. In this paper, we address the question whether PIR protocols can be based on some "general" (preferably, the weakest possible) assumption.

Starting with the work of Yao [40], the program of identifying the weakest possible assumptions to reach various cryptographic tasks was launched. This program enjoyed a great success and for most cryptographic primitives we have very good grasp of both necessary and sufficient conditions; see, e.g. [21, 38, 36]. What about private information retrieval? On the lower-bound front, in addition to the information-theoretic lower-bound [9], recent work has established that single-server private information retrieval with less than n communication (even $n-1$ bits) already implies the existence of one-way functions [2] and, more generally, the existence of *Oblivious Transfer* (OT) protocols [12] (the connection between PIR and OT is discussed in more details below). The most general assumption based on which it is (currently) known how to construct OT is that one-way trapdoor permutations exist [20]. [2] Thus, in a sense, the most general assumption one can hope to use for constructing single-server private information retrieval protocols is the assumption that one-way trapdoor permutations exist (or trapdoor functions with polynomial pre-image size; see [3]).

[1] In [8] it is shown, in the setting where there are several servers storing identical database x, that intractability assumptions might be of help in constructing efficient PIR protocols.

[2] Impagliazzo and Rudich [23] have shown that OT is unlikely to be implemented based one one-way functions only (i.e. without trapdoor) since the proof of security (using black-box reductions) would yield a proof that P is not equal to NP. Also, Impagliazzo and Luby [22] have shown that oblivious transfer protocols already imply the existence of one-way functions. (In fact, OT was shown to be complete for any two-party computation [25, 26].) We also note that there are known constructions for OT which are based on *concrete* assumptions, such as the Diffie-Hellman assumption; in this case a trapdoor may not be required.

In this paper, we show that this is indeed feasible. That is, we show, under the sole assumption that one-way trapdoor permutations exist (without relying on special properties of any specific assumption), that single-server private information retrieval with strictly less than n communication is possible (or more precisely, of communication $n - \frac{cn}{K} + O(K^2)$, where $K \ll n$ is the security parameter and c is some constant[3]). We note however that, while the communication complexity is below the information-theoretic lower bounds of [9], it is nowhere close to what can be achieved based on specific assumptions. This quantitative question remains for future study.

As we already mentioned, single-server private information retrieval has a close connection to the notion of Oblivious Transfer (OT), introduced by Rabin [37]. A different variant of Oblivious Transfer, called 1-out-of-2 OT, was introduced in [13] and, more generally, 1-out-of-n OT was considered in [4].[4] All these notions were shown to be equivalent [5] and complete for all two party computations [25]. As mentioned, communication-efficient implementation of 1-out-of-n OT can be viewed as a single-server PIR protocol with an additional guarantee that only one (out of n) secrets is learned by the user. This notion (in the setting of several non-communicating servers) was first considered in [16] and called *Symmetric Private Information Retrieval (or SPIR)*. Kushilevitz and Ostrovsky [27] noted that in a setting of single-server PIR their protocol can be made into 1-out-of-n OT protocol (i.e., SPIR) with communication complexity $O(n^\epsilon)$ for any $\epsilon > 0$ (again, based on a specific algebraic assumption). Naor and Pinkas [30] have subsequently shown how to turn any PIR protocol into SPIR protocol with one invocation of PIR protocol and logarithmic number of invocations of 1-out-of-2 (string) OT. Combining our results with the results of [30] and with known implementations of OT based on any one-way trapdoor permutation [20], we get 1-out-of-n OT (i.e., SPIR) protocol based on any one-way trapdoor permutation whose communication complexity is strictly less than n.

Organization and Techniques: Section 2 includes some definitions that are used in this paper. In addition, it describes several tools from the literature that are used by our constructions. These include some facts about the Goldreich-Levin *hard-core predicates* [19], some properties of *universal one-way hash functions*, introduced by Naor and Yung [31], and properties of *interactive hashing* protocol, introduced by Ostrovsky, Venkatesan and Yung [33] [5]. In Section 3 we

[3] Further improvements are possible; see Section 3.2.

[4] Loosely speaking, 1-out-of-n OT is a protocol for 2 players: A *sender* who initially has n secrets $x_1, \ldots, x_n$ and a *receiver* who initially holds an index $1 \leq i \leq n$. At the end of the protocol the receiver knows x_i but has no information about the other secrets, while the sender has no information about the index i. Note that OT is different from PIR in that there is no communication complexity requirement (beyond being polynomially bounded) but, on the other hand, "secrecy" is required for *both* players.

[5] Interactive hashing has found many applications in cryptography (cf. [33, 29, 14, 34, 35, 18, 10, 6]) since, in some settings, it can replace collision-resistant hash-functions but it can be implemented from general cryptographic assumptions. The drawback

describe our basic PIR protocols based on one-way trapdoor permutations. This protocol is further extended in Section 4 to deal with faulty behavior by the server.

2 Preliminaries

2.1 Notation

We use the following notations throughout the paper. The data string is denoted by x, its length is denoted by n. The index of the bit that the user wishes to retrieve from this string is denoted by i. We use K to denote a security parameter.

For a finite set A, we denote by $a \in_R A$ the experiment of choosing an element of A according to the *uniform* distribution (and independently of all other random choices made).

2.2 Definitions

In this section we define the notions of *one-way trapdoor permutations* and of *hard-core predicates*. The reader is referred to [17] for an extended background related to these definitions.

Definition 1. *A collection of functions $\mathcal{G} = (\mathcal{G}_K)$ is called a collections of one-way trapdoor permutations if the following hold:*

- *There exists a probabilistic polynomial-time generating algorithm, I, that on input 1^K outputs a pair (g, g^{-1}) where g is (an index of) a function in $\mathcal{G}_K$ and g^{-1} is a string called the "trapdoor for g".*
- *Each function $g \in \mathcal{G}_K$ is a permutation over $\{0,1\}^K$ and is computable in polynomial time (that is, there exists an algorithm that given $g \in \mathcal{G}$, and $x \in \{0,1\}^*$ computes the value of $g(x)$ in time polynomial in $|x|$).*
- *Each g is easy to invert given its trapdoor g^{-1}. That is, there exists an algorithm that given $y \in \{0,1\}^K$ and the string g^{-1} computes the (unique) value x such that $g(x) = y$ (i.e. $x = g^{-1}(y)$) in time polynomial in K.*
- *It is hard to invert the functions in $\mathcal{G}$ without having the trapdoor. Formally, for every probabilistic polynomial-time algorithm B, every integer c, and sufficiently large K*

$$\Pr_{g \in I_{\mathcal{G}}(1^K), y \in_R \{0,1\}^K}(B(g,y) = g^{-1}(y)) < \frac{1}{K^c} \ ,$$

where "$g \in I_{\mathcal{G}}(1^K)$" denotes choosing a function g according to the probability distribution induced by the generating algorithm I.

---

of this primitive is its high round-complexity (our protocol for a malicious server inherits this drawback; the question of how to reduce the round-complexity of this protocol is an interesting open problem).

Remark: There are definitions of one-way trapdoor permutations that give more power to the adversary. For example, the adversary may *adaptively* ask for many inverses of his choosing and only then try to invert the given permutation on a randomly chosen point. Another strengthening of the adversary, which is of interest in some cases, is requiring that it can recognize if g is "well-formed". The way in which we use the trapdoor permutations in our protocols, none of these issues come up and so we stick to the above simpler definition.

Next, we will need the notion of hard-core predicates. Specifically, we will use the Goldreich-Levin hard-core predicates [19]. For a string $r \in \{0,1\}^K$ let us denote $r(x) = \langle r, x \rangle$, where $\langle \cdot, \cdot \rangle$ is the standard inner-product modulo 2. The Goldreich-Levin Theorem [19] states that if g is a one-way permutation then there is no algorithm that can compute $r(x)$ given $g(x)$ and r. Formally, for every probabilistic polynomial-time algorithm B, every integer c, and sufficiently large K

$$\Pr_{g \in I_{\mathcal{G}}(1^K), x \in_R \{0,1\}^K, r \in_R \{0,1\}^K} (B(g(x), r) = r(x)) \quad < \quad \frac{1}{2} + \frac{1}{K^c} \ .$$

Remark: the above definitions concentrate on the case of one-way permutations; however, they can be easily generalized to deal with more general notions. In particular, the Goldreich-Levin Theorem [19] applies to any one-way function.

2.3 Some Useful Machinery

Let $\mathcal{G}$ be some arbitrary family of one-way trapdoor permutations over $\{0,1\}^K$. It is sometimes convenient to view strings in $\{0,1\}^K$ as elements of the field $\mathbf{GF}[2^K]$. With this view in mind, let

$$\mathcal{H} = \left\{ h_{a,b} : \mathbf{GF}[2^K] \to \mathbf{GF}[2^K] \ \mid \ h(x) = ax + b, \quad a, b \in \mathbf{GF}[2^K], \quad a \neq 0 \right\}.$$

Given $\mathcal{G}$ and $\mathcal{H}$, Naor and Yung [31] define the following family of functions

$$\mathcal{F} = \left\{ f : \{0,1\}^K \to \{0,1\}^{K-1} \ \mid \ g \in \mathcal{G}, h \in \mathcal{H}, f(x) = \mathrm{chop}(h(g(x))) \right\}$$

where the chop operator takes a string and chops its last bit.
For a function $f \in \mathcal{F}$ we sometimes denote $f = (g, h)$ to indicate the functions $g \in \mathcal{G}, h \in \mathcal{H}$ based on which f is defined. Moreover, if I is the generating algorithm for $\mathcal{G}$ then we denote by $I_{\mathcal{F}}$ a generating algorithm for $\mathcal{F}$ that generates (g, g^{-1}) by applying I, generates $h \in \mathcal{H}$ according to the uniform distribution and let $f = (g, h)$.
The following are basic properties of $\mathcal{F}$.

1. Each function $f \in \mathcal{F}$ is $2 \to 1$. In other words, for every $x \in \{0,1\}^K$ there is a (unique) string, denoted $x^\star$, such that $f(x^\star) = f(x)$ and $x^\star \neq x$.

2. Every function $f = (g, h)$ in $\mathcal{F}$ is efficiently computable. Moreover, given the trapdoor g^{-1} it is easy to compute, for every $y \in \{0,1\}^{K-1}$ the two strings $x, x^\star$ such that $f(x) = f(x^\star) = y$. [6]
3. Collisions are hard to find for $\mathcal{F}$ [31] (i.e., given x and $f(x)$ it is hard to find the string $x^\star$). Formally, for every x, for every probabilistic polynomial-time algorithm B, every integer c, and sufficiently large K

$$\Pr_{f=(g,h)\in I_{\mathcal{F}}(1^K)} \left(B(x, f(x)) = x^\star\right) \quad < \quad \frac{1}{K^c} \ .$$

Note that property 3 does not guarantee that specific bits of $x^\star$ are hard to find. Instead we will make use of *hard-core* bits.

We shall use in an essential way an *interactive hashing* protocol of Ostrovsky, Venkatesan and Yung [33]. Interactive hashing found many applications in cryptography (cf. [33, 14, 29, 34, 35, 18, 10, 6]). This is a protocol between two players Alice and Bob, where both Alice and Bob are probabilistic polynomial-time machines. Alice is given as an input 1^K, a function $g \in \mathcal{G}_K$ and an input $x \in \{0,1\}^K$; Bob is given 1^K. The interactive hashing protocol proceeds as follows:

- Bob chooses uniformly at random $K-1$ vectors $H_1, \ldots, H_{K-1}$ in $\{0,1\}^K$ subject to the constraint that these $K-1$ vectors are linearly independent (viewing them as elements of the linear space Z_2^K).
- The players interact in $K-1$ rounds where in round i they do the following:
  - Bob sends to Alice H_t
  - Alice sends to Bob $\langle H_t, g(x)\rangle$ (the inner product of H_t and $g(x)$).

The communication in this protocol, consisting of the strings $H_1, \ldots, H_{K-1}$ sent by Bob and the bits $\langle H_1, g(x)\rangle, \ldots, \langle H_{K-1}, g(x)\rangle$, define $K-1$ linear equations and since all the H_t's are linearly independent these equations admit two solutions, denoted $\{y, y^\star\}$ (we use the same notation as was used above for the pre-images of $f \in \mathcal{F}$ to stress the analogy between these two tools; this analogy will also be used in our protocols). We now state several facts regarding interactive hashing that make it useful for our purposes:

- If Alice follows the protocol then one of $\{y, y^\star\}$ is $g(x)$ (recall that x is an input to Alice).
- Bob sends total of $O(K^2)$ bits to Alice. Alice sends total of $K-1$ bits in response.
- It is hard for Alice to find inverses of both $y, y^\star$, even if Alice does not follow the protocol. Formally, for every probabilistic polynomial-time algorithm A', for every integer c and sufficiently large K, if g is chosen according to $I_{\mathcal{G}}(1^K)$ then after A' executes the protocol with Bob, the probability that A' outputs x_0, x_1 such that both $g(x_0) = y$ and $g(x_1) = y^\star$ is less than $\frac{1}{K^c}$.

[6] Note that every $h \in \mathcal{H}$ is $1 \to 1$ and easy to invert; therefore, given y one can try the two options for the chopped bit, invert h and then invert g using the trapdoor. We also note that this property was not considered in [31] since they deal with arbitrary one-way permutations and not only with *trapdoors* permutations.

Interactive hashing, as described up-to this point, works with any one-way permutation. In [33] one more property was used, which is needed in the current paper as well. Specifically, we will apply interactive hashing with one-way *trapdoor* permutations; this modifications gives the following crucial property:

- Given the trapdoor for g (i.e., the string g^{-1}) and the communication (i.e., the strings $H_1, \ldots, H_{K-1}$ and the bits $\langle H_1, g(x)\rangle, \ldots, \langle H_{K-1}, g(x)\rangle$) Bob can compute both x_0 and x_1 (i.e., the strings such that $g(x_0) = y$ and $g(x_1) = y^\star$).

2.4 PIR Protocols

A *Private Information Retrieval* (PIR) is a protocol for two players: a server $\mathcal{S}$ who knows an n-bit string x (called the *database*), and a user $\mathcal{U}$ holding an index $i \in [n]$ and interested in retrieving the value x_i. When considering the privacy requirement of PIR protocols there are several possible types of "faulty" behaviors by the server: the server might be *honest-but-curious* or it might be *malicious*. Below we detail the definition for each of these types; we note however that the difference is especially important when dealing with multi-round protocols (as those described in this work).

An honest-but-curious server is a one that behaves according to the predefined protocol and just tries to deduce information about i from the communication it sees. This is formulated as follows: Fix a data string x; for every $i, i' \in [n]$ (where $i \neq i'$) the distribution of communications generated by the protocol when the user is interested in bit i is indistinguishable from the distribution generated when the user is interested in index i'.[7] We stress here that x is fixed and the server is not allowed to change it during the protocol's execution.

A malicious server is a one that does not necessarily follow the protocol. It should be immediately noticed that there are several "bad" behaviors by a malicious server which cannot be avoided; e.g., the server may refuse to participate in the protocol or it may change the content of the database (say, it can act as if $x = 0^n$). The privacy requirement in this case makes sure however that, no matter what the server does, the identity of the index i is not revealed. Formally, for every $i, i' \in [n]$ (where $i \neq i'$) no probabilistic polynomial-time server $\mathcal{S}'$ can distinguish executions of the protocol when the user's index is i from executions of the protocol when the user's index is i'. We stress that here, the server is allowed to modify its messages in an arbitrary manner during the protocol execution in order to be able to distinguish.

3 A PIR Protocol with respect to a Honest-but-Curious Server

In this section we present the honest-but-curious PIR protocol which proves that it is possible to construct a PIR protocol from any family of one-way trap-

[7] For lack of space we omit the formal definition of *indistinguishability* which is a standard one [40].

door permutations, with communication complexity smaller than n. (Later we describe some simple improvements on this protocol.)

Theorem 1. *If one-way trapdoor permutations exist then there exists honest-but-curious single-server PIR protocol whose communication complexity is at most*

$$n - \frac{n}{2K} + O(K).$$

(More precisely, the user sends $O(K)$ bits and the server sends at most $n - \frac{n}{2K}$ bits.)

(Some slightly better bounds are mentioned in Section 3.2 below).

Let $\mathcal{G}$ be a collection of one-way trapdoor permutations, as guaranteed by the theorem, and let $\mathcal{F}$ be a family of $2 \to 1$ functions constructed based on $\mathcal{G}$, as described in Section 2.3. Assume, without loss of generality, that n is divisible by $2K$ and let $\ell = n/(2K)$. The protocol works as follows.

1. The user picks two functions $f_L = (g_L, h_L)$ and $f_R = (g_R, h_R)$ (including the corresponding trapdoors g_L^{-1} and g_R^{-1}) using the generating algorithm $I_{\mathcal{F}}(1^K)$. It sends the functions f_L, f_R to the server (without the trapdoors).
2. Both the server and the user view x as if it is composed of 2ℓ sub-strings $z_{1,L}, z_{1,R}, z_{2,L}, z_{2,R}, \ldots, z_{\ell,L}, z_{\ell,R}$ each of size K (we refer to these strings as "blocks"). The server now applies f_L to each block $z_{j,L}$ and applies f_R to each block $z_{j,R}$. It sends all the outcomes

$$\begin{array}{cc} f_L(z_{1,L}) & f_R(z_{1,R}) \\ f_L(z_{2,L}) & f_R(z_{2,R}) \\ \vdots & \vdots \\ f_L(z_{\ell,L}) & f_R(z_{\ell,R}) \end{array}$$

   to the user.
3. The user, having the trapdoors for both f_L and f_R, can compute for each block z the two possible pre-images $\{z, z^\star\}$. Assume that the bit x_i is in some block $z_{s,L}$, for some s. The user picks random $r_L, r_R \in \{0,1\}^K$ such that the hard-core predicates corresponding to r_L, r_R satisfy

$$r_L(z_{s,L}) \neq r_L(z^\star_{s,L}) \quad \text{and} \quad r_R(z_{s,R}) = r_R(z^\star_{s,R}).$$

   It sends r_L, r_R to the server. (If the index x_i is in block $z_{s,R}$ then r_L, r_R are chosen subject to the constraint $r_R(z_{s,R}) \neq r_R(z^\star_{s,R})$ and $r_L(z_{s,L}) = r_L(z^\star_{s,L})$.)
4. For each $j = 1, \ldots, \ell$ the server computes and sends the bit $b_j = r_L(z_{j,L}) \oplus r_R(z_{j,R})$.
5. By the choice of r_L, r_R the bit b_s allows the user to compute the value of $z_{s,L}$ (or the value of $z_{s,R}$ depending on the way that r_L, r_R were chosen).[8] This gives the user the bit x_i (as well as all other bits in the corresponding block).

[8] The user ignores all the other bits b_j, for $j \neq s$.

Correctness: The correctness follows from the description of the protocol and the basic properties of $\mathcal{F}$. The idea is that for the pair of blocks in which the user is interested, $z_{s,L}, z_{s,R}$, the hard-core predicates are chosen in a way that they are sensitive on the block which the user wishes to retrieve, and are constant on the other block. This allows the user to distinguish the target z from $z^\star$.

Communication complexity: The only messages sent by the user are those for specifying f_L, f_R, r_L, r_R; all together $O(K)$ bits. The server, on the other hand, sends for each pair of blocks $2(K-1)$ bits in Step 2 and an additional bit in Step 4. All together, $\ell \cdot (2K-1) = n - \frac{n}{2K}$ bits. Therefore, the communication complexity is as claimed by the theorem.

3.1 Proof of Security

The only information that the user sends which depends on the index it is interested in is the choice of r_L, r_R (Step 3). We need to show that these strings maintain the privacy of the user's index. For this we introduce some notation. We say that a block $z_{s,L}$ (resp. $z_{s,R}$) is of type "E" (equal) if $r_L(z_{s,L}) = r_L(z^\star_{s,L})$ (resp., if $r_R(z_{s,R}) = r_R(z^\star_{s,R})$); similarly, we say that a block $z_{s,L}$ (resp. $z_{s,R}$) is of type "N" (not equal) if $r_L(z_{s,L}) \neq r_L(z^\star_{s,L})$ (resp., if $r_R(z_{s,R}) \neq r_R(z^\star_{s,R})$). Hence, the choice of r_L, r_R defines a sequence of ℓ pairs in $\{E,N\}^2$ with the only restriction being that the pair in which the index i resides must be either (N,E) or (E,N) (depending on whether i is in the left block or the right block). We also use $\star$ to denote a "don't-care". So if, for example, the user wishes to retrieve the first block it picks r_L, r_R subject to the constraint that the corresponding sequence is $(N,E), (\star,\star), \ldots, (\star,\star)$.

Using the above notation, we will now prove that the server cannot distinguish any pair of indices i, i' the user may wish to retrieve. Obviously, if i, i' are in the same block then the user behaves in an identical way in both cases and there is no way for the server to distinguish the two cases. The next case is where i, i' are in the same pair of blocks; say, i is in $z_{s,L}$ and i' in $z_{s,R}$. For simplicity of notations assume $s = 1$ then in the first case r_L, r_R are chosen uniformly from those that induce the sequence

$$(N,E), (\star,\star), \ldots, (\star,\star)$$

while in the second case r_L, r_R are chosen from those that induce the sequence

$$(E,N), (\star,\star), \ldots, (\star,\star).$$

We omit the details for this case since it is a degenerate case of the more general scenario where, say, i is in $z_{s,L}$ and i' in $z_{s',R}$. Again, for simplicity of notations assume $s = 1, s' = 2$; then, we have to distinguish the following two sequences:

$$(N,E), (\star,\star), (\star,\star), \ldots, (\star,\star)$$

and

$$(\star,\star), (E,N), (\star,\star), \ldots, (\star,\star).$$

(Note that if, for example, the server can tell that for some s the corresponding pair is of type, say, (E,E) then it can conclude that none of the blocks $z_{s,L}, z_{s,R}$ is of interest for the user.) We now show that if the server is able to distinguish the above two sequences it can also predict the hard-core predicate associated with the family $\mathcal{G}$.

The first step uses a hybrid argument to claim that if one can distinguish the two distribution of r_L, r_R as above (given x, f_L and f_R) then it can also distinguish two adjacent distributions among the following list of distributions:

$$\begin{aligned}
\Pi_1 &: \quad (N,E),(\star,\star),(\star,\star),\ldots,(\star,\star)\\
\Pi_2 &: \quad (\star,E),(\star,\star),(\star,\star),\ldots,(\star,\star)\\
\Pi_3 &: \quad (\star,\star),(\star,\star),(\star,\star),\ldots,(\star,\star)\\
\Pi_4 &: \quad (\star,\star),(E,\star),(\star,\star),\ldots,(\star,\star)\\
\Pi_5 &: \quad (\star,\star),(E,N),(\star,\star),\ldots,(\star,\star)
\end{aligned}$$

(If each pair of adjacent distributions is indistinguishable then so are Π_1 and Π_5, contradicting the assumption that the server can distinguish.) Suppose, for example, that one can distinguish Π_1 and Π_2 (other cases are similar or even simpler; they might require flipping the roles of f_L and f_R). Then, it is also possible to distinguish Π_1 and

$$\Pi_2' : \quad (E,E),(\star,\star),(\star,\star),\ldots,(\star,\star).$$

To make the distinguishing property more concrete assume, without loss of generality, that for some data string x,

$$Pr_{f_L,f_R\in I_{\mathcal{F}}(1^K),(r_L,r_R)\in\Pi_1}(D(x,f_L,f_R,r_L,r_R)=1) \quad \leq \quad \frac{1}{2}-\epsilon$$

and

$$Pr_{f_L,f_R\in I_{\mathcal{F}}(1^K),(r_L,r_R)\in\Pi_2'}(D(x,f_L,f_R,r_L,r_R)=1) \quad \geq \quad \frac{1}{2}+\epsilon.$$

We use this algorithm D to construct an algorithm B that on input $g \in I_{\mathcal{G}}(1^K)$, $y \in_R \{0,1\}^K$ and $r \in_R \{0,1\}^K$ predicts the hard-core predicate $r(g^{-1}(y))$, with probability $0.5+\epsilon$. This contradicts the Goldreich-Levin Theorem [19] (See Section 2.2). Algorithm B works as follows:

1. Choose h_L at random subject to the constraint

   $$\mathrm{chop}(h_L(y)) = \mathrm{chop}(h_L(g(z_{1,L}))).^9$$

   Let $f_L = (g, h_L)$ and $r_L = r$. (Note that, with respect to f_L we have $z^\star_{1,L} = g^{-1}(y)$. Also crucial is the fact that since D does not have y (only B does) the distribution of h_L looks random to D).

[9] Specifically, in the unlikely event that $g(z_{1,L}) = y$ we are done; otherwise, choose $v \in \{0,1\}^K$ at random and let v' be identical to v with the last bit flipped. Then, we solve the system of equations $a \cdot y + b = v$ and $a \cdot g(z_{1,L}) + b = v'$ to find a, b (i.e., h_L). In particular $a = (v - v')/(y - g(z_{1,L}))$ (note that this is well defined since $y \neq g(z_{1,L})$ and different than 0 since $v \neq v'$).

2. Choose a function $f_R \in I_{\mathcal{F}}(1^K)$ (including the corresponding trapdoor!) and compute the string $z^\star_{1,R}$ (by using the trapdoor). Pick a random r_R subject to the constraint that $r_R(z^\star_{1,R}) = r_R(z_{1,R})$.
3. Invoke D on input (x, f_L, f_R, r_L, r_R). If the output is "1" (in which case the input is more likely to be from Π'_2; i.e., $r_L(z_{1,L})$ and $r_L(z^\star_{1,L})$ are more likely to be not-equal) then B's output is $1 - r_L(z_{1,L})$. If the output is "0" (in which case the input is more likely to be from Π_1; i.e., $r_L(z_{1,L})$ and $r_L(z^\star_{1,L})$ are more likely to be equal) then B's output is $r_L(z_{1,L})$. (Note that while B does not know what $z^\star_{1,L}$ is, it knows $z_{1,L}$ and hence can apply r_L to it.

It can be verified that the distribution of inputs provided to D is exactly what is needed and hence the correctness of B follows.

3.2 Some Improvements

We tried to make the description of the protocol above as simple as possible. There are however certain modifications that one can apply to it in order to slightly improve the efficiency. One such improvement is instead of using two functions f_L, f_R to use d such functions $f_1, \ldots, f_d$ (where d may depend on K and/or n). Then, the user can choose hard-core predicates $r_1, \ldots, r_d$ such that the one corresponding to the index i gets two different values (on the corresponding $z, z^\star$) while each of the other hard-core predicates get the same value (on $z, z^\star$). Then, when the server returns the exclusive-or of the d bits this allows the user to reconstruct the block of interest.

A second (more significant) modification that one can make is, instead of using $\mathcal{F}$ as above, where each $f \in \mathcal{F}$ is obtained by chopping a single bit from $h(g(x))$, we can chop some s bits (specifically, $s = O(\log \log n)$). Now, in Step 2 of the protocol the server needs to send only $K - s$ bits per block. In Step 3 the user can pick s strings r's that will allow him to retrieve only the block of interest. Finally, in Step 4 (if combined with the previous modification) for each d blocks it needs to send back s bits. This gives a complexity of $n - \frac{(d-1)cn \log \log n}{dK}$ bits from the server to the user (for any constant c) and $O(Kd \log \log n)$ bits from the user to the server.

4 A PIR Protocol with respect to a Malicious Server

In this section we deal with the case where the server is malicious. It is instructive to consider first the protocol of Section 3 and examine the possibilities of a malicious server to violate the privacy of the protocol. Suppose that the server *after* receiving the functions f_L, f_R from the user (in Step 1) can find a pair of strings $\alpha_1, \alpha_2 \in \{0,1\}^K$ such that $f_L(\alpha_1) = f_L(\alpha_2)$ (note that the properties of $\mathcal{F}$ guarantee that for every x and a randomly chosen $f \in \mathcal{F}$ it is hard to find $x^\star$; but it does *not* guarantee that after choosing f one cannot find a pair $x, x^\star$ with respect to this f; this is exactly the weakness that we wish to use). Then, the server can replace say $z_{1,L}$ by α_1. Now, when getting r_L, r_R from the user (in

Step 3) it can tell whether the first block is of type "E" or "N" (since it knows both $z_{1,L}$ and $z^{\star}_{1,L}$ which are just α_1 and α_2). So, for example, if the block is of type "E" then it follows that i is not in the first block. This violates the privacy of i.

To overcome the above difficulties, we replace the use of the family $\mathcal{F}$ by the use of interactive hashing. While the two tools have several similarities, interactive hashing is the right tool to make sure that the server cannot, for example, force both α_1 and α_2 to be mapped in the same way. However, there is another technical difficulty in generalizing the honest-but-curious case to the malicious case. Consider the proof of security in Section 3.1. A crucial point in that proof is that we can make $z_{1,L}$ (which is fixed) and $g^{-1}(y)$ be mapped to the same value. In the malicious case this cannot be done because the server need not fix the database and may choose it in some arbitrary way (possibly depending on the communication). Intuitively, this means that the fact that the distinguisher can tell blocks of type "E" (equal) from blocks of type "N" (not equal) does not necessarily help us in predicting the hard-core bit. This will require us to come up with some extra new machinery (see the definition of $\hat{\mathcal{G}}$ below).
We prove the following theorem:

Theorem 2. *If one-way trapdoor permutations exist then there exists malicious single-server PIR protocol whose communication complexity is at most*

$$n - \frac{n}{6K} + O(K^2).$$

(More precisely, the user sends $O(K^2)$ bits and the server sends at most $n - \frac{n}{6K}$ bits. Also, if the server is honest then with a negligible probability the protocol fails; i.e., the user does not get the bit x_i but its privacy is still maintained.[10]*)*

Let $\mathcal{G}$ be a collection of one-way trapdoor permutations, as guaranteed by the theorem. As a first step we construct, based on $\mathcal{G}$, a new family of one-way trapdoor permutations $\hat{\mathcal{G}}$ which is defined as follows. Each function $\hat{g} \in \hat{\mathcal{G}}_K$ is defined using 4 functions $g_{00}, g_{01}, g_{10}, g_{11} \in \mathcal{G}_{K-2}$. Let x be a string in $\{0,1\}^K$ and write $x = b_1 b_2 w$, where $b_1, b_2 \in \{0,1\}$ and $w \in \{0,1\}^{K-2}$. We define

$$\hat{g}(x) = b_1 b_2 g_{b_1 b_2}(w).$$

Clearly each such $\hat{g}$ is a permutation over $\{0,1\}^K$. The trapdoor $\hat{g}^{-1}$ corresponding to $\hat{g}$ consists of the corresponding 4 trapdoors; i.e., $(g_{00}^{-1}, g_{01}^{-1}, g_{10}^{-1}, g_{11}^{-1})$. The generating algorithm for $\hat{\mathcal{G}}$, denoted $I_{\hat{\mathcal{G}}}(1^K)$ simply works by applying $I_{\mathcal{G}}(1^{K-2})$ four times for generating $g_{00}, g_{01}, g_{10}, g_{11}$ (with their trapdoors).

As before assume, without loss of generality, that n is divisible by $2K$ and let $\ell = n/(2K)$. The protocol works as follows.

[10] As pointed out in Section 2.4, a "bad" server can always refuse to let the user retrieve the bit; hence, this is not considered a violation of the correctness requirement.

1. The user picks two functions $\hat{g}_L$ and $\hat{g}_R$ (including the corresponding trapdoors $\hat{g}_L^{-1}$ and $\hat{g}_R^{-1}$) using the generating algorithm $I_{\hat{\mathcal{G}}}(1^K)$. It sends the functions $\hat{g}_L, \hat{g}_R$ to the server (without the trapdoors).
2. As before the server and the user view the string x as if it is composed of 2ℓ "blocks" $z_{1,L}, z_{1,R}, z_{2,L}, z_{2,R}, \ldots, z_{\ell,L}, z_{\ell,R}$ each of size K.
   Now the server and the user play 2ℓ interactive hashing protocols as follows. First, the user chooses $K-1$ linearly independent vectors in $\{0,1\}^K$ denoted $(H_1^L, \ldots, H_{K-1}^L)$. Now, for each t from 1 to $K-1$ (in rounds) do:
   - The user sends to the server H_t^L.
   - The server sends to the user the bits $\langle H_t^L, \hat{g}_L(z_{1,L})\rangle, \ldots, \langle H_t^L, \hat{g}_L(z_{\ell,L})\rangle$.

   The same is repeated for the "right" blocks. That is, the user chooses another set of $K-1$ linearly independent vectors $H_1^R, \ldots, H_{K-1}^R$ and (in rounds) get from the server the values $\langle H_t^R, \hat{g}_R(z_{1,R})\rangle, \ldots, \langle H_t^L, \hat{g}_R(z_{\ell,R})\rangle$.
3. The user, having the trapdoors for both $\hat{g}_L$ and $\hat{g}_R$, can compute for each block z the two possible pre-images $\{z, z^\star\}$. We call a block *bad* if the first two bits of $z, z^\star$ are equal; otherwise it is called *good*. If more than 1/3 of the blocks are bad then the protocol halts (it is important to note that the functions in $\hat{\mathcal{G}}$ do not change the first two bits; therefore *both* players, including the server who does not have the trapdoor, can tell which block is bad and which is not). We call a pair of blocks $z_{j,L}, z_{j,R}$ *good* if both blocks are good; otherwise the pair is bad.
4. *Dealing with bad pairs of blocks:*
   The user chooses two more vectors H_K^L (independent of $H_1^L, \ldots, H_{K-1}^L$) and H_K^R (independent of $H_1^R, \ldots, H_{K-1}^R$). It sends these vectors to the server. In return, for each bad pair $z_{j,L}, z_{j,R}$, the server sends $\langle H_K^L, \hat{g}_L(z_{j,L})\rangle$ and $\langle H_K^R, \hat{g}_R(z_{j,R})\rangle$. In this case both $z_{j,L}, z_{j,R}$ become known to the user.
5. *Dealing with good pairs of blocks:*
   Assume that the bit x_i is in some block $z_{s,L}$, for some good pair $z_{s,L}, z_{S,R}$ (if i is in a pair where at least one of the blocks is bad then in fact the user already knows the block from the previous step and can continue in an arbitrary manner). The user picks random $r_L, r_R \in \{0,1\}^K$ such that

   $$r_L(z_{s,L}) \neq r_L(z^\star_{s,L}) \quad \text{and} \quad r_R(z_{s,R}) = r_R(z^\star_{s,R}).$$

   (If the index x_i is in block $z_{s,R}$ then r_L, r_R are chosen subject to the constraint $r_R(z_{s,R}) \neq r_R(z^\star_{s,R})$ and $r_L(z_{s,L}) = r_L(z^\star_{s,L})$.)
   (a) The user sends r_L, r_R to the server.
   (b) For every good pair $z_{j,L}, z_{j,R}$ the server computes and sends the bit $b_j = r_L(z_{j,L}) \oplus r_R(z_{j,R})$.
   (c) By the choice of r_L, r_R the bit b_s allows the user to compute the value of $z_{s,L}$ (or the value of $z_{s,R}$ depending on the way that r_L, r_R were chosen). This gives the user the bit x_i (as well as all other bits in the corresponding block).

Remark: Improvements similar to those described in Section 3.2 are possible in this case as well; details are omitted for lack of space.

Correctness: The correctness is similar to the correctness of the protocol in Section 3; one difference, which is not crucial for the correctness argument, is the use of the interactive hashing (i.e., $H_1^L, \ldots, H_{K-1}^L$ and $H_1^R, \ldots, H_{K-1}^R$) instead of "standard hashing" (i.e., apply the functions $h_L, h_R \in \mathcal{H}$ and chop the last bit). The second difference, is the treatment of bad pairs; however, from the point of view of correctness this is an easy case since both blocks of each such pair become known to the user. The only significant difference is the fact that the protocol may halt without the user retrieving x_i (Step 3). However, the properties of interactive hashing guarantee that if the server plays honestly, then the probability of each block being bad (i.e., both pre-images start with the same 2 bits) is $1/4$; hence, By Chernoff bound, the probability in the case of *honest* server that at least $1/3$ of the blocks are bad is exponentially small in the number of blocks (i.e., $2\ell = n/K$). (Note that if the server is dishonest in a way that makes more than $1/3$ of the blocks bad then the protocol is aborted.)

Communication complexity: The only messages sent by the user are those for specifying the vectors $H_1^L, \ldots, H_K^L$ and $H_1^R, \ldots, H_K^R$ as well as $\hat{g}_L, \hat{g}_R, r_L, r_R$; all together $O(K^2)$ bits. The server, on the other hand, sends for each pair of blocks $2(K-1)$ bits in the interactive hashing protocol (Step 2). If the protocol halts in Step 3 (either because the server is dishonest or just because of "bad luck") then there is no more communication. Otherwise, for each bad pair the server sends two more bits (and at most $2/3$ of the pairs are bad) and for each good pair it sends only one additional bit (and at least $1/3$ of the pairs are good). All together, at most $n - \frac{n}{6K}$ bits. Therefore, the communication complexity is as claimed by the theorem.

4.1 Proof of Security (sketch)

Here we provide the high level ideas for the proof of security in the malicious case. Suppose that the malicious server can distinguish two indices i and i'. The first (simple-yet-important) observation is that if the index that the user wishes the retrieve happens to be (in a certain execution) in a bad pair of blocks then all the messages sent by the user during this execution are independent of the index. This allows us to concentrate on the good pairs only.

Using the same notation as in the honest-but-curious case (Section 3.1), and repeating a similar hybrid argument we conclude that (in a typical case) there is a distinguisher that can tell pairs r_L, r_R which are drawn from the distribution

$$\Pi_1 : \quad (N, E), (\star, \star), (\star, \star), \ldots, (\star, \star)$$

and pairs which are drawn from the distribution

$$\Pi_2' : \quad (E, E), (\star, \star), (\star, \star), \ldots, (\star, \star).$$

This again is turned into a predictor for the Goldreich-Levin hard-core predicate. Specifically, let D be the distinguisher between Π_1 and Π_2'. Our prediction algorithm B on input $g \in I_{\mathcal{G}}(1^{K-2}), w \in_R \{0,1\}^{K-2}$ construct an input for D

as follows: As before it chooses $\hat{g}_R \in I_{\hat{\mathcal{G}}}(1^K)$, including its trapdoor (the corresponding r_R is chosen at random, based on the transcript of the interactive hashing, subject to the constraint that $r_R(z^\star_{1,R}) = r_R(z_{1,R})$). Next, B chooses 3 functions $g', g'', g''' \in I_{\mathcal{G}}(1^{K-2})$ and uses them together with g (in a random order) to define a function $\hat{g} \in \hat{\mathcal{G}}$ (note that $\hat{g}$ is distributed as if it was chosen directly from $I_{\hat{\mathcal{G}}}(1^K)$). Suppose that g is $g_{b_1 b_2}$ with respect to $\hat{g}$. Next B makes sure that in the interactive hashing protocol corresponding to block $z_{1,L}$ one of the two pre-images will be $b_1 b_2 g^{-1}(w)$ (the properties of interactive hashing guarantee that this is possible; this is done by standard "rewinding" techniques, see [33, 29]). Now, there are two cases: either the first block is bad (in which case, as explained above, it cannot be of help for the distinguisher D) or the block is good. If the block is good then this means that one of the two pre-images is $b_1 b_2 g^{-1}(w)$ and the other is $b'_1 b'_2 g^{-1}_{b'_1 b'_2}(w')$, for some function $g_{b'_1 b'_2}$ different than g (by the definition of the block being good). Since for each function other than g, the algorithm B knows the trapdoor then obtaining from D the information whether the block is of type "E" or type "N" suffices for computing $r_L(g^{-1}(w))$ as required.

5 Concluding Remarks

In this paper we show how based on one-way trapdoor permutations, one can get single-server PIR protocols with communication complexity smaller than n, hence overcoming impossibility results that show that no such protocols exist under certain weaker assumptions [9, 2, 12]. A major open problem is to lower the communication complexity so that it will be comparable to what can be achieved based on specific assumptions [27, 7].

Another interesting observation is that combining our results with results of Naor and Pinkas [30], one can obtain a single-server SPIR protocol [16, 27] (i.e., a 1-out-of-n OT with "small" communication complexity) based on any one-way trapdoor permutations whose communication complexity is strictly smaller than n. In contrast, all previous communication-efficient SPIR protocols required specific algebraic assumptions [27, 39, 7, 30]. Specifically, [30] show how to implement SPIR based on a single invocation of PIR and an additional $\log n$ invocations of 1-out-of-2 OT on K-bit strings (their construction uses pseudo-random functions, however those can be implemented from any one-way function [21]). Since implementing 1-out-of-2 OT based on one-way trapdoor permutations can be done with communication complexity which is polynomial in K [20], the total communication complexity of our SPIR protocol is still smaller than n (for sufficiently small K) and we need only the assumption of a one-way trapdoor permutation. This result can also be easily extended to 1-out-of-n *string* Oblivious Transfer with total communication less than the total size of all the secrets.

Acknowledgment

We thank the anonymous referees for some useful comments. We also thank LeGamin bistro for good snacks that fueled this research.

References

1. A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *Proc. of 24th ICALP*, volume 1256 of *Lecture Notes in Computer Science*, pages 401–407, 1997.
2. A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single-server private information retrieval. In *Proc. of the 31th Annu. ACM Symp. on the Theory of Computing*, 1999.
3. M. Bellare, S. Halevi, A. Sahai, and S. Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 283–298. Springer-Verlag, 1998.
4. G. Brassard, C. Crepeau and J.-M. Robert All-or-nothing disclosure of secrets In *Advances in Cryptology: Proceedings of Crypto '86* Springer-Verlag, 1987, pp. 234-238.
5. C. Crépeau. Equivalence between two flavors of oblivious transfers. In *Proc. of CRYPTO '87*, pages 350–354, 1988.
6. C. Cachin, C. Crepeau, and J. Marcil. Oblivious transfer with a memory-bounded receiver. In *Proc. 39th IEEE Symposium on Foundations of Computer Science* (FOCS), pp. 493-502, 1998.
7. C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology - EUROCRYPT '99*, 1999.
8. B. Chor and N. Gilboa. Computationally private information retrieval. In *Proc. of the 29th Annu. ACM Symp. on the Theory of Computing*, pages 304–313, 1997.
9. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. of the 36th Annu. IEEE Symp. on Foundations of Computer Science*, pages 41–51, 1995. Journal version in JACM, Vol. 45(6), 1998, pp. 965-981.
10. I. Damgard. Interactive hashing can simplify zero-knowledge protocol design without computational assumptions. In D. R. Stinson, editor, *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.
11. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for database private information retrieval. In *Proc. of the 17th Annu. ACM Symp. on Principles of Distributed Computing*, pages 91–100, 1998.
12. G. Di Crescenzo, T. Malkin, and R. Ostrovsky. Single-database private information retrieval implies oblivious transfer. In *Advances in Cryptology - EUROCRYPT 2000*, Lecture Notes in Computer Science, volume 1807, Springer-Verlag, 2000, pp. 122–139 (this volume).
13. S. Even, O. Goldreich and A. Lempel A Randomized Protocol for Signing Contracts Communications of the ACM, Vol 28, 1985, pp. 637-447.
14. J. Feigenbaum and R. Ostrovsky A note on one-prover, instance-hiding zero-knowledge proof systems In Advances in Cryptology – Asiacrypt'91, Lecture Notes in Computer Science, volume 739, Springer, Berlin, 1993, pp. 352–359.

15. Y. Gertner, S. Goldwasser, and T. Malkin. A random server model for private information retrieval. In M. Luby, J. Rolim, and M. Serna, editors, *RANDOM '98, 2nd International Workshop on Randomization and Approximation Techniques in Computer Science*, volume 1518 of *Lecture Notes in Computer Science*, pages 200–217. Springer, 1998.
16. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. of the 30th Annu. ACM Symp. on the Theory of Computing*, pages 151–160, 1998.
17. O. Goldreich. *Foundations of Cryptography (fragments of a book).* Electronic Colloquium on Computational Complexity, 1995. Electronic publication: http://www.eccc.uni-trier.de/eccc-local/ECCC-Books/eccc-books.html.
18. O. Goldreich, S. Goldwasser, and N. Linial. Fault-tolerant computation in the full information model: the Two-party Case In *Proc. of the 32nd Annu. IEEE Symp. on Foundations of Computer Science*, pages 447–457, 1991. Journal Version in *SIAM J. on Computing*, Vol. 27(3), 1998, pp. 505-544.
19. O. Goldreich and L. Levin. A hard predicate for all one-way functions. In *Proc. of the 21st Annu. ACM Symp. on the Theory of Computing*, pages 25–32, 1989.
20. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. of the 19th Annu. ACM Symp. on the Theory of Computing*, pages 218–229, 1987.
21. J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. Technical Report TR-91-068, International Computer Science Institute, 1991.
22. R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proc. of the 30th Annu. IEEE Symp. on Foundations of Computer Science*, pages 230–235, 1989.
23. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proc. of the 21st Annu. ACM Symp. on the Theory of Computing*, pages 44–61, 1989.
24. Y. Ishai and E. Kushilevitz. Improved upper bounds on information theoretic private information retrieval. In *Proc. of the 31th Annu. ACM Symp. on the Theory of Computing*, 1999.
25. J. Kilian. Basing cryptography on oblivious transfer. In *Proc. of the 20th Annu. ACM Symp. on the Theory of Computing*, pages 20–31, 1988.
26. J. Kilian. A general completeness theorem for two-party games. In *Proc. of the 23th Annu. ACM Symp. on the Theory of Computing*, pages 553–560, 1991.
27. E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of the 38th Annu. IEEE Symp. on Foundations of Computer Science*, pages 364–373, 1997.
28. E. Mann. Private access to distributed information. Master's thesis, Technion - Israel Institute of Technology, Haifa, 1998.
29. M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP can be based on general complexity assumptions. In E. F. Brickell, editor, *Advances in Cryptology - CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992. Final version in *J. Cryptology*, 11(2):87–108, 1998.
30. M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. of the 31th Annu. ACM Symp. on the Theory of Computing*, pages 245–254, 1999.
31. M. Naor and M. Yung. Universal one-way functions and their cryptographic applications. In *Proc. of the 21st Annu. ACM Symp. on the Theory of Computing*, pages 33–43, 1989.

32. R. Ostrovsky and V. Shoup. Private information storage. In *Proc. of the 29th Annu. ACM Symp. on the Theory of Computing*, pages 294–303, 1997.
33. R. Ostrovsky, R. Venkatesan, and M. Yung. Fair games against an all-powerful adversary. Presented at DIMACS Complexity and Cryptography workshop, October 1990, Princeton. Prelim. version in *Proc. of the Sequences II workshop* 1991, Springer-Verlag, pp. 418-429. Final version in *AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 13 *Distributed Computing and Cryptography*, Jin-Yi Cai, editor, pp. 155-169. AMS, 1993.
34. R. Ostrovsky, R. Venkatesan, and M. Yung. Secure Commitment Against Powerful Adversary: A Security Primitive based on Average Intractability. In Proceedings of 9th Symposium on Theoretical Aspects of Computer Science (STACS-92) (LNCS 577 Springer Verlag Ed. A. Finkel and M. Jantzen) pp. 439-448 February 13-15 1992, Paris, France.
35. R. Ostrovsky, R. Venkatesan, and M. Yung. Interactive hashing simplifies zero-knowledge protocol design. In *Advances in Cryptology - EUROCRYPT '93*, Lecture Notes in Computer Science. Springer-Verlag, 1993.
36. R. Ostrovsky and A. Wigderson One-Way Functions are Essential for Non-Trivial Zero-Knowledge. In *Proceedings of the Second Israel Symposium on Theory of Computing and Systems* (ISTCS-93) pp. 1-10., IEEE 1993.
37. M. O. Rabin How to exchange secrets by oblivious transfer Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.
38. J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proc. of the 22nd Annu. ACM Symp. on the Theory of Computing*, pages 387–394, 1990.
39. J. P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *ASIACRYPT '98*, 1998.
40. A. C. Yao. Theory and application of trapdoor functions. In *Proc. of the 23th Annu. IEEE Symp. on Foundations of Computer Science*, pages 80–91, 1982.

Single Database Private Information Retrieval Implies Oblivious Transfer

Giovanni Di Crescenzo[1], Tal Malkin[2], and Rafail Ostrovsky[1]

[1] Telcordia Technologies, Inc., 445 South Street, Morristown, NJ, 07960.
E-mail: {giovanni,rafail}@research.telcordia.com.
[2] AT&T Labs – Research, 180 Park Ave., Florham Park, NJ, 07932.
E-mail: tal@research.att.com.
Work done at the Massachusetts Institute of Technology.

Abstract. A Single-Database Private Information Retrieval (PIR) is a protocol that allows a user to privately retrieve from a database an entry with as small as possible communication complexity. We call a PIR protocol *non-trivial* if its total communication is strictly less than the size of the database. Non-trivial PIR is an important cryptographic primitive with many applications. Thus, understanding which assumptions are necessary for implementing such a primitive is an important task, although (so far) not a well-understood one. In this paper we show that any non-trivial PIR implies Oblivious Transfer, a far better understood primitive. Our result not only significantly clarifies our understanding of any non-trivial PIR protocol, but also yields the following consequences:

- Any non-trivial PIR is *complete* for all two-party and multi-party secure computations.
- There exists a communication-efficient reduction from any PIR protocol to a 1-out-of-n Oblivious Transfer protocol (also called SPIR).
- There is strong evidence that the assumption of the existence of a one-way function is necessary but not sufficient for any non-trivial PIR protocol.

1 Introduction

Relationships between cryptographic primitives. One of the central questions in cryptography is to study which assumptions (if any) are necessary to implement a cryptographic protocol or task. For most primitives this answer is well understood, and falls in two categories: either one-way functions are necessary and sufficient, or stronger assumptions are necessary (i.e., one-way functions with some additional properties like trapdoor may be required). For example, pseudo-random generators [20], signature schemes [32, 36], commitment schemes [20, 30] and zero-knowledge proofs for NP [20, 30, 18, 34] are all equivalent to the existence of a one-way function. On the other hand there is a class of primitives that probably needs additional assumptions, including, for example, public-key cryptosystems, key-exchange, oblivious transfer [22], non-interactive zero-knowledge proofs of knowledge for NP [11] and any non-trivial

secure two-party [4] and multi-party function evaluation [25]. Single Database *Private Information Retrieval* has received a lot of attention in the literature, however its place in the above setting was not understood. In this paper we address (and resolve) its position.

PRIVATE INFORMATION RETRIEVAL. A Private Information Retrieval (PIR) scheme allows a user to retrieve information from a database while maintaining the query private from the database managers. More formally, the database is modeled as an n-bit string x out of which the user retrieves the i-th bit x_i, while giving the database no information about the index i. The communication complexity of such a scheme is denoted by $c(n)$. A trivial PIR scheme consists of sending the entire data string to the user (i.e. $c(n)=n$), thus satisfying the PIR privacy requirement in the information-theoretic sense. We call any PIR protocol with $c(n)<n$ *non-trivial*. The problem of constructing non-trivial PIR was originally introduced by Chor et al. [8] and further studied in [8, 1, 7, 33, 27, 29, 3, 12, 16, 15, 6, 23, 28]. In [8] this problem was studied in the setting of multiple non-communicating copies of the database (further improvements were given in [1, 23]). That is, [8] show that if there are at least two or more copies of the database, then non-trivial PIR (for example, with two copies of the database, with communication complexity $c(n)=O(n^{1/3})$) is indeed possible. In the original work [8] also show that it is information-theoretically impossible to achieve a non-trivial PIR with a single copy of the database. Kushilevitz and Ostrovsky [27] have shown a way to get around this impossibility result using computational assumptions[1]. In particular, [27] show that assuming that the quadratic residuosity (number-theoretic) problem is hard, they can get Single-Database PIR protocol with $c(n) < n^{\epsilon}$ for any $\epsilon > 0$. Further constructions of single-database PIR schemes, improving either the communication or the assumption, followed [29, 37, 6, 28]. In particular, Cachin et al. [6] construct PIR with polylogarithmic communication complexity, under the so-called Φ-hiding (number-theoretic) assumption. This is essentially optimal communication complexity since the security parameter needs to be at least poly-logarithmic in n. Recently, [28] have shown a single database PIR based on any one-way trapdoor permutation, though their communication, while less then n, is bigger than schemes based on specific number-theoretic assumptions [27, 29, 37, 6]. On the other hand, [3] have shown that any non-trivial single database PIR implies the existence of a one-way function.

OBLIVIOUS TRANSFER. The Oblivious Transfer (OT) protocol was introduced by Rabin [35], one-out-of-two Oblivious Transfer, denoted $\binom{2}{1}$-OT , was introduced in [13], and one-out-of-n Oblivious Transfer, denoted $\binom{n}{1}$-OT , was introduced in [2]. All these OT variants were shown to be equivalent to one another [10, 2]. In this paper, we will mainly use the last two versions. Roughly speaking, $\binom{2}{1}$-OT is a protocol between two players, a sender Alice and a receiver Bob.

[1] Also, [7, 33] consider the use of computational assumptions in the settings of multiple non-communicating databases.

Alice has two bits, and Bob wishes to get one of them such that (a) Alice does not know which bit Bob got; and (b) Bob does not learn any information about the bit that he did not get. When generalized to $\binom{n}{1}$-OT we can see that the formulation of this primitive is "close" to single-database PIR, in that they both share requirement (a). However, non-trivial PIR has *an additional* requirement regarding the communication complexity (to be less than the number of bits) and *does not require* condition (b) – which is essential for the definition of Oblivious Transfer. The $\binom{n}{1}$-OT protocol that combines both requirements (a), (b) and the small communication requirement was considered in [16], who call it Symmetric-PIR.

In [24], it was shown that OT is complete, namely it can be used to construct any other protocol problem. [21] have shown that OT implies the existence of one-way functions. Moreover, [22] have shown that assuming OT is probably stronger than assuming existence of one-way functions (OWF) in the following sense. They show that it is impossible to construct a black-box reduction from OT to OWF (where the OT protocol uses the promised OWF as a black box, and the proof is black-box). Furthermore, proving any such black-box construction (even if the proof itself is not black-box), is as hard as separating $\mathcal{P}$ from $\mathcal{NP}$. Thus [22] gives a strong evidence that OWF are currently not sufficient to construct OT, namely that OT is a strictly stronger assumption.

Our Results

In this paper, we present a reduction transforming any nontrivial single-database PIR into Oblivious Transfer. The significance of this reduction is threefold: (1) it provides "negative" results, asserting that PIR cannot be constructed based on weak computational assumptions; (2) It provides a general "positive" result, namely that PIR is also a complete primitive, and any non-trivial implementation of Single-Database PIR may be used to construct any other secure protocol; and (3) it provides a specific "positive" result, allowing transformation from communication efficient single-database PIR to communication-efficient $\binom{n}{1}$-OT (also called Symmetric-PIR [16]). We elaborate below.

COMPLEXITY OF PIR. As mentioned above, the original paper of Chor et al. [8] shows that it is information-theoretically impossible to implement a non-trivial Single-Database PIR. That is, if the user needs information-theoretic privacy, the communication cannot be less than n. Thus, *some* computational assumption is necessary. Naturally, this leads to the following question.

> *Under which computational assumptions can non-trivial Single-Database PIR be achieved?*

While this question has received a lot of attention recently [27, 29, 37, 6, 3, 28], only limited progress has been achieved thus far towards a solution. In particular, as described above, there has been a large gap between the assumptions known to be *sufficient*, and those known to be *necessary*. On one hand, the only assumption

previously known to be necessary for non-trivial PIR is the existence of one-way functions [3]; on the other hand, the weakest assumptions known to be sufficient are trapdoor permutations [28]. In this paper we make an important step towards closing this gap, by showing the following

Main Theorem (Informal Statement) *If there exists any non-trivial Single-Database PIR then there exists an OT.*

That is, even saving one bit compared to the (information-theoretic) trivial protocol of sending the entire database, already requires OT. It is interesting to note that we can also reduce any code for non-trivial single-database PIR to a code for OT; this is similar to code-to-code reductions in [4]. Moreover, our theorem holds even if the communication sent by the user in the given PIR scheme is unbounded, as long as the database sends less than n bits.

OT protocol implies the existence of a one-way function [21]. Single database PIR also implies the existence of a one-way function [3], but in light of [22] our result is strictly stronger and implies the following:

Corollary (Informal Statement) *One-way functions are necessary but probably not sufficient to construct non-trivial Single-Database PIR.*

Completeness of Any non-trivial Single-Database PIR. The following corollary, demonstrating the importance of the PIR primitive, follows from the result of the completeness of OT [24]:

Corollary (Informal Statement) *Any non-trivial Single-Database PIR is complete for all two-party and multi-party secure computation.*

That is, an implementation of the PIR primitives allows a secure computation of *any* function.

Symmetric-PIR (or communication-efficient $\binom{n}{1}$-OT). In the standard formulation of PIR, there is no concern about how many bits of the database the user learns. If one makes an additional requirement that the user must learn only one bit (or secret) of the database, then this can be viewed as communication-efficient $\binom{n}{1}$-OT (called Symmetrically Private Information Retrieval (SPIR)). SPIR schemes were first introduced in [16] in the setting of multiple databases. In [27] SPIR were shown to exist in the setting of a single database. The single-database SPIR schemes of [27, 16, 37] were based on specific algebraic assumptions. Naor and Pinkas [31] have shown a general reduction transforming any single database PIR into single-database SPIR using one call to the underlying PIR protocol, a logarithmic number of calls to one-out-of-two (string) Oblivious Transfer, and the existence of pseudo-random generators. Combining our main result with that of [31] we get:

Theorem (Informal Statement) *If there exists any non-trivial Single-Database PIR scheme with communication $c(n)$ and security parameter k, then there exists $\binom{n}{1}$-OT (i.e., SPIR) with communication $c(n) \cdot \mathrm{poly}(k)$.*

We stress that the efficient communication complexity of the SPIR scheme we construct is the main point of the last theorem. Indeed, in the context of computational assumptions, SPIR is equivalent to the $\binom{n}{1}$-OT variant of Oblivious Transfer. However, this theorem provides a stronger result, since the communication complexity obtained (which is the main parameter in the SPIR context) is efficient, costing only a factor depending on the security parameter (not on n) over the underlying PIR. In particular, when given PIR scheme with a sublinear communication, the resulting SPIR scheme also has sublinear communication.

PROOF OUTLINE. The variant of OT that we use here is the $\binom{2}{1}$-OT . We prove our results using the following three steps: (1) communication-efficient PIR implies $\binom{2}{1}$-OT for honest parties; (2) communication-efficient PIR implies $\binom{2}{1}$-OT (for possibly dishonest parties); (3) communication-efficient PIR implies communication-efficient SPIR.

2 Preliminaries and Definitions

In this section we give some general conventions that we will use in the paper and the formal definitions for PIR, SPIR, and OT.

GENERAL CONVENTIONS Let $\mathbb{N}$ be the set of natural numbers and define $[k] = \{1, \ldots, k\}$. If S is a set, the notation $x \leftarrow S$ denotes the *random process* of selecting element x from set S with uniform probability distribution over S and independently from all other random choices. If A is an algorithm, the notation $y \leftarrow A(x)$ denotes the random process of obtaining y when running algorithm A on input x, where the probability space is given by uniformly and independently choosing the random coins (if any) of algorithm A. By $\mathrm{Prob}[R_1; \ldots ; R_n : E]$ we denote the probability of event E, after the execution of random processes $R_1, \ldots, R_n$. We denote a distribution D as $\{R_1; \ldots ; R_m : v\}$, where v denotes the values that D can assume, and $R_1, \ldots, R_m$ is a sequence of random processes generating value v. By *algorithm* we refer to a (probabilistic) Turing machine. An *interactive Turing machine* is a probabilistic Turing machine with a communication tape. A pair (A, B) of interactive Turing machines running in probabilistic polynomial time is an *interactive protocol*. A *transcript* of an execution of an interactive protocol is the sequence of messages that appear on the communication tapes of the two machines forming the protocol during that execution. The notation $t_{A,B}(x, r_A, y, r_B)$ denotes the transcript of an execution of an interactive protocol (A, B) with inputs x for A and y for B and with random strings r_A for A and r_B for B. If $t = t_{A,B}(x, r_A, y, r_B)$ is such a transcript, the output of A (resp. B) on this execution is denoted by $A(x, r_A, t)$ (resp. $B(y, r_B, t)$). The notation $(r_B, t) \leftarrow t_{A,B}(x, r_A, y, \cdot)$ denotes the random process of selecting a random string r_B uniformly at random (and independently of all other choices), and setting $t = t_{A,B}(x, r_A, y, r_B)$. Similarly we denote $(r_A, t) \leftarrow t_{A,B}(x, \cdot, y, r_B)$ for the case where A's random string is chosen uniformly at random, and $(r_A, r_B, t) \leftarrow t_{A,B}(x, \cdot, y, \cdot)$ for the case where the random strings for both A and B are chosen uniformly at random.

PRIVATE INFORMATION RETRIEVAL. Informally, a private information retrieval (PIR) scheme is an interactive protocol between two parties, a database $\mathcal{D}$ and a user $\mathcal{U}$. The database holds a *data string* $x \in \{0,1\}^n$, and the user holds an *index* $i \in [n]$. In its one-round version, the protocol consists of (a) a query sent from the user to the database (generated by an efficient randomized query algorithm, taking as an input the index i and a random string $r_{\mathcal{U}}$); (b) an answer sent by the database (generated by an efficient deterministic (without loss of generality) answer algorithm, taking as an input the query sent by the user and the database x); and (c) an efficient reconstruction function applied by the user (taking as an input the index i, the random string $r_{\mathcal{U}}$, and the answer sent by the database). At the end of the execution of the protocol, the following two properties must hold: (1) after applying the reconstruction function, the user obtains the i-th data bit x_i; and (2) the distributions on the query sent to the database are computationally indistinguishable for any two indices i, i'. (That is, a computationally bounded database does not receive any information about the index of the user). We now give a formal definition of a PIR scheme.

Definition 1. (Private Information Retrieval Scheme.) Let $(\mathcal{D},\mathcal{U})$ be an interactive protocol, and let $\mathcal{R}$ be a polynomial time algorithm[2]. We say that $(\mathcal{D},\mathcal{U},\mathcal{R})$ is a *private information retrieval (PIR) scheme* if:

1. (*Correctness.*) For each $n \in \mathbb{N}$, each $i \in \{1,\ldots,n\}$, each $x \in \{0,1\}^n$, where $x = x_1 \circ \cdots \circ x_n$, and $x_l \in \{0,1\}$ for $l = 1,\ldots,n$, and for all constants c, and all sufficiently large k,

$$\text{Prob}[\,(r_{\mathcal{D}}, r_{\mathcal{U}}, t) \leftarrow t_{\mathcal{D},\mathcal{U}}(\,(1^k, x), \cdot, (1^k, n, i), \cdot) : \mathcal{R}(1^k, n, i, r_{\mathcal{U}}, t) = x_i\,] \geq 1 - k^{-c}.$$

2. (*User Privacy.*) For each $n \in \mathbb{N}$, each $i, j \in \{1,\ldots,n\}$, each $x \in \{0,1\}^n$, where $x = x_1 \circ \cdots \circ x_n$, and $x_l \in \{0,1\}$ for $l = 1,\ldots,n$, for each polynomial time $\mathcal{D}'$, for all constants c, and all sufficiently large k, it holds that $|p_i - p_j| \leq k^{-c}$, where

$$p_i = \text{Prob}[\,(r_{\mathcal{D}'}, r_{\mathcal{U}}, t) \leftarrow t_{\mathcal{D}',\mathcal{U}}(\,(1^k, x), \cdot, (1^k, n, i), \cdot) : \mathcal{D}'(1^k, x, r_{\mathcal{D}'}, t) = 1\,]$$
$$p_j = \text{Prob}[\,(r_{\mathcal{D}'}, r_{\mathcal{U}}, t) \leftarrow t_{\mathcal{D}',\mathcal{U}}(\,(1^k, x), \cdot, (1^k, n, j), \cdot) : \mathcal{D}'(1^k, x, r_{\mathcal{D}'}, t) = 1\,].$$

We say that $(\mathcal{D},\mathcal{U},\mathcal{R})$ is an *honest-database PIR scheme* if it is a PIR scheme in which the user-privacy requirement is relaxed to hold only for $\mathcal{D}'$ that follow the protocol execution as $\mathcal{D}$.

For sake of generality, the above definition does not pose any restriction on the number of rounds of protocol $(\mathcal{D},\mathcal{U})$; however, we remark that the most studied case in the literature is that of one-round protocols (as discussed above).

[2] For clarity, we chose to include the reconstruction function $\mathcal{R}$ as an explicit part of the PIR definition. We note however that replacing $\mathcal{R}$ by $\mathcal{U}$ in the correctness requirement yields an equivalent definition (where the reconstruction function is an implicit part of $\mathcal{U}$, who executes it to produce an output).

SYMMETRICALLY PRIVATE INFORMATION RETRIEVAL. Informally, a symmetrically private information retrieval (SPIR) scheme is a PIR scheme satisfying an additional privacy property: data privacy. Namely, for each execution, there exists an index i, such that the distributions on the user's view are computationally indistinguishable for any two databases x, y such that $x_i = y_i$. (That is, a computationally bounded user does not receive information about more than a single bit of the data). We now give a formal definition of a SPIR scheme.

Definition 2. (Symmetrically Private Information Retrieval Scheme)
Let $(\mathcal{D}, \mathcal{U}, \mathcal{R})$ be a PIR scheme. We say that $(\mathcal{D}, \mathcal{U}, \mathcal{R})$ is a *symmetrically private information retrieval (SPIR) scheme* if in addition it holds that

3. (*Data Privacy.*) For each $n \in \mathbb{N}$, for each polynomial time $\mathcal{U}'$, each $i' \in \{1, \ldots, n\}$, and each random string $r_{\mathcal{U}'}$, there exists an $i \in \{1, \ldots, n\}$, such that for each $x, y \in \{0,1\}^n$ where $x = x_1 \circ \cdots \circ x_n$ and $y = y_1 \circ \cdots \circ y_n$, $x_l, y_l \in \{0,1\}$ for $l = 1, \ldots, n$, and such that $x_i = y_i$, for all constants c and all sufficiently large k, it holds that $|p_x - p_y| \leq k^{-c}$, where

$$p_x = \text{Prob}[\,(r_{\mathcal{D}}, t) \leftarrow t_{\mathcal{D},\mathcal{U}'}((1^k, x), \cdot, (1^k, n, i'), r_{\mathcal{U}'}) : \mathcal{U}'(1^k, n, i', r_{\mathcal{U}'}, t) = 1\,]$$
$$p_y = \text{Prob}[\,(r_{\mathcal{D}}, t) \leftarrow t_{\mathcal{D},\mathcal{U}'}((1^k, y), \cdot, (1^k, n, i'), r_{\mathcal{U}'}) : \mathcal{U}'(1^k, n, i', r_{\mathcal{U}'}, t) = 1\,].$$

OBLIVIOUS TRANSFER. Informally, a $\binom{2}{1}$-Oblivious Transfer ($\binom{2}{1}$-OT) is an interactive protocol between Alice, holding two bits b_0, b_1, and Bob, holding a selection bit c. At the end of the protocol, Bob should obtain the bit b_c, but no information about $b_{\bar{c}}$, whereas Alice should obtain no information about c. (By "obtaining no information" we mean that the two possible views are indistinguishable.) The extension to $\binom{n}{1}$-OT is immediate. A formal definition follows.

Definition 3. ($\binom{2}{1}$-Oblivious Transfer)
Let (Alice, Bob) be an interactive protocol. We say that (Alice, Bob) is a $\binom{2}{1}$-*Oblivious Transfer* ($\binom{2}{1}$-*OT*) protocol with security parameter k if it holds that:

1. *(Correctness).* For all $b_0, b_1, c \in \{0,1\}$, all constants d, and all sufficiently large k,

$$\text{Prob}[\,(r_A, r_B, t) \leftarrow t_{\text{Alice,Bob}}((1^k, b_0, b_1), \cdot, (1^k, c), \cdot) : \text{Bob}(1^k, c, r_B, t) = b_c\,] \geq 1 - k^{-d}.$$

2. *(Privacy against Alice).* For all probabilistic polynomial time Alice$'$, all $b_0, b_1 \in \{0,1\}$, all constants d, and all sufficiently large k,

$$\text{Prob}\,[\,c \leftarrow \{0,1\}; (r_{A'}, r_B, t) \leftarrow t_{\text{Alice}',\text{Bob}}((1^k, b_0, b_1), \cdot, (1^k, c), \cdot) : \text{Alice}'(1^k, b_0, b_1, r_{A'}, t) = c\,] \leq 1/2 + k^{-d}.$$

3. *(Privacy against Bob).* For all probabilistic polynomial time Bob$'$, all $c' \in \{0,1\}$, and all random strings $r_{B'}$, there exists $c \in \{0,1\}$ such that for all constants d, and all sufficiently large k,

$$\text{Prob}\,[\,(b_0, b_1) \leftarrow \{0,1\}^2; (r_A, t) \leftarrow t_{\text{Alice,Bob}'}((1^k, b_0, b_1), \cdot, (1^k, c'), r_{B'}) : \text{Bob}'(1^k, c', r_{B'}) = b_{\bar{c}}\,] \leq 1/2 + k^{-d}.$$

We say that (Alice, Bob) is an *honest-Bob-*$\binom{2}{1}$*-OT* protocol if it is a $\binom{2}{1}$-OT protocol in which privacy against Bob is relaxed to hold only when Bob is honest (but curious). That is, condition (3) in Definition 3 is relaxed to

3'. *(Privacy against honest-but-curious-Bob).* For all probabilistic polynomial time CuriousB, for all constants d, and all sufficiently large k,

$$\text{Prob}\,[\,(b_0,b_1)\leftarrow\{0,1\}^2;(r_A,r_B,t)\leftarrow t_{\text{Alice,Bob}}((1^k,b_0,b_1),\cdot,(1^k,c),\cdot):$$
$$\text{CuriousB}(1^k,c,r_B,t)=b_{\bar{c}}\,]\leq 1/2+k^{-d}.$$

We say that (Alice, Bob) is an *honest-parties-*$\binom{2}{1}$*-OT* protocol if it is a $\binom{2}{1}$-OT protocol where privacy requirements are relaxed to hold only when both Alice and Bob are honest-but-curious; that is, (Alice, Bob) should satisfy correctness, privacy against honest-but-curious Bob (as defined above), and privacy against honest-but-curious Alice (which is similarly defined).
We remark that the definitions of $\binom{2}{1}$-OT and its honest-but-curious versions are extended in the obvious way to the case of $\binom{n}{1}$-OT, for any $n\geq 3$.

Communication Complexity. Let $(\mathcal{D},\mathcal{U},\mathcal{R})$ be a PIR scheme. We define the *communication complexity* of $(\mathcal{D},\mathcal{U},\mathcal{R})$ as the maximal length $c(n)$ of a transcript returned by a possible execution of $(\mathcal{D},\mathcal{U},\mathcal{R})$ where n is the size of $\mathcal{D}$'s input (i.e. the length of the database). We define the *database communication complexity* as the maximal length $c_{\mathcal{D}}(n)$ of the communication sent by $\mathcal{D}$ in any execution of $(\mathcal{D},\mathcal{U},\mathcal{R})$, and similarly the *user communication complexity* $c_{\mathcal{U}}(n)$. That is, $c(n)=c_{\mathcal{D}}(n)+c_{\mathcal{U}}(n)$. The communication complexity of a SPIR scheme and of an $\binom{n}{1}$-OT scheme are similarly defined.

SPIR vs. $\binom{n}{1}$-OT. It can be easily verified that $\binom{n}{1}$-OT is equivalent to SPIR with a database of length n. The reason we need two concepts (and the reason we formulated the definitions in two different, though equivalent, ways), is the different motivations for using these primitives (and the way they were historically defined). In particular, we note that when constructing a SPIR protocol, the communication complexity is a crucial parameter.

3 PIR Implies honest-Bob-$\binom{2}{1}$-OT

In this section we construct an honest-Bob-$\binom{2}{1}$-OT protocol from any PIR scheme with database communication complexity $c_{\mathcal{D}}(k)<k$ (and arbitrary user communication complexity $c_{\mathcal{U}}(k)$), for database of length k.[3]

[3] In this section and the next we denote the database length by k, since the way it will be used will be for a database whose length depends (polynomially) on the security parameter. This is to avoid confusion with the length of the actual database n in the last section, where we construct SPIR using this $\binom{2}{1}$-OT .

THE PROTOCOL DESCRIPTION. Let $\mathcal{P} = (\mathcal{D}, \mathcal{U}, \mathcal{R})$ be a PIR scheme with database communication $c_{\mathcal{D}}(k) < k$. Our $\binom{2}{1}$-OT protocol consists of simultaneously invoking polynomially many[4] independent executions of $\mathcal{P}$ with a random data string for $\mathcal{D}$ (ran by Alice) and random indices for $\mathcal{U}$ (ran by Bob). In addition, Bob sends to Alice two sequences of indices (one consists of the indices retrieved in the PIR invocations, and one a sequence of random indices), and in response Alice sends to Bob her two secret bits appropriately masked, so that Bob can reconstruct only one of them. A formal description of protocol (Alice, Bob) is in Figure 1. We note that some related techniques to those in our construction have appeared in [5]; however, we remark that the protocol of [5] cannot be used in our case, mainly because of the differences in the models. We next prove that (Alice, Bob) is a honest-Bob-$\binom{2}{1}$-OT protocol.

CORRECTNESS. In order to prove the correctness of (Alice, Bob), we need to show that Bob outputs b_c with probability at least $1 - k^{-\omega(1)}$. First, notice that if Bob is able to correctly reconstruct all bits $x^j(i^j)$ for $j = 1, \ldots, m$, after the m executions of the PIR protocol in step 1, then he is able to compute the right value for b_c in step 5. Next, from the correctness of $\mathcal{P} = (\mathcal{D}, \mathcal{U}, \mathcal{R})$, Bob, who is playing as $\mathcal{U}$, is able to reconstruct all bits $x^j(i^j)$ with probability at least $(1 - k^{-\omega(1)})^m$ since the m executions of $(\mathcal{D}, \mathcal{U})$ are all independent. This probability is then at least $1 - k^{-\omega(1)}$ since m is polynomial in k.

PRIVACY AGAINST ALICE. In order to prove that (Alice, Bob) satisfies the property of privacy against Alice, we need to show that for any probabilistic polynomial time algorithm Alice$'$, the probability that Alice$'$, at the end of the protocol, is able to compute the bit c input to Bob is at most $1/2 + k^{-\omega(1)}$ (where probability is taken over the uniform distribution of c and the random strings of Alice$'$ and Bob). Informally, this follows from the user's privacy in the PIR subprotocol $\mathcal{P}$, which guarantees that in each invocation Alice gets no information about the index used by Bob, and thus cannot tell between the sequence of real indices used, and the sequence of random indices (since both these sequences are distributed uniformly). A more formal argument follows. Assume for the sake of contradiction that the property is not true; namely, there exists a probabilistic polynomial time algorithm Alice$'$, which, after running protocol (Alice$'$, Bob), is able to compute c with probability at least $1/2 + k^{-d}$, for some constant d and infinitely many k. In step 3, Bob sends two m-tuples (I_0, I_1) of indices to Alice$'$, such that I_c is the tuple of indices used by Bob in the PIR invocations of step 1, and $I_{\bar{c}}$ is a tuple containing random indices. Therefore, Alice$'$ is able to guess with probability at least $1/2 + k^{-d}$ which one of I_0, I_1 is the tuple of retrieved indices. This implies, by a hybrid argument, that for some position $j \in \{1, \ldots, m\}$, Alice$'$ can guess with probability at least $1/2 + k^{-d}/m$ whether in the j-th PIR invocation the index used was i_0^j or i_1^j. Since all PIR invocations

[4] The number of invocations, m, is a parameter whose value can be set based on the communication complexity of $\mathcal{P}$ and the target (negligible) probability of error in OT, but will always be polynomial in k as will become clear below.

Honest-Bob-$\binom{2}{1}$-OT

Alice's inputs: 1^k (where k is a security parameter) and $b_0, b_1 \in \{0,1\}$.
Bob's inputs: 1^k and $c \in \{0,1\}$.
Additional (common) inputs: a parameter m polynomial in k, and a PIR protocol $(\mathcal{D}, \mathcal{U}, \mathcal{R})$.

Instructions for Alice and Bob:

1. For every $j \in \{1, \ldots, m\}$ do:
   - Alice uniformly chooses a data string $x^j \in \{0,1\}^k$ (where x^j can be written as $x^j(1) \circ \cdots \circ x^j(k)$, for $x^j(i) \in \{0,1\}$).
   - Bob uniformly chooses an index $i^j \in [k]$
   - Alice and Bob invoke the PIR protocol $(\mathcal{D}, \mathcal{U}, \mathcal{R})$ where Alice plays the role of $\mathcal{D}$ on input $(1^k, x^j)$ and Bob plays the role of $\mathcal{U}$ on input $(1^k, k, i^j)$. (That is, Alice and Bob execute $(\mathcal{D}, \mathcal{U})$ on the above inputs, and then Bob applies the reconstruction function $\mathcal{R}$ to obtain the bit $x^j(i^j)$).
2. Bob sets $(i_c^1, \ldots, i_c^m) \stackrel{\text{def}}{=} (i^1, \ldots, i^m)$ ($\star$ `the indices retrieved` $\star$) and uniformly chooses $(i_{\bar{c}}^1, \ldots, i_{\bar{c}}^m)$ from $[k]^m$. ($\star$ `random indices` $\star$)
3. Bob sends to Alice $(i_0^1, \ldots, i_0^m)$ and $(i_1^1, \ldots, i_1^m)$.
4. Alice sets $z_0 \stackrel{\text{def}}{=} b_0 \oplus x^1(i_0^1) \oplus \ldots \oplus x^m(i_0^m)$, and $z_1 \stackrel{\text{def}}{=} b_1 \oplus x^1(i_1^1) \oplus \ldots \oplus x^m(i_1^m)$ and sends z_0, z_1 to Bob;
5. Bob computes $b_c = z_c \oplus x^1(i^1) \oplus \ldots \oplus x^m(i^m)$ and **outputs:** b_c.

Fig. 1. A protocol (Alice,Bob) for honest-Bob-$\binom{2}{1}$-OT , using a PIR protocol $\mathcal{P} = (\mathcal{D}, \mathcal{U}, \mathcal{R})$ with $c_{\mathcal{D}}(k) < k$ database communication complexity.

are independent (implying that the indices in different positions within I_0 and I_1 are independent), it is straightforward to use Alice$'$ to construct a $\mathcal{D}'$ which distinguishes in a single PIR execution between the index used by the user and a random index, with probability at least $1/2 + k^{-d}/m$. Since m is polynomial, this is a non-negligible advantage, and thus contradicts the user privacy of $\mathcal{P}$.

PRIVACY AGAINST HONEST-BUT-CURIOUS BOB. In order to prove that the pair (Alice, Bob) satisfies the property of privacy against a honest-but-curious Bob, we need to show that the probability that Bob, after behaving honestly in the protocol, is able to compute the bit $b_{\bar{c}}$ is at most $1/2+k^{-\omega(1)}$ (where probability is taken over the uniform distribution of b_0, b_1, and the random strings of Alice and Bob). In order to prove this property for an appropriate polynomial number m of invocations of $(\mathcal{D}, \mathcal{U})$ in step 1, we start by considering a single invocation. In the following lemma we consider the probability p that a malicious user $\mathcal{U}'$, after invoking $(\mathcal{D}, \mathcal{U}')$ where $\mathcal{D}$ uses a uniformly chosen database, fails in reconstructing a bit in a *random* location j in the database. Note that j is not known to $\mathcal{U}'$ when running $(\mathcal{D}, \mathcal{U}')$.[5] We also note that no further requirements about $\mathcal{U}'$ or its computational power are necessary. In the following we show that if the database communication complexity is less than the length of the data, this failure probability is non-negligible. This is shown by first bounding the binary entropy of the failure probability.

Lemma 1. *Let $\mathcal{P} = (\mathcal{D}, \mathcal{U}, \mathcal{R})$ be a PIR scheme with database communication complexity $c_{\mathcal{D}}(k)$. For every interactive Turing machine $\mathcal{U}'$, every reconstruction algorithm $\mathcal{R}'$, every $r_{\mathcal{U}'}$, and every k, let*

$$p \stackrel{def}{=} \text{Prob}\,[\, x = x_1 \circ \cdots \circ x_k \leftarrow \{0,1\}^k;\ (r_{\mathcal{D}}, t) \leftarrow t_{\mathcal{D},\mathcal{U}'}((1^k, x), \cdot, 1^k, r_{\mathcal{U}'});\ j \leftarrow [k] : \mathcal{R}'(1^k, r_{\mathcal{U}'}, t, j) \neq x_j\,]$$

Then it holds that $H(p) \geq \frac{k - c_{\mathcal{D}}(k)}{k}$, where $H(p)$ is the binary entropy function $H(p) \stackrel{def}{=} p\log(1/p) + (1-p)\log(1/(1-p))$.

Proof. We need to prove that, for every $\mathcal{U}'$ and $\mathcal{R}'$, after running $(\mathcal{D}, \mathcal{U}')$ with a uniform data string for $\mathcal{D}$, the probability that $\mathcal{R}'$ fails in reconstructing a data bit in a uniformly chosen location j, has binary entropy which is bounded below by $\frac{k - c_{\mathcal{D}}(k)}{k}$. This is proved using standard information theory arguments (e.g., similar arguments have been used in [3]). For background and terminology used in the proof below, see for example [9].

Let X be the random variable ranging over the data strings (where X_j corresponds to the j-th bit), and A be the random variable ranging over the database answers. Thus, the length of A is at most $c_{\mathcal{D}}(k)$, implying that $H(A) \leq c_{\mathcal{D}}(k)$ (where H is the entropy function for random variables). Let $\hat{X} \in \{0,1\}^k$ denote the user's reconstruction of the data string X, namely (following the notation in

[5] Indeed, if $\mathcal{U}'$ had known which location j he would have to reconstruct, he could run the honest user algorithm $\mathcal{U}$ with input j, and could reconstruct the correct bit with high probability using the reconstruction function $\mathcal{R}$.

the lemma), $\hat{X}_j = \mathcal{R}'(1^k, r_{\mathcal{U}'}, t, j)$ for $j \in [k]$. Let $p_j \stackrel{\text{def}}{=} \text{Prob}[\hat{X}_j \neq X_j]$ be the probability of failure in reconstructing the j-th bit. The probability of failure in reconstructing a *random* bit-location is therefore $p = (1/k) \cdot \sum_{j=1}^{k} p_j$. By Fano's inequality (see [9]), we have that $H(p_j) \geq H(X_j|A)$, for all $j = 1, \ldots, k$, where $H(p_j)$ refers to the binary entropy function, and $H(X_j|A)$ is the entropy of X_j given A. By the chain rule for entropy,

$$H(X|A) = \sum_{j=1}^{k} H(X_j|A, X_{j-1}, \ldots X_1) \leq \sum_{j=1}^{k} H(X_j|A)$$

On the other hand,

$$H(X|A) = H(X) - H(A) + H(A|X) = k - H(A) \geq k - c_{\mathcal{D}}(k),$$

where the last equality follows since A is determined by X. Putting all the above together and using the concavity of the entropy function, we obtain that

$$H(p) = H(\frac{1}{k}\sum_{j=1}^{k} p_j) \geq \frac{1}{k}\sum_{j=1}^{k} H(p_j) \geq \frac{1}{k}\sum_{j=1}^{k} H(X_j|A) \geq \frac{H(X|A)}{k} \geq \frac{k - c_{\mathcal{D}}(k)}{k}$$

□

Remark 1. Note that Lemma 1 holds even when $c_{\mathcal{D}}(k)$ is defined as the *expected* database communication complexity (rather than the worst-case one). This is because the proof above holds for any $c_{\mathcal{D}}(k) \geq H(A)$, and indeed the expected length of A is bounded below by the entropy of A (according to the entropy bound on data compression [9]).

The relation between the failure probability p and its binary entropy is given by the following fact (the proof follows from the expression for the entropy function and is omitted).

Fact 1 *For every $\epsilon > 0$ there exists a constant $c > 0$ such that for every $0 \leq p < c$, $p\log(1/p) \leq H(p) \leq (1+\epsilon)p\log(1/p)$.*

The above fact allows us to translate the lower bound on $H(p)$ into a lower bound on p. For example, a loose manipulation of the fact yields that, for any $\delta > 0$ and small enough p, $p > H(p)^{1+\delta}$. More generally, if $H(p)$ is non-negligible then p is also non-negligible. For sake of concreteness, we state a corollary bounding the failure probability, using $\delta = 1$. This will be sufficient for our needs, although as explained tighter corollaries can be derived.

Corollary 1. *Let $\mathcal{P} = (\mathcal{D}, \mathcal{U}, \mathcal{R})$ be a PIR scheme with database communication complexity $c_{\mathcal{D}}(k)$. The there exists a constant $c > 0$ such that for every interactive Turing machine $\mathcal{U}'$, every reconstruction algorithm $\mathcal{R}'$, every $r_{\mathcal{U}'}$, and every k, letting p be as in Lemma 1, we have that either $p > c$, or $p \geq (1 - c_{\mathcal{D}}(k)/k)^2$.*

Thus, if the communication complexity $c_{\mathcal{D}}(k) < k$, the probability that the user fails to reconstruct a bit in a random location after a single execution is non-negligible. For example, if $c_{\mathcal{D}}(k) = k - 1$ this failure probability is at least $1/poly(k)$, and if $c_{\mathcal{D}}(k) \leq k/2$ the failure probability is constant.

Finally, recall that in our protocol Alice and Bob run m independent invocations of $(\mathcal{D}, \mathcal{U})$, and (since Bob is honest-but-curious), $I_{\bar{c}} = (i_{\bar{c}}^1, \ldots, i_{\bar{c}}^m)$ is a uniformly chosen m-tuple, independent of the random choices made in the PIR invocations. Moreover, Bob is able to reconstruct $b_{\bar{c}}$ if and only if he can reconstruct the exclusive-or of all values $x^1(i_{\bar{c}}^1) \oplus \cdots \oplus x^m(i_{\bar{c}}^m)$, since he receives $z_{\bar{c}}$ from Alice in step 4. This, together with Corollary 1, yields that for an appropriately chosen polynomial number m, the failure probability is exponentially close to 1, namely Bob's probability of correctly reconstructing $b_{\bar{c}}$ is negligible. We conclude that our protocol maintains privacy against honest-but-curious Bob.

We have proved that the protocol of Figure 1 maintains correctness, privacy against Alice, and privacy against honest-but-curious Bob. We have therefore proved the following theorem.

Theorem 1. *If there exists a single database PIR scheme with database communication complexity $c_{\mathcal{D}}(k) < k$, where k is the length of the database, then there exists an honest-Bob-$\binom{2}{1}$-OT protocol with security parameter k.*

Similarly, it is easy to see that using a PIR scheme for which the data privacy requirement holds with respect to honest databases (rather than maliciously ones) in the protocol of Figure 1 yields an $\binom{2}{1}$-OT protocol for which both privacy requirement hold with respect to honest Alice and Bob.

Theorem 2. *If there exists a honest-database PIR scheme with database communication complexity $c_{\mathcal{D}}(k) < k$, where k is the length of the database, then there exists an honest-parties-$\binom{2}{1}$-OT protocol with security parameter k.*

The following remarks about the full strength of Theorem 1 follow from the proof above.

Round and Communication Complexity. Our protocol for honest-Bob-$\binom{2}{1}$-OT requires the same number of rounds as the underlying PIR protocol $\mathcal{P}$, and in particular if $\mathcal{P}$ has one round, so is the new protocol. This is so, since all the messages that need to be sent by Bob (in steps 1,3 of our protocol) can be computed in parallel and sent to Alice in a single message, and similarly all messages that need to be sent back by Alice (in steps 1,4) can be sent to Bob in a single message. We also note that our theorem holds even when we consider *expected* communication complexity (rather than maximal).

Computational Power of the Parties. Our transformation from PIR to honest-Bob-$\binom{2}{1}$-OT preserves the computational power of the parties; namely, if $\mathcal{D}$ (resp., $\mathcal{U}$) runs in polynomial time, then so does Alice (resp., Bob). In terms of privacy, our result is stronger than stated in Theorem 1; namely, the privacy against the honest-but-curious Bob is information-theoretic (to see this, observe that in the proof of this property we never make any assumption on the computational power of Bob, but rather rely on Lemma 1 which is information-theoretic).

On the other hand, the privacy against Alice requires the same assumptions as on the computational power of $\mathcal{D}$ in the PIR protocol $(\mathcal{D}, \mathcal{U})$; however, notice that Alice *must* be computationally bounded, since there exists no single database PIR protocol with communication complexity smaller than the size of the database and private against a computationally unbounded database [8].

Our Reduction. We note that our construction is a *black-box* reduction in the following sense: the $\binom{2}{1}$-OT uses the underlying PIR protocol as a subroutine with the only guarantee that the total number of bits that user gets regarding the database is strictly less then the total size of the database (i.e., without relying on any specific features of the implementation, and without making any additional assumptions about the implementation.) Thus any idealized implementation of this primitive (as a black-box) will also work for our purposes. As a consequence, our reduction is also "code-to-code". That is, any *implementation* of non-trivial Single-Database PIR protocol will also give an implementation of OT. In this aspect, our reduction is similar to [4].

4 PIR Implies $\binom{2}{1}$-OT (even for dishonest parties)

In this section, we transform the protocol given in Figure 1 into a protocol that is resilient against arbitrary (possibly dishonest) parties. That is, we prove the following analogue of Theorem 1.

Theorem 3. *If there exists a single database PIR scheme with database communication complexity $c_{\mathcal{D}}(k) < k$, where k is the length of the database, then there exists an $\binom{2}{1}$-OT protocol with security parameter k. Moreover, if the original PIR scheme requires a constant number of rounds then so does the resulting $\binom{2}{1}$-OT protocol.*

Proof. Let $\mathcal{P}$ be a PIR scheme with database communication $c_{\mathcal{D}}(k) < k$. Theorem 1 guarantees an implementation of $\binom{2}{1}$-OT for honest-but-curious Bob. Such an implementation may be transformed into one for dishonest parties, using (by now standard) techniques originating in [18, 19], based on commitment schemes and zero-knowledge proofs for NP-complete languages. The resulting reduction, however, would return a protocol for $\binom{2}{1}$-OT having a number of rounds polynomial in k even if the original PIR scheme has a constant number of rounds. Below we sketch a more direct reduction, combining ideas in [19] with techniques for witness-indistinguishability protocols from [14], which yields a constant round $\binom{2}{1}$-OT whenever $\mathcal{P}$ is a constant round PIR.

Let us denote by (Alice, Bob) the $\binom{2}{1}$-OT scheme obtained applying Theorem 1 to $\mathcal{P}$. In order to achieve privacy against a possibly dishonest Bob, it is enough to design the scheme so that the following two properties are satisfied: (1) the two m-tuples of indices $(i_0^1, \ldots, i_0^m)$ and $(i_1^1, \ldots, i_1^m)$ are uniformly and independently distributed over $[n]^m$; (2) Bob's messages during the execution of the PIR subprotocols are computed according to the specified program, and using randomness that is independently distributed from the above two m-tuple of indices. In order to achieve the first property, the two m-tuples are computed

using a flipping coin subprotocol at the beginning of protocol (Alice,Bob). In order to achieve the second property, at the beginning of the protocol Bob commits to the randomness to be later used while running the PIR subprotocol. Specifically, the protocol (Alice,Bob) is modified as follows.

At the beginning of protocol (Alice, Bob):

1. Bob commits to a sufficiently random string R and to randomly chosen indices $(l_0^1, \ldots, l_0^m)$ and $(l_1^1, \ldots, l_1^m)$ by sending three commitment keys com_R, com_0, com_1;
2. Alice sends random indices $(h_0^1, \ldots, h_0^m)$ and $(h_1^1, \ldots, h_1^m)$;
3. Bob sets $i_d^j = (h_d^j + l_d^j \bmod n) + 1$, for $j = 1, \ldots, m$ and $d = 0, 1$;

When required to use indices $(i^1, \ldots, i^m)$ in step 1 of (Alice, Bob), for each message he sends:

4. Bob proves that the message has been correctly computed according to the PIR subprotocol, using the string R committed in step 1 above as random tape, and using as a tuple of indices one of the two m-tuples committed in step 1 above. This can be written as an NP statement and can be efficiently reduced to a membership statement T for an NP complete language. Bob proves T to Alice by using a witness-indistinguishable proof system.

When required to send indices $(i_d^1, \ldots, i_d^m)$, for $d = 0, 1$, in step 3 of (Alice, Bob):

5. Bob proves that the two tuples he is sending have been correctly computed in the following sense: one is the same used in the PIR subprotocols and one is the one out of the two committed in step 1 above not used in the PIR subprotocols. This can be written as an NP statement and can be efficiently reduced to a membership statement T for an NP complete language. Bob proves T to Alice by using a witness-indistinguishable proof system.

We note that the parallel execution of an atomic zero-knowledge proof system for an NP-complete language as the one in [18] is known to be witness-indistinguishable from results in [14] and can be implemented using only 3 rounds of communication, and therefore can be used in steps 4 and 5 above.

Now, let us briefly show that the modified protocol (Alice, Bob) is a $\binom{2}{1}$-OT protocol. First of all, observe that the described modification does not affect the property of correctness, which therefore continues to hold. Then observe that the fact that the privacy against Alice continues to hold follows from the witness-indistinguishability of the proof system used, and the privacy against a possibly dishonest Bob follows from the soundness of the proof system used. Moreover, the overall number of rounds of the modified protocol (Alice, Bob) is constant and no additional complexity assumption is required, since commitment schemes and 3-round witness-indistinguishable proof systems for NP complete languages can be implemented using any one-way function [20, 30] and one-way functions, in turn, can be obtained by any low-communication PIR protocol [3]. □

We remark that in the case $c(k) < k/2$ the above transformation can be made more efficient (by a polynomial factor) using a direct derivation of commitment

schemes from low communication PIR, provided in [3]. Finally, using Theorem 2 and the same techniques as above, Theorem 3 can be strengthened to transform even an honest-database PIR into a $\binom{2}{1}$-OT protocol; that is:

Theorem 4. *If there exists a single database honest-database PIR scheme with database communication complexity $c_{\mathcal{D}}(k) < k$, where k is the length of the database, then there exists an $\binom{2}{1}$-OT protocol with security parameter k.*

5 PIR Implies SPIR

We are now ready to complete the proof of the following theorem.

Theorem 5. *If there exists a single database PIR scheme with communication complexity $c(n) < n$, where n is the length of the database, then there exists a single database SPIR scheme with security parameter k and communication complexity $c(n) \cdot q(k)$ for some polynomial q.*

Proof. First, by the result of Naor and Pinkas [31], we know that given a family of pseudo-random functions, a $\binom{2}{1}$-OT primitive, and a single database PIR with communication complexity $c(n)$, there exists a single database SPIR protocol which uses $\log n$ invocations of $\binom{2}{1}$-OT , and additional communication complexity $c(n \cdot poly(k))$ where n is the length of the data string and k is the security parameter. Next, since PIR implies one-way functions (first proved in [3] and also directly follows from the results in the previous section), PIR also implies pseudo-random functions [17, 20]. Finally, by our result in the previous section, PIR implies $\binom{2}{1}$-OT (where the communication complexity is some polynomial $poly'$ in the security parameter). Thus, we get that PIR implies SPIR with communication complexity $c'(n)$, satisfying $c'(n) = c(n \cdot poly(k)) + poly'(k) \log n = poly''(k) \cdot c(n)$, where $poly, poly', poly''$ are polynomials, k is a security parameter, and n is the length of the database. The second equality uses the fact that $c(n) > \log n$, which follows from a result proven in [3], namely that in PIR where the database sends less than n bits, the user must send at least $\log n$ bits of communication. □

Acknowledgments. We thank Amos Beimel, Yevgeniy Dodis, Oded Goldreich, Yuval Ishai and Silvio Micali for useful comments.

References

1. A. Ambainis. Upper Bound on the Communication Complexity of Private Information Retrieval. In *Proc. of 24th ICALP*, 1997.
2. G. Brassard, C. Crepeau and J.-M. Robert All-or-Nothing Disclosure of Secrets In *Crypto '86* Springer-Verlag, 1987, pp. 234-238.
3. A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-Way Functions are Essential for Single-Server Private Information Retrieval. In *Proc. of 31st STOC*, 1999.

4. A. Beimel, T. Malkin, S. Micali. The All-or-Nothing Nature of Two-Party Secure Computation. In *Proc. of CRYPTO 99*, 1999.
5. C. Cachin, C. Crepeau, and S. Marcil. Oblivious Transfer with a Memory Bounded Receiver. In *Proc. of 39th FOCS*, 1998.
6. C. Cachin, S. Micali, and M. Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. In *Proc. of EUROCRYPT 99*, 1999.
7. B. Chor and N. Gilboa. Computationally Private Information Retrieval. In *Proc. of 29th STOC*, 1997.
8. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. In *Proc. of 36th FOCS*, 1995.
9. T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.
10. C. Crépeau. Equivalence Between Two Flavors of Oblivious Transfers. In *Proc. of CRYPTO '87*, 1987.
11. A. De Santis and P. Persiano, Zero-Knowledge Proofs of Knowledge without Interaction, In *Proc. of 33rd FOCS*, 1992.
12. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Universal Service-Providers for Database Private Information Retrieval. In *Proc. of 17th PODC*, 1998.
13. S. Even, O. Goldreich, and A. Lempel. A Randomized Protocol for Signing Contracts. *Comm. of ACM*, 28:637–647, 1985.
14. U. Feige and A. Shamir, Witness Indistinguishable and Witness Hiding Protocols. In *Proc. of 23rd STOC*, 1990.
15. Y. Gertner, S. Goldwasser, and T. Malkin. A Random Server Model for Private Information Retrieval. In *Proc. of 2nd RANDOM*, 1998.
16. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting Data Privacy in Private Information Retrieval Schemes. In *Proc. of 30th STOC*, 1998.
17. O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. In *Journal of the ACM*, vol. 38, 1991, pages 792–807, 1986.
18. O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing but Their Validity, and a Methodology of Cryptographic Protocol Design. In *Journal of the ACM*, vol. 38, 1991, pages 691–729.
19. O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game. In *Proc. of 19th STOC*, 1987.
20. J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. A Pseudorandom Generator From Any One-Way Function. *SIAM J. on Computing*, 1999, vol. 28, n. 4.
21. R. Impagliazzo and M. Luby, "One-Way Functions are Essential for Complexity-Based Cryptography" In *Proc. of FOCS 89.*
22. R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. In *Proc. of 21st STOC*, 1989.
23. Y. Ishai and E. Kushilevitz. Improved Upper Bounds on Information-Theoretic Private Information Retrieval. In *Proc. of 31st STOC*, 1999.
24. J. Kilian. Founding Cryptography on Oblivious Transfer. In *Proc. of 20th STOC*, 1988.
25. J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and Completness In Private Computations. In *SIAM J. on Computing*, includes [26].
26. E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and Completness in Multi-party Private Computations. In *Proc. of 35th FOCS 94*, 1994.
27. E. Kushilevitz and R. Ostrovsky. Replication is Not Needed: Single-Database Computationally Private Information Retrieval. In *Proc. of 38th FOCS*, 1997.

28. E. Kushilevitz and R. Ostrovsky. One-Way Trapdoor Permutations are Sufficient for Single-database Computationally-Private Information Retrieval. In *Proc. of EUROCRYPT '00*, 2000.
29. E. Mann. Private Access to Distributed Information. Master's thesis, Technion - Israel Institute of Technology, Haifa, 1998.
30. M. Naor. Bit Commitment Using Pseudorandom Generators. *Journal of Cryptology*, 4:151–158, 1991.
31. M. Naor and B. Pinkas. Oblivious Transfer and Polynomial Evaluation. In *Proc. of 31st STOC*, 1999.
32. M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. In *Proc. of 21st STOC*, 1989.
33. R. Ostrovsky and V. Shoup. Private Information Storage. In *Proc. of 29th STOC*, 1997.
34. R. Ostrovsky and A. Wigderson One-Way Functions are Essential for Non-Trivial Zero-Knowledge. In *Proceedings of the Second Israel Symposium on Theory of Computing and Systems* (ISTCS-93) pp. 1-10., IEEE 1993.
35. M. O. Rabin. How to Exchange Secrets by Oblivious Transfer. Technical Report TR-81, Harvard University, 1981.
36. J. Rompel. One-Way Functions are Necessary and Sufficient for Secure Signatures. In *Proc. of 22nd STOC*, 1990.
37. J. P. Stern. A New and Efficient All-or-Nothing Disclosure of Secrets Protocol. In *Proc. of ASIACRYPT '98*, 1998.

Authenticated Key Exchange Secure Against Dictionary Attacks

Mihir Bellare[1], David Pointcheval[2], and Phillip Rogaway[3]

[1] Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA, mihir@cs.ucsd.edu, WWW home page: http://www-cse.ucsd.edu/users/mihir

[2] Dépt. d'Informatique–CNRS, École Normale Supérieure, 45 rue d'Ulm, 75230 Paris Cedex 05, France, david.pointcheval@ens.fr, WWW home page: http://www.dmi.ens.fr/~pointche

[3] Dept. of Computer Science, University of California at Davis, Davis, CA 95616, USA, rogaway@cs.ucdavis.edu, WWW home page: http://www.cs.ucdavis.edu/~rogaway

Abstract. Password-based protocols for authenticated key exchange (AKE) are designed to work despite the use of passwords drawn from a space so small that an adversary might well enumerate, off line, all possible passwords. While several such protocols have been suggested, the underlying theory has been lagging. We begin by defining a model for this problem, one rich enough to deal with password guessing, forward secrecy, server compromise, and loss of session keys. The one model can be used to define various goals. We take AKE (with "implicit" authentication) as the "basic" goal, and we give definitions for it, and for entity-authentication goals as well. Then we prove correctness for the idea at the center of the Encrypted Key-Exchange (EKE) protocol of Bellovin and Merritt: we prove security, in an ideal-cipher model, of the two-flow protocol at the core of EKE.

1 Introduction

THE PROBLEM. This paper continues the study of password-based protocols for authenticated key exchange (AKE). We consider the scenario in which there are two entities—a client A and a server B—where A holds a password pw and B holds a key related to this. The parties would like to engage in a conversation at the end of which each holds a session key, sk, which is known to nobody but the two of them. There is present an active adversary $\mathcal{A}$ whose capabilities include enumerating, off-line, the words in a dictionary D, this dictionary being rather likely to include pw. In a protocol we deem "good" the adversary's chance to defeat protocol goals will depend on how much she interacts with protocol participants—it won't significantly depend on her off-line computing time.

The above protocol problem was first suggested by Bellovin and Merritt [6], who also offer a protocol, Encrypted Key Exchange (EKE), and some informal security analysis. This protocol problem has become quite popular, with further

papers suggesting solutions including [7, 15–18, 22, 10, 11, 10, 21]. The reason for this interest is simple: password-guessing attacks are a common avenue for breaking into systems, and here is a domain where good cryptographic protocols can help.

CONTRIBUTIONS. Our first goal was to find an approach to help manage the complexity of definitions and proofs in this domain. We start with the model and definitions of Bellare and Rogaway [4] and modify or extend them appropriately. The model can be used to define the execution of authentication and key-exchange protocols in many different settings. We specify the model in pseudo-code, not only in English, so as to provide succinct and unambiguous execution semantics. The model is used to define the ideas of proper partnering, freshness of session keys, and measures of security for authenticated key exchange, unilateral authentication, and mutual authentication. Some specific features of our approach are: partnering via session IDs (an old idea of Bellare, Petrank, Rackoff, and Rogaway—see Remark 1); a distinction between accepting a key and terminating; incorporation of a technical correction to [4] concerning Test queries (this arose from a counter-example by Rackoff—see Remark 5); providing the adversary a separate capability to obtain honest protocol executions (important to measure security against dictionary attacks); and providing the adversary corruption capabilities which enable a treatment of forward secrecy.

We focus on AKE (with no explicit authentication). Philosophically, AKE is more "basic" than a goal like mutual authentication (MA). Pragmatically, AKE is simpler and takes fewer flows (two instead of three). Earlier work [3] began by defining MA and then embellishing the definition to handle an associated key exchange. Protocol development followed the same course. That approach gets complicated when one adds in the concern for password-guessing security.

Under our approach resistance to dictionary attacks is just a question of advantage vs. resource expenditure. It shows up in theorems, not definitions (once the model is adequately refined). A theorem asserting security of some protocol makes quantitative how much *computation* helps and just how much *interaction* does. One sees whether or not one has security against dictionary attacks by looking to see if maximal adversarial advantage grows primarily with the ratio of interaction to the size of the password space.

In Section 4 we define EKE2, which is essentially the pair of flows at the center of Bellovin and Merritt's Diffie-Hellman based Encrypted Key Exchange protocol [6]. We show that EKE2 is a secure AKE protocol, in the ideal-cipher model. Security here entails forward secrecy.

RELATED WORK. Recently people have been trying to get this area onto firmer foundations. The approach has been to build on the ideas of Bellare and Rogaway [3, 4], extending their definitions to deal with dictionary attacks. Lucks [17] was the first work in this vein. Halevi and Krawczyk [14] provide definitions and protocols for password-based unilateral authentication (UA) in the model in which the client holds the public key for the server, a problem which is different from, but related to, the one we are considering. Some critiques of [14] are made by [9], who also give their own, simulation-based notion for password-based UA.

In contemporaneous work to ours MacKenzie and Swaminathan [18], building on [3, 14], give definitions and proofs for a password-based MA protocol, and then a protocol that combines MA and AKE. Boyko, MacKenzie and Patel, building on [1, 20], give definitions and a proof for a Diffie-Hellman based protocol. In both papers the authors' motivation is fundamentally the same as our own: to have practical and provably secure password-based protocols.

ONGOING WORK. In [5] we provide a simple AKE protocol for the asymmetric trust model: the client holds pw and the server holds $f(pw)$, where f is a one-way function. If the adversary corrupts the server she must still expend time proportional to the quality of the password. We are working on the analysis. We are also investigating the security of EKE2 when its encryption function $\mathcal{E}$ is instantiated by $\mathcal{E}_{pw}(x) = x \cdot H(pw)$ where H is a random oracle and the arithmetic is in the underlying group.

2 Model

The model described in this section is based on that of [3, 4]. In particular we take from there the idea of modeling instances of principals via oracles available to the adversary; modeling various kinds of attacks by appropriate queries to these oracles; having some notion of partnering; and requiring semantic security of the session key via Test queries.

PROTOCOL PARTICIPANTS. We fix a nonempty set ID of *principals*. Each principal is either a *client* or a *server*: ID is the union of the finite, disjoint, nonempty sets $Client$ and $Server$. Each principal $U \in ID$ is named by a string, and that string has some fixed length. When $U \in ID$ appears in a protocol flow or as an argument to a function, we mean to the string which names the principal.

LONG-LIVED KEYS. Each principal $A \in Client$ holds some password, pw_A. Each server $B \in Server$ holds a vector $pw_B = \langle pw_B[A] \rangle_{A \in Client}$ which contains an entry per client. Entry $pw_B[A]$ is called the *transformed-password*. In a protocol for the *symmetric* model $pw_A = pw_B[A]$; that is, the client and server share the same password. In a protocol for the *asymmetric* model, $pw_B[A]$ will typically be chosen so that it is hard to compute pw_A from A, B, and $pw_B[A]$. The password pw_A (and therefore the transformed password $pw_B[A]$) might be a poor one. Probably some human chose it himself, and then installed $pw_B[A]$ at the server. We call the pw_A and pw_B long-lived keys (LL-keys).

Figure 1 specifies how a protocol is run. It is in *Initialization* that pw_A and pw_B arise: everybody's LL-key is determined by running a *LL-key generator*, PW. A simple possibility for PW is that the password for client A is determined by $pw_A \xleftarrow{R} PW_A$, for some finite set PW_A, and $pw_B[A]$ is set to pw_A. Notice that, in Figure 1, PW takes a superscript h, which is chosen from space Ω. This lets PW's behavior depend on an idealized hash function. Different LL-key generators can be used to capture other settings, like a public-key one.

EXECUTING THE PROTOCOL. Formally, a protocol is just a probabilistic algorithm taking strings to strings. This algorithm determines how instances of the

Initialization	$H \xleftarrow{R} \Omega$; $\langle pw_A, pw_B \rangle_{A \in Client,\ B \in Server} \xleftarrow{R} PW^h()$ **for** $i \in \mathsf{N}$ and $U \in ID$ **do** $state^i_U \leftarrow$ READY; $acc^i_U \leftarrow term^i_U \leftarrow used^i_U \leftarrow$ FALSE $sid^i_U \leftarrow pid^i_U \leftarrow sk^i_U \leftarrow$ UNDEF
Send $(U,\ i,\ M)$	$used^i_U \leftarrow$ TRUE; **if** $term^i_U$ **then return** INVALID $\langle msg\text{-}out, acc, term^i_U, sid, pid, sk, state^i_U \rangle \leftarrow$ $P^h(\langle U, pw_U, state^i_U, M \rangle)$ **if** acc **and** $\neg acc^i_U$ **then** $sid^i_U \leftarrow sid$; $pid^i_U \leftarrow pid$; $sk^i_U \leftarrow sk$; $acc^i_U \leftarrow$ TRUE **return** $\langle msg\text{-}out, sid, pid, acc, term^i_U \rangle$
Reveal $(U,\ i)$	**return** sk^i_U
Corrupt (U, pw)	**if** $U \in Client$ **and** $pw \neq$ DONTCHANGE **then** **for** $B \in Server$ **do** $pw_B[U] = pw[B]$ **return** $\langle pw_U,\ \{state^i_U\}_{i \in \mathsf{N}} \rangle$
Execute (A, i, B, j)	**if** $A \notin Client$ **or** $B \notin Server$ **or** $used^i_A$ **or** $used^j_B$ **then return** INVALID $msg\text{-}in \leftarrow B$ **for** $t \leftarrow 1$ **to** ∞ **do** $\langle msg\text{-}out, sid, pid, acc, term_A \rangle \xleftarrow{R}$ Send $(A,\ i,\ msg\text{-}in))$ $\alpha_t \leftarrow \langle msg\text{-}out, sid, pid, acc, term_A \rangle$ **if** $term_A$ **and** $term_B$ **then return** $\langle \alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \alpha_t \rangle$ $\langle msg\text{-}out, sid, pid, acc, term_B \rangle \xleftarrow{R}$ Send $(B,\ j,\ msg\text{-}in))$ $\beta_t \leftarrow \langle msg\text{-}out, sid, pid, acc, term_B \rangle$ **if** $term_A$ **and** $term_B$ **then return** $\langle \alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \alpha_t, \beta_t \rangle$
Test $(U,\ i)$	$sk \xleftarrow{R} SK$; $b \xleftarrow{R} \{0,1\}$; **if** $\neg term^i_U$ **then return** INVALID **if** $b = 1$ **then return** sk^i_U **else return** sk
Oracle (M)	**return** $h(M)$

Fig. 1. The model. The protocol is P, the LL-key generator is PW, and the session-key space SK. Probablity space Ω depends on the model of computation.

principals behave in response to signals (messages) from their enviornment. It is the adversary who sends these signals. As with the LL-key generator, P may depend on h.

Adversary $\mathcal{A}$ is a probabilistic algorithm with a distinguished query tape. Queries written on this tape are answered as specified in Figure 1. The following English-language description may clarify what is happening.

During the exeuction there may be running many *instances* of each principal $U \in ID$. We call instance i of principal U an *oracle*, and we denote it Π^i_U. Each instance of a principal might be embodied as a process (running on some machine) which is controlled by that principal.

A client-instance speaks first, producing some first message, *Flow1*. A server-instance responds with a message of its own, *Flow2*, intended for the client-instance which sent *Flow1*. This process is intended to continue for some fixed

number of flows (usually 2–5), until both instances have *terminated*. By that time each instance should have *accepted*, holding a particular *session key* (SK), *session id* (SID), and *partner id* (PID). Let us describe these more fully.

At any point in time an oracle may *accept*. When an oracle accepts it holds a session key sk, a session id sid, and a partner id pid. Think of these values as having been written on a write-only tape. The SK is what the instance was aiming to get. It can be used to protect an ensuing conversation. The SID is an identifier which can be used to uniquely name the ensuing session. It is also useful definitionally. The PID names the principal with which the instance believes it has just exchanged a key. The SID and PID aren't secret—indeed we will hand them to the adversary—but the SK certainly is. A client-instance and a server-instance can accept at most once.

Remark 1. In this paper we use session IDs as our approach to defining partnering. This idea springs from discussions in 1995 among Bellare, Petrank, Rackoff, and Rogaway. In [3] the authors define partnering via "matching conversations," while in [4] the authors define partnering by way of an existentially guaranteed partnering function. Though all three approaches are reasonable, the use of matching-conversations can be criticized as focussing on a syntactic element that is ultimately irrelevant, while partnering via an existentially-guarateed partnering function allows for some unintuitive partnering functions. An explicit SID seems an elegant way to go. Specification document defining "real" protocols (eg., SSL and IPSec) typically do have SIDs, and in cases where an SID was not made explicit one can readilly define one (eg., by the concatenation of all protocol flows). □

Remark 2. We emphasize that accepting is different from terminating. When an instance terminates, it is done—it has what it wants, and won't send out any further messages. But an instance may wish to accept now, and terminate later. This typically happens when an instance believes it is now holding a good session key, but, prior to using that key, the instance wants confirmation that its desired communication partner really exists, and is also holding that same session key. The instance can accomplish this by accepting now, but waiting for a confirmation message to terminate. The distinction between terminating and accepting may at first seem artificial, but the distinction is convenient and it is typical of real MA protocols. It can be seen as an "asymmetry-breaking device" for dealing with the well-known issue that the party who sends the last flow is never sure if it was received. □

Our communications model places the adversary at the center of the universe. The adversary $\mathcal{A}$ can make queries to any instance: she has an endless supply of Π_U^i oracles ($U \in ID$ and $i \in \mathsf{N}$). There are all together six types of queries that $\mathcal{A}$ can make. The responses to these queries are specified in Figure 1. We now explain the capability that each kind of query captures.

(1) $\mathsf{Send}\,(U,\, i,\, M)$ — This sends message M to oracle Π_U^i. The oracle computes what the protocol says to, and sends back the response. Should the oracle accept,

this fact, as well as the SID and PID, will be made visible to the adversary. Should the oracle terminate, this too will be made visible to the adversary. To initiate the protocol with client A trying to enter into an exchange with server B the adversary should send message $M = B$ to an unused instance of A. A Send-query models the real-world possibility of an adversary $\mathcal{A}$ causing an instance to come into existence, for that instance to receive communications fabricated by $\mathcal{A}$, and for that instance to respond in the manner prescribed by the protocol.

(2) Reveal $(U,\ i)$ — If oracle Π_U^i has accepted, holding some session key sk, then this query returns sk to the adversary. This query models the idea (going back to Denning and Sacco [12]) that loss of a session key shouldn't be damaging to other sessions. A session key might be lost for a variety of reasons, including hacking, cryptanalysis, and the prescribed-release of that session key when the session is torn down.

(3) Corrupt (U, pw) — The adversary obtains pw_U and the states of all instances of U (but see Remark 3). This query models the possibility of subverting a principal by, for example, witnessing a user type in his password, installing a "Trojan horse" on his machine, or hacking into a machine. Obviously this is a very damaging type of query. Allowing it lets us deal with forward secrecy and the extent of damage which can be done by breaking into a server. A Corrupt query directed against a client U may also be used to replace the value of $pw_B[U]$ used by server B. This is the role of the second argument to Corrupt. Including this capability allows a dishonest client A to try to defeat protocol aims by installing a strange string as a server B's transformed password $pw_B[A]$.

(4) Execute (A, i, B, j) — Assuming that client oracle Π_A^i and server oracle Π_B^j have not been used, this call carries out an honest execution of the protocol between these oracles, returning a transcript of that execution. This query may at first seem useless since, using Send queries, the adversary already has the ability to carry out an honest execution between two oracles. Yet the query is essential for properly dealing with dictionary attacks. In modeling such attacks the adversary should be granted access to plenty of honest executions, since collecting these involves just passive eavesdropping. The adversary is comparatively constrained in its ability to actively manipulate flows to the principals, since bogus flows can be auditied and punative measures taken should there be too many.

(5) Test $(U,\ i)$ — If Π_U^i has accepted, holding a session key sk, then the following happens. A coin b is flipped. If it lands $b = 0$, then sk is returned to the adversary. If it lands $b = 1$, then a random session key, drawn from the distribution from which session keys are supposed to be drawn, is returned. This type of query is only used to measure adversarial success—it does not correspond to any actual adversarial ability. You should think of the adversary asking this query just once.

(6) Oracle (M) — Finally, we give the adversary oracle access to a function h, which is selected at random from some probability space Ω. As already remarked, not only the adversary, but the protocol and the LL-key generator may depend on h. The choice of Ω determines if we are woking in the *standard* model, *ideal-hash* model, or *ideal-cipher* model. See the discussion below.

Remark 3. As described in Figure 1, a Corrupt query directed against U releases the LL-key pw_U and also the current state of all instances of U. We call this the "strong-corruption model." A weaker type of Corrupt query returns *only* the LL-key of that principal. We call this the "weak-corruption model." The weak-corruption model corresponds to acquiring a principal's password by coaxing it out of him, as opposed to completely compromising his machine. □

Remark 4. Notice that a Corrupt query to U does not result in the release of the session keys owned by U. The adversary already has the ability to obtain session keys through Reveal queries, and releasing those keys by a Corrupt query would make forward secrecy impossible. □

Remark 5. Soon after the appearance of [4], Rackoff [19] came up with an example showing how the definition given in that paper was not strong enough to guarantee security for certain applications using the distributed session key. The authors of [4] traced the problem to a simple issue: they had wrongly made the restriction that the Test query be the adversary's last. Removal of this restriction solved the problem. This minor but important change in the definition of [4], made in 1995, has since been folklore in the community of researchers in this area, and is explicitly incorporated into our current work. □

STANDARD MODEL, IDEAL-HASH MODEL, IDEAL-CIPHER MODEL. Figure 1 refers to probability space Ω. We consider three possiblities for Ω, giving rise to three different models of computation.

In the **standard model** Ω is the distribution which puts all the probability mass on one function: the constant function which returns the empty-string, ε, for any query M. So in the standard model, all mention of h can be ignored.

Fix a finite set of strings $\mathcal{C}$. In the **ideal-hash model** (also called the random-oracle model) choosing a random function from Ω means choosing a random function h from $\{0,1\}^*$ to $\mathcal{C}$. This models the use of a cryptographic hash function which is so good that, for purposes of analysis, one prefers to think of it as a public random function.

Fix finite sets of strings $\mathcal{G}$ and $\mathcal{C}$ where $|\mathcal{G}| = |\mathcal{C}|$. In the **ideal-cipher model** choosing a random function h from Ω amounts to giving the protocol (and the adversary) a perfect way to encipher strings in $\mathcal{G}$: namely, for $K \in \{0,1\}^*$, we set $\mathcal{E}_K \colon \mathcal{G} \to \mathcal{C}$ to be a random one-to-one function, and we let $\mathcal{E}_K \colon \{0,1\}^* \to \mathcal{G}$ be defined by $\mathcal{D}_K(y)$ is the value x such that $\mathcal{E}_K(x) = y$, if $y \in \mathcal{C}$, and BAD otherwise. We let $h(\mathsf{encrypt}, K, M) = \mathcal{E}_K(M)$ and $h(\mathsf{decrypt}, K, C) = \mathcal{D}_K(C)$. The capabilities of the ideal-hash model further include those of the ideal-cipher model, by means of a query $h(\mathsf{hash}, x)$ which, for shorthand, we denote $H(x)$.

The ideal-cipher model is very strong (even stronger than the ideal-hash model) and yet there are natural and apparently-good ways to instantiate an ideal cipher for use in practical protocols. See [8]. Working in this model does not render trivial the goals that this paper is interested in, and it helps make for protocols that don't waste any bits. A protocol will always have a clearly-indicated model of computation for which it is intended so, when the protocol is fixed, we do not make explicit mention of the model of compuation.

The *basic* notion of freshness (no requirement for forward secrecy):
if [RevealTo $(U,\ i)$] **or** [RevealToPartnerOf $(U,\ i)$] **or** [SomebodyWasCorrupted] **then** unfresh **else** fresh
A notion of freshness the incorporates a requirement for *forward secrecy*:
if [RevealTo $(U,\ i)$] **or** [RevealToPartnerOf $(U,\ i)$] **or** [SomebodyWasCorruptedBeforeTheTestQuery **and** Manipulated(U, i)] **then** fs-unfresh **else** fs-fresh

Fig. 2. Session-key freshness. A Test query is made to oracle Π_U^i. The chart specifies how, at the end of the execution, the session key of that oracle should be regarded (fresh or unfresh, and fs-fresh or fs-unfresh). Notation is described in the accompanying text.

Remark 6. The ideal-cipher model is richer than the RO-model, and you can't just say "apply the Feistel construction to your random oracle to make the cipher." While this may be an approach to instantiating an ideal-cipher, there is no formal sense we know in which you can simulate the ideal-cipher model using only the RO-model. □

3 Definitions

Our definitional approach is from [4], but adaptations must be made since partnerning is defined in a different manner than in [4] (as discussed in Section 2), and since we now consider forward secrecy as one of our goals.

PARTNERING USING SIDS. Fix a protocol P, adversary $\mathcal{A}$, LL-key generator PW, and session-key space SK. Run P in the manner specified in Section 2. In this execution, we say that oracles Π_U^i and $\Pi_{U'}^{i'}$ are **partnered** (and each oracle is said to be a partner of the other) if both oracles accept, holding (sk, sid, pid) and (sk', sid', pid') respectively, and the following hold:

(1) $sid = sid'$ and $sk = sk'$ and $pid = U'$ and $pid' = U$.

(2) $U \in$ *Client* and $U' \in$ *Server*, or $U \in$ *Server* and $U' \in$ *Client*.

(3) No oracle besides Π_U^i and $\Pi_{U'}^{i'}$ accepts with a PID of pid.

The above definition of partnering is quite strict. For two oracles to be partners with one another they should have the same SID and the same SK, one should be a client and the other a server, each should think itself partnered with the other, and, finally, no third oracle should have the same SID. Thus an oracle that has accepted will have a single partner, if it has any partner at all.

TWO FLAVORS OF FRESHNESS. Once again, run a protocol with its adversary. Suppose that the adversary made exactly one Test query, and it was to Π_U^i. Intuitively, the oracle Π_U^i should be considered unfresh if the adversary may know the SK contained within it.

In Figure 2 we define two notions of freshness—with and witout forward secrecy (fs). Here is the notation used in that figure. We say "RevealTo $(U,\ i)$"

is true iff there was, at some point in time, a query $\mathsf{Reveal}\,(U,\ i)$. We say "RevealToPartnerOf $(U,\ i)$" is true iff there was, at some point in time, a query $\mathsf{Reveal}\,(U',\ i')$ and $\Pi_{U'}^{i'}$ is a partner to Π_U^i. We say "SomebodyWasCorrupted" is true iff there was, at some point in time, a query $\mathsf{Corrupt}\,(U', pw)$ for some U', pw. We say "SomebodyWasCorruptedBeforeTheTestQuery " is true iff there was a $\mathsf{Corrupt}\,(U', pw)$ query and this query was made before the $\mathsf{Test}\,(U,\ i)$ query. We say that "Manipulated(U, i)" is true iff there was, at some point in time, a $\mathsf{Send}\,(U,\ i,\ M)$ query, for some string M.

EXPLANATION. In our definition of security we will be "giving credit" to the adversary $\mathcal{A}$ if she specifies a fresh (or fs-fresh) oracle and then correctly identifies if she is provided the SK from that oracle or else a random SK. We make two cases, according to whether or not "forward secrecy" is expected. Recall that forward secrecy entails that loss of a long-lived key should not compromise already-distributed session keys.

Certainly an adversary can know the SK contained within an oracle Π_U^i if she did a Reveal query to Π_U^i, or if she did a Reveal query to a partner of Π_U^i. This accounts for the first two disjuncts in each condition of Figure 2. The question is whether or not a Corrupt query may divulges the SK. Remember that a Corrupt query does actually return the SK, but it does return the LL-key. For the "basic" notion of security (fresh/unfresh) we pessimistically assume that a Corrupt query does reveal the session key, so any Corrupt query makes all oracles unfresh. (One could tighten this a little, if desired.) For the version of the definition with forward secrecy a Corrupt query may reveal a SK only if the Corrupt query was made *before* the Test query. We also require that the Test query was to an oracle that was the target of a Send query (as opposed to an oracle that was used in an Execute query). (Again, this can be tightened up a little.) This acts to build in the following requirement: that even *after* the Corrupt query, session keys exchanged by principals who behave honestly are *still* fs-fresh. This is a nice property, and since it seems to always be achieved in protocols which achieve forward secrecy, we have lumped it into that notion. This was done amounts to saying that an "honest" oracle—one that is used only for an Execute call—is always fs-fresh, even if there is a Corrupt query. (Of course you still have to exclude the the possiblity that the oracle was the target of a Reveal query, or that its partner was.)

Remark 7. Forward secrecy, in the strong-corruption model, is not achievable by two-flow protocols. The difficulty is the following. A two-flow protocol is client-to-server then server-to-client. If the client oracle is corrupted after the server oracle has terminated but before the client oracle has received the response, then the server oracle will be fs-fresh but the adversary can necessarilly compute the shared SK since the adversary has the exact same information that the client oracle would have had the client oracle received the server oracle's flow.

One way around this is to go to the weak-corruption model. A second way around this is to add a third flow to the protocol. A final way around this is to define a slightly weaker notion of forward secrecy, *weak* forward-secrecy, in which an oracle is regarded as "wfs-unfresh" if it fs-unfresh, or the test query is to a

manipulated oracle, that oracle is unpartnered at termination, and somebody gets corrupted. Otherwise the oracle is wfs-fresh. □

AKE SECURITY (WITH AND WITHOUT FORWARD SECRECY). In a protocol execution of $P, PW, SK, \mathcal{A}$ we say that $\mathcal{A}$ *wins*, in the AKE sense, if she asks a single Test-query, Test (U, i), where Π_U^i has terminated and is fresh, and $\mathcal{A}$ outputs a single bit, b', and $b' = b$ (where b is the bit selected during the Test query). The ake advantage of $\mathcal{A}$ in attacking (P, PW, SK) is twice the probability that $\mathcal{A}$ wins, minus one. (The adversay can trivially win with probability 1/2. Multiplying by two and subtracting one simply rescales this probability.) We denote the **ake advantage** by $\mathsf{Adv}^{\mathrm{ake}}_{P,PW,SK}(\mathcal{A})$.

We similarly define the **ake-fs advantage**, $\mathsf{Adv}^{\text{ake-fs}}_{P,PW,SK}(\mathcal{A})$, where now one insists that the oracle Π_U^i to which the Test-query is directed be fs-fresh.

AUTHENTICATION. In a protocol execution of $P, PW, SK, \mathcal{A}$, we say that an adversary violates client-to-server authentication if some server oracle terminates but has no partner oracle. We let the **c2s advantage** be the probablity of this event, and denote it by $\mathsf{Adv}^{\mathrm{c2s}}_{P,PW,SK}(\mathcal{A})$. We say that an adversary violates server-to-client authentication if some client oracle terminates but has no partner oracle. We let the **s2c advantage** be the probability of this event, and denote it by $\mathsf{Adv}^{\mathrm{s2c}}_{P,PW,SK}(\mathcal{A})$. We say that an adversary violates mutual authentication if some oracle terminates, but has no partner oracle. We let the **ma advantage** denote the probablity of this event, and denote it by $\mathsf{Adv}^{\mathrm{ma}}_{P,PW,SK}(\mathcal{A})$.

MEASURING ADVERSARIAL RESOURCES. We are interested in an adversary's maximal advantage in attacking some protocol as a function of her resources. The resources of interest are:

- t — the adversary's running time. By convention, this includes the amount of space it takes to describe the adversary.
- q_{se}, q_{re}, q_{co}, q_{ex}, q_{or} — these count the number of Send, Reveal, Corrupt, Execute, and Oracle queries, respectively.

When we write $\mathsf{Adv}^{\mathrm{ake}}_{P,PW,SK}(\mathit{resources})$, overloading the Adv-notation, it means the maximal possible value of $\mathsf{Adv}^{\mathrm{ake}}_{P,PW,SK}(\mathcal{A})$ among all adversaries that expend at most the specified resources. By convention, the time to sample in PW (one time) and to sample in SK (one time) are included in $\mathsf{Adv}_{P,PW,SK}(\mathit{resources})$ (for each type of advantage).

DIFFIE-HELLMAN ASSUMPTION. We will prove security under the computational Diffie-Hellman assumption. The concrete version of relevance to us is the following. Let $G = \langle g \rangle$ be a finite group. We assume some fixed representation for group elements, and implicitly switch between group elements and their string representations. Let $\mathcal{A}$ be an adversary that outputs a list of group elements, $z_1, \ldots, z_q$. Then we define

$$\mathsf{Adv}^{\mathrm{dh}}_{G}(\mathcal{A}) = \Pr[x, y \leftarrow \{1, \ldots, |G|\} : \; g^{xy} \in \mathcal{A}(g^x, g^y)], \text{ and}$$
$$\mathsf{Adv}^{\mathrm{dh}}_{G}(t, q) = \max_{\mathcal{A}} \{ \; \mathsf{Adv}^{\mathrm{dh}}_{G,g}(\mathcal{A}) \; \} \; ,$$

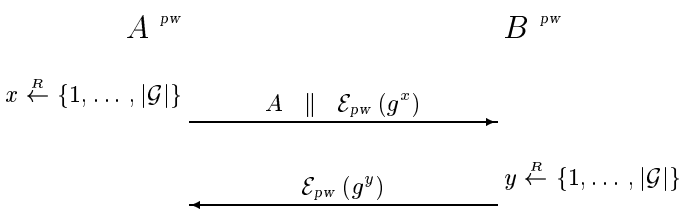


Fig. 3. The protocol EKE2. Depicted are flows of an honest execution. The shared session key is $sk = H(A \,\|\, B \,\|\, g^x \,\|\, g^y \,\|\, g^{xy})$ and shared session ID is $sid = A \,\|\, \mathcal{E}_{pw}(g^x) \,\|\, B \,\|\, \mathcal{E}_{pw}(g^y)$. The partner ID for A is $pid_A = B$ and the partner ID for B is $pid_B = A$.

where the maximum is over all adversaries that run in time at most t and output a list of q group elements. As before, t includes the description size of adversary $\mathcal{A}$.

4 Secure AKE: Protocol EKE2

In this section we prove the security of the two flows at the center of Bellovin and Merritt's EKE protocol [6]. Here we define the (slightly modified) "piece" of EKE that we are interested in.

DESCRIPTION OF EKE2. This is a Diffie-Hellman key exchange in which each flow is enciphered by the password, the SK is $sk = H(A \,\|\, B \,\|\, g^x \,\|\, g^y \,\|\, g^{xy})$, and the SID and PID are appropriately defined. The name of the sender also accompanies the first flow. See Figures 3 and 4.

Arithmetic is in a finite cyclic group $\mathcal{G} = \langle g \rangle$. This group could be $\mathcal{G} = \mathbb{Z}_p^\star$, or it could be a prime-order subgroup of this group, or it could be an elliptic curve group. We denote the group operation multiplicatively. The protocol uses a cipher $\mathcal{E} : \mathit{Password} \times \mathcal{G} \to \mathcal{C}$, where $pw_A \in \mathit{Password}$ for all $A \in \mathit{Client}$. There are many concrete constructions that could be used to instantiate such an object; see [8]. In the analysis this is treated as an ideal cipher. Besides the cipher we use a hash function H. It outputs ℓ-bits, where ℓ is the length of the session key we are trying to distribute. Accordingly, the session-key space SK associated to this protocol is $\{0,1\}^\ell$ equipped with a uniform distribution.

SECURITY THEOREM. The following indicates that the security of EKE2 is about as good as one could hope for. We consider the simple case where *Password* has size N and all client passwords are chosen uniformly (and independently) at random from this space. Formally this initialization is captured by defining the appropriate LL-key generator PW. It picks $pw_A \xleftarrow{R} \mathit{Password}$ for each $A \in \mathit{Client}$ and sets $pw_B[A] = pw_A$ for each $B \in \mathit{Server}$ and $A \in \mathit{Client}$. It then sets $pw_B = \langle pw_B[A] \rangle_{A \in Client}$ and outputs $\langle pw_A, pw_B \rangle_{A \in Client,\ B \in Server}$. The theorem below assumes that the space *Password* is known in the sense that it is possible to sample from it efficiently.

if $state = \text{READY}$ **and** $U \in Client$ **then** // A sends the first flow
  $\langle A \rangle \leftarrow U$ $\langle B \rangle \leftarrow msg\text{-}in$, where $B \in Server$
  $x \xleftarrow{R} \{1, \ldots, |G|\}$ $X \leftarrow g^x$ $X^* \leftarrow \mathcal{E}_{pw}(X)$ $msg\text{-}out \leftarrow A \parallel X^*$
  $sid \leftarrow pid \leftarrow sk \leftarrow \varepsilon$ $acc \leftarrow term \leftarrow \text{FALSE}$ $state \leftarrow \langle x, B \rangle$
  return $(msg\text{-}out, acc, term, sid, pid, sk, state)$

else if $state = \text{READY}$ **and** $U \in Server$ **then** // B sends the second flow
  $\langle B \rangle \leftarrow U$ $\langle A, X^* \rangle \leftarrow msg\text{-}in$, where $A \in Client$ and X^* is a ciphertext
  $y \xleftarrow{R} \{1, \ldots, |G|\}$ $Y \leftarrow g^y$ $Y^* \leftarrow \mathcal{E}_{pw}(Y)$
  $X \leftarrow \mathcal{D}_{pw}(X^*)$ $K \leftarrow X^y$ $msg\text{-}out \leftarrow Y^*$
  $sid \leftarrow A \parallel X^* \parallel B \parallel Y^*$ $pid \leftarrow A$ $sk \leftarrow H(A \parallel B \parallel X \parallel Y \parallel K)$
  $acc \leftarrow term \leftarrow \text{TRUE}$ $state \leftarrow \text{DONE}$
  return $(msg\text{-}out, acc, term, sid, pid, sk, state)$

else if $state = \langle x, B \rangle$ **and** $U \in Client$ **then** // A receives the second flow
  $\langle Y^* \rangle \leftarrow msg\text{-}in$, where Y^* is a ciphertext
  $Y \leftarrow \mathcal{D}_{pw}(Y^*)$ $K \leftarrow Y^x$
  $sid \leftarrow A \parallel X^* \parallel B \parallel Y^*$ $pid \leftarrow B$ $sk \leftarrow H(A \parallel B \parallel X \parallel Y \parallel K)$
  $acc \leftarrow term \leftarrow \text{TRUE}$ $state \leftarrow \text{DONE}$
  return $(msg\text{-}out, acc, term, sid, pid, sk, state)$

Fig. 4. Definition of EKE2. The above defines both client and server behavior, $P^h(\langle U, pw, state, msg\text{-}in \rangle)$.

Theorem 1. *Let* $q_{\text{se}}, q_{\text{re}}, q_{\text{co}}, q_{\text{ex}}, q_{\text{or}}$ *be integers and let* $q = q_{\text{se}} + q_{\text{re}} + q_{\text{co}} + q_{\text{ex}} + q_{\text{or}}$. *Let Password be a finite set of size* N *and assume* $1 \leq N \leq \sqrt{|\mathcal{G}|}/q$. *Let* PW *be the associated LL-key generator as discussed above. Let* P *be the EKE2 protocol and let* SK *be the associated session-key space. Assume the weak-corruption model. Then*

$$\mathsf{Adv}^{\text{ake-fs}}_{P,PW,SK}(t, q_{\text{se}}, q_{\text{re}}, q_{\text{co}}, q_{\text{ex}}, q_{\text{or}}) \leq \frac{q_{\text{se}}}{N} + q_{\text{se}} \cdot q_{\text{or}} \cdot \mathsf{Adv}^{\text{dh}}_{\mathcal{G},g}(t', q_{\text{or}}) + \frac{O(q^2)}{|\mathcal{G}|} + \frac{O(1)}{\sqrt{|\mathcal{G}|}}$$

where $t' = t + O(q_{\text{se}} + q_{\text{or}})$. □

Remark 8. Since EKE2 is a two-flow protocol, Remark 7 implies that it cannot achieve forward secrecy in the strong-corruption model. Accordingly the above theorem considers the weak-corruption model with regard to forward secrecy. The resistance to dictionary attacks is captured by the first term which is the number of send queries divided by the size of the password space. The other terms can be made negligible by an appropriate choice of parameters for the group $\mathcal{G}$. □

Remark 9. The upper bound imposed in the theorem on the size N of the password space is not a restriction because if the password space were larger the question of dictionary attacks becomes moot: the adversary cannot exhaust the password space off-line anyway. Nonetheless it may be unclear why we require

such a restriction. Intuitively, as long as the password space is not too large the adversary can probably eliminate at most one candidate password from consideration per Send query, but for a larger password space it might in principle be able to eliminate more at a time. This doesn't damage the success probability because although it eliminates more passwords at a time, there are also more passwords to consider. □

The proof of Theorem 1 is omitted due to lack of space and can be found in the full version of this paper [2]. We try however to provide a brief sketch of the main ideas in the analysis.

Assume for simplicity there is just one client A and one server B. Consider some adversary $\mathcal{A}$ attacking the protocol. We view $\mathcal{A}$ as trying to guess A's password. We consider at any point in time a set of "remaining candidates." At first this equals *Password*, and as time goes on it contains those candidate passwords that the adversary has not been able to eliminate from consideration as values of the actual password held by A. We also define a certain "bad" event in the execution of the protocol with this adversary, and show that as long as this event does not occur, two things are true:

(1) A's password, from the adversary's point of view, is equally likely to be any one from the set of remaining passwords, and

(2) The size of the set of remaining passwords decreases by at most one with each oracle query, and the only queries for which a decrease occurs are reveal or test queries to manipulated oracles.

The second condition implies that the number of queries for which the decrease of size in the set of remaining candidates occurs is bounded by q_{se}. We then show that the probability of the bad event can be bounded in terms of the advantage function of the DH problem over $\mathcal{G}$.

Making this work requires isolating a bad event with two properties. First, whenever it happens we have a way to "embed" instances of the DH problem into the protocol so that adversarial success leads to our obtaining a solution to the DH problem. Second, absence of the bad event leads to an inability of the adversary to obtain information about the password at a better rate than eliminating one password per reveal or test query to a manipulated oracle. Bounding the probability of the bad event involves a "simulation" argument as we attempt to "plant" DH problem instances in the protocol. Bounding adversarial success under the assumption the bad event does not happen is an information-theoretic argument. Indeed, the difficulty of the proof is in choosing the bad event so that one can split the analysis into an information-theoretic component and a computational component in this way.

5 Adding Authentication

In this section we sketch generic transformations for turning an AKE protocol P' into a protocol P that provides client-to-server authentication, server-to-client authentication, or both. The basic approach is well-known in folklore—use

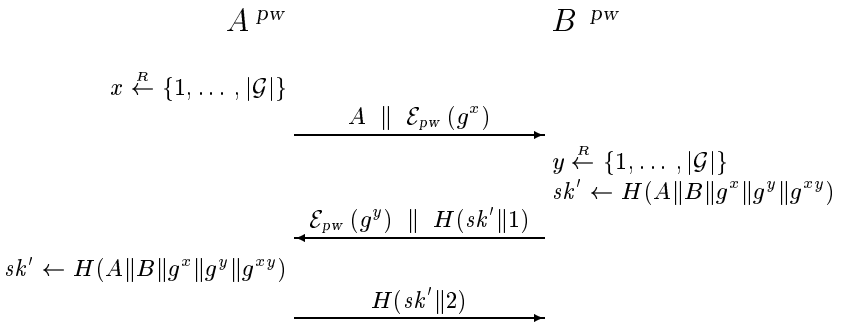


Fig. 5. Flows of an honest execution of AddMA(EKE2). The shared SK is $sk = H(sk' \,\|\, 0)$ and the shared SID is $sid = A \,\|\, \mathcal{E}_{pw}(g^x) \,\|\, B \,\|\, \mathcal{E}_{pw}(g^y)$. The PID for A is B and the PID for B is A.

the distributed session key to construct a simple "authenticator" for the other party—but one has to be careful in the details, and people often get them wrong.

The ease with which an AKE protocol can be modified to provide authentication is one of the reasons for using AKE as a starting point.

In what follows we assume that the AKE protocol P' is designed to distribute session keys from a space $SK = U_\ell$, the uniform distribution on ℓ-bit strings.

While a pseudorandom function is sufficient for adding authentication to an AKE protocol, for simplicity (and since one likely assumes it anyway, in any practical password-based AKE construction) we assume (at least) the random-oracle model. The random hash function is denoted H. Its argument (in our construction) will look like $sk' \,\|\, i$, where sk' is an ℓ-bit string and i is a fixed-length string encoding one of the numbers 0, 1, or 2. We require that the AKE protocol P never evaluates H at any point of the form $sk' \,\|\, 0$, $sk' \,\|\, 1$, or $sk' \,\|\, 2$, where $sk' \in \{0,1\}^\ell$.

THE TRANSFORMATIONS. The transformation AddCSA (add client-to-server authentication) works as follows. Suppose that in protocol P' the client A has accepted sk'_A, sid'_A, pid'_A, and suppose that A then terminates. In protocol $P = \text{AddCSA}(P')$ have A send one additional flow, $auth_A = H(sk'_A \,\|\, 2)$, have A accept $sk_A = H(sk'_A \,\|\, 0)$, $sid_A = sid'_A$, $pid_A = pid'_A$, and have A terminate, saving no state. On the server side, suppose that in P' the server B accepts sk'_B, sid'_B, pid'_B, and B terminates. In protocol P have B receive one more flow, $auth'_A$. Have B check if $auth'_A = H(sk'_B \,\|\, 2)$. If so, then B accepts $sk_B = H(sk'_B \,\|\, 0)$, $sid_B = sid'_B$, $pid_B = pid'_B$, and then B terminates, without saving any state. Otherwise, B terminates (rejecting), saving no state.

Transformations AddSCA (add server-to-client authentication) and AddMA (add mutual authentication) are analogous. The latter is illustrated in Figure 5. In all of these transformation, when a party ends up sending two consecutive flows, one can always collapse them into one.

Remark 10. It is crucial in these transformations that the SK produced by P' is not used both to produce an authenticator and as the final session key; if one does this, the protocol is easily seen to be insecure under our definitions. This is a common "error" in the design of authentication protocols. It was first discussed [3]. □

PROPERTIES. Several theorems can be pursued about how the security of P' relates to that of AddCSA(P'), AddSCA(P'), and AddMA(P'). These capture the following. If P' is good in the sense of $\mathsf{Adv}^{\mathrm{ake}}$ then AddCSA(P') is good in the sense of $\mathsf{Adv}^{\mathrm{ake}}$ and $\mathsf{Adv}^{\mathrm{c2s}}$. If P' is good in the sense of $\mathsf{Adv}^{\mathrm{ake}}$ then AddSCA(P') is good in the sense of $\mathsf{Adv}^{\mathrm{ake}}$ and $\mathsf{Adv}^{\mathrm{s2c}}$. If P' is good in the sense of $\mathsf{Adv}^{\mathrm{ake}}$ then AddMA(P') is good in the sense of $\mathsf{Adv}^{\mathrm{ake}}$, $\mathsf{Adv}^{\mathrm{s2c}}$, and $\mathsf{Adv}^{\mathrm{c2s}}$. The weak form of forward secrecy mentioned in Remark 7 is also interesting in connection with AddCSA and AddMA, since these transformations apparently "upgrade" good weak forward secrecy, $\mathsf{Adv}^{\mathrm{ake\text{-}wfs}}$, to good ordinary forward secrecy, $\mathsf{Adv}^{\mathrm{ake\text{-}fs}}$.

SIMPLIFICATIONS. The generic transformations given by AddCSA, AddSCA and AddMA do not always give rise to the most efficient method for the final goal. Consider the protocol AddMA(EKE2) of Figure 5. It would seem that the encryption in the second flow can be eliminated and one still has a good protocol for AKE with MA. However, we know of no approach towards showing such a protocol secure short of taking the first two flows of that protocol and showing that they comprise a good AKE protocol with server-to-client authentication, and then applying AddCSA transformation.

Given the complexity of proofs in this domain and the tremendous variety of simple and plausibly correct protocol variants, it is a major open problem in this area to find techniques which will let us deal with the myriad of possibilities, proving the correct ones correct, without necessitating an investment of months of effort to construct a "rigid" proof for each and every possibility.

Acknowledgments

We thank Charlie Rackoff for extensive discussions on the subject of session-key exchange over the last five years, and for his corrections to our earlier works. We thank Victor Shoup for useful comments and criticisms on this subject. We thank the Eurocrypt 2000 committee for their excellent feedback and suggestions.

Mihir Bellare is supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering. Phillip Rogaway is supported in part by NSF CAREER Award CCR-9624560. Much of Phil's work on this paper was carried out while on sabbatical in the Dept. of Computer Science, Faculty of Science, Chiang Mai University, Thailand.

References

1. M. Bellare, R. Canetti, and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. *Proc. of the 30th STOC.* ACM Press, New York, 1998.

2. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. Full version of this paper, available from `http://www-cse.ucsd.edu/users/mihir`
3. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. *CRYPTO '93*, LNCS 773, pages 232–249. Springer-Verlag, Berlin, 1994.
4. M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: the Three Party Case. *Proc. of the 27th STOC*. ACM Press, New York, 1995.
5. M. Bellare and P. Rogaway, work in progress.
6. S. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks. *Proc. of the Symposium on Security and Privacy*, pages 72–84. IEEE, 1992.
7. S. Bellovin and M. Merritt. Augmented Encrypted Key Exchange: A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise. *Proceedings of the 1st Annual Conference on Computer and Communications Security*, ACM, 1993.
8. J. Black and P. Rogaway. Ciphers with Arbitrary Finite Domains. Manuscript, 2000.
9. M. Boyarsky. Public-Key Cryptography and Password Protocols: The Multi-User Case. *Proceedings of the 6th Annual Conference on Computer and Communications Security*, ACM, 1999.
10. V. Boyko, P. MacKenzie, and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. *Eurocrypt 2000*, LNCS 1807, pages 157–172. Springer-Verlag, Berlin, 2000.
11. P. Buhler, T. Eirich, M. Steiner, and M. Waidner. Secure Password-Based Cipher Suite for TLS. Proceedings of Network and Distributed Systems Security Symposium. February 2000.
12. D. Denning and G. Sacco. Timestamps in Key Distribution Protocols. *Communications of the ACM*, 24, 1981, pp 533–536.
13. L. Gong, M. Lomas, R. Needham, and J. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, June 1993.
14. S. Halevi and H. Krawczyk. Public-Key Cryptography and Password Protocols. *ACM Transactions on Information and System Security*, Vol. 2, No. 3, pp. 230–268, August 1999. Earlier version in *Proc. of the 5th CCS* conference, ACM Press, New York, 1998.
15. D. Jablon. Strong Password-Only Authenticated Key Exchange. *ACM Computer Communications Review*, October 1996.
16. D. Jablon. Extended Password Key Exchange Protocols Immune to Dictionary Attacks. *Proc. of WET-ICE '97*, pp. 248–255. IEEE Computer Society, June 1997.
17. S. Lucks. Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys. *Proc. of the Security Protocols Workshop*, LNCS 1361. Springer-Verlag, Berlin, 1997.
18. P. MacKenzie and R. Swaminathan. Secure Authentication with a Short Secret. Manuscript. November 2, 1999. Earlier version as Secure Network Authentication with Password Identification. Submission to IEEE P1363a. August 1999. Available from `http://grouper.ieee.org/groups/1363/addendum.html`
19. C. Rackoff, private communication, 1995.
20. V. Shoup. On Formal Models for Secure Key Exchange. Theory of Cryptography Library Record 99-12, `http://philby.ucsd.edu/cryptolib/` and invited talk at ACM Computer and Communications Security conference, 1999.

21. M. Roe, B. Christianson, and D.Wheeler. Secure Sessions from Weak Secrets. Technical report from University of Cambridge and University of Hertfordshire. Manuscript, 1998.
22. T. Wu. The Secure Remote Password Protocol. *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pp. 97–111, 1998.

Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman

Victor Boyko[1], Philip MacKenzie[2], and Sarvar Patel[2]

[1] MIT Laboratory for Computer Science, boyko@theory.lcs.mit.edu
[2] Bell Laboratories, Lucent Technologies, {philmac, sarvar}@lucent.com

Abstract. When designing password-authenticated key exchange protocols (as opposed to key exchange protocols authenticated using cryptographically secure keys), one must not allow any information to be leaked that would allow verification of the password (a weak shared key), since an attacker who obtains this information may be able to run an *off-line dictionary attack* to determine the correct password. We present a new protocol called PAK which is the first Diffie-Hellman-based password-authenticated key exchange protocol to provide a formal proof of security (in the random oracle model) against both passive and active adversaries. In addition to the PAK protocol that provides mutual *explicit* authentication, we also show a more efficient protocol called PPK that is provably secure in the *implicit*-authentication model. We then extend PAK to a protocol called PAK-X, in which one side (the client) stores a plaintext version of the password, while the other side (the server) only stores a verifier for the password. We formally prove security of PAK-X, even when the server is compromised. Our formal model for password-authenticated key exchange is new, and may be of independent interest.

1 Introduction

Two entities, who only share a password, and who are communicating over an insecure network, want to authenticate each other and agree on a large session key to be used for protecting their subsequent communication. This is called the *password-authenticated key exchange* problem. If one of the entities is a user and the other is a server, then this can be seen as a problem in the area of *remote user access*. Many solutions for remote user access rely on cryptographically secure keys, and consequently have to deal with issues like key management, public-key infrastructure, or secure hardware. Many solutions that are password-based, like telnet or Kerberos, have problems that range from being totally insecure (telnet sends passwords in the clear) to being susceptible to certain types of attacks (Kerberos is vulnerable to off-line dictionary attacks [30]).

Over the past decade, many password-authenticated key exchange protocols that promised increased security have been developed, e.g., [8, 9, 19, 18, 28, 21, 22, 24, 29]. Some of these have been broken [26], and, in fact, only two very recent ones have been formally proven secure. The SNAPI protocol in [25] is proven secure in the random oracle model [5, 6, 14], assuming the security of RSA (and

also Decision Diffie-Hellman [11], when perfect forward secrecy is desired). The simple and elegant protocol in [3] is proven as secure as Decision Diffie-Hellman in a model that includes random oracles and ideal block ciphers. (Our protocol and the protocol of [3] were developed independently.)

We present a new password-authenticated key exchange protocol called **PAK** (Password Authenticated Key exchange), which provides perfect forward secrecy and is proven to be as secure as Decision Diffie-Hellman in the random oracle model. Compared to the protocol of [25], PAK (1) does not require the RSA assumption, (2) has fewer rounds, and (3) is conceptually simpler, with a simpler proof. Compared to the protocol of [3], PAK does not require an ideal block cipher assumption for security, but has a more complicated proof. (We note that the ideal block cipher assumption is used much less often in the literature than the random oracle assumption.) In the full paper [13], we also show how the security of PAK can be related to the Computational Diffie-Hellman problem.

In addition to PAK, we also show a more efficient 2 round protocol called PPK (Password-Protected Key exchange) that is provably secure in the implicit-authentication model. We then extend PAK to a protocol called PAK-X, in which one side (the client) stores a plaintext version of the password, while the other side (the server) only stores a verifier for the password. We formally prove security of PAK-X, even when the server is compromised. Security in this case refers to an attacker not being able to pose as a client after compromising the server; naturally, it would be trivial to pose as the server.

Our formal model for password-authenticated key exchange is new, and may be of independent interest. It is based on the formal model for secure key exchange by Shoup [27] (which follows the work of [2]), enhanced with notions of password authentication security from [20, 25]. This model is based on the multi-party simulatability tradition (e.g. [1]), in which one first defines an ideal system that models, using a trusted center, the service to be performed (in this case, password-authenticated key exchange), and then one proves that the protocol running in the real world is essentially equivalent to that ideal system.

2 Background

User Authentication: Techniques for user authentication are broadly based on one or more of the following categories: (1) what a user knows, (2) what a user is, or (3) what a user has. The least expensive and most convenient solutions for user authentication have been based on the first category, of "what a user knows," and that is what we will focus on in this paper.

In fact, we will focus on the harder problem of *remote user authentication*. The need for remote user authentication is greatly increasing, due mainly to the explosive growth of the Internet and other types of networks, such as wireless communication networks. In any of these environments, it is safest to assume that the underlying links or networks are insecure, and we should expect that a powerful adversary would be capable of eavesdropping, deleting and inserting messages, and also initiating sessions.

Now we consider the question, "What can a user know?" It is common knowledge that users cannot remember long random numbers, hence if the user is required to know a large secret key (either symmetric or private/public) then these keys will have to be stored on the user's system. Furthermore, keeping these secret requires an extra security assumption and introduces a new point of weakness. Even if a user is required to know some public but non-generic data, like the server's public key, this must be stored on the user's system and requires an extra assumption that the public key cannot be modified. In either case, (1) there is a significant increase in administration overhead because both secret and public keys have to be generated and securely distributed to the user's system and the server, and (2) this would not allow for users to walk up to a generic station that runs the authentication protocol and be able to perform secure remote authentication to a system that was previously unknown to that station (such as, perhaps, the user's home system).

To solve these problems one may wish to use a trusted third party, either on-line (as in Kerberos) or off-line (i.e., a certification authority). However, the fact that the third party is "trusted" implies another security requirement. Also, the users or servers must at some point interact with the third party before they can communicate remotely, which increases the overhead of the whole system. Naturally, if an organized and comprehensive PKI emerges, this may be less of a problem. Still, password-only protocols seem very inviting because they are based on direct trust between a user and a server, and do not require the user to store long secrets or data on the user's system. They are thus cheaper, more flexible, and less administration-intensive. They also allow for a generic protocol which can be pre-loaded onto users' systems.

Password-Authentication Protocols: Traditional password protocols are susceptible to *off-line dictionary attacks*: Many users choose passwords of relatively low entropy, so it is possible for the adversary to compile a dictionary of likely passwords. Obviously, we can't prevent the adversary from trying the passwords on-line, but such an attack can be made infeasible by simply placing a limit on the number of unsuccessful authentication attempts. On the other hand, an off-line search through the dictionary is quite doable. Here is an example an attack against a simple challenge-response protocol: The adversary overhears a challenge R and the associated response $f(P, R)$ that involves the password. Now she can go off-line and run through all the passwords P' from a dictionary of likely passwords, comparing the value $f(P', R)$ with $f(P, R)$. If one of the values matches the response, then the true password has been discovered.

A decade ago, Lomas et.al. [23] presented the first protocols which were resistant to these types of off-line dictionary attacks. The protocols assumed that the client had the server's public key and thus were not strictly password-only protocols. Other protocols for this scenario were developed in [19, 20, 12].

The EKE protocol [8] was the first password authenticated key exchange protocol that did not require the user to know the server's public key. The idea of EKE was to use the password to symmetrically encrypt the protocol messages of a standard key exchange (e.g., Diffie-Hellman [15]). Then an attacker

making a password guess could decrypt the symmetric encryption, but could not break the asymmetric encryption in the messages, and thus could not verify the guess. Following EKE, many password authenticated key exchange protocols were proposed [9, 19, 18, 28, 21, 22, 24, 29]. Some of these protocols were, in addition, designed to protect against server compromise, so that an attacker that was able to steal data from a server could not later masquerade as a user without having performed a dictionary attack.[1] All of these protocol proposals contained informal arguments for security. However, the fact that some versions of these protocols were subsequently shown to be insecure [26] should emphasize the importance of formal proofs of security.

Models for Secure Authentication and Key Exchange: Bellare and Rogaway [4] presented the first formal model of security for entity authentication and key exchange, for the symmetric two party case. In [7] they extend it to the three party case. Blake-Wilson et.al. [10] further extend the model to cover the asymmetric setting. Independently, [25] and [3] present extensions to the model to allow for password authentication. Halevi and Krawczyk [20] and Boyarsky [12] present models which include both passwords and asymmetric keys (since they deal with password-based protocols that also rely on server public keys).

Bellare, Canetti, and Krawczyk [2] present a different model for security of entity authentication and key exchange, based on the multi-party simulatability tradition [1]. Shoup [27] refines and extends their model. We present a further extension of [27] that includes password authentication.

3 Model

For our proofs, we extend the formal notion of security for key exchange protocols from Shoup [27] to password-authenticated key exchange. We assume that the adversary totally controls the network, a la [4].

Security for key exchange in [27] is defined using an ideal system, which describes the service (of key exchange) that is to be provided, and a real system, which describes the world in which the protocol participants and adversaries work. The ideal system should be defined such that an "ideal world adversary" cannot (by definition) break the security. Then, intuitively, a proof of security would show that anything an adversary can do in the real system can also be done in the ideal system, and thus it would follow that the protocol is secure in the real system.

3.1 Ideal system

Our ideal system follows [27], except for the addition of password authentication and a slight modification to explicitly handle mutual authentication. We assume that there is a set of (honest) *users*, indexed $i = 1, 2, \ldots$. Each user i may have

[1] Naturally, given the data from a server, an attacker could perform an off-line dictionary attack, since the server must know something to verify a user's password.

several *instances* $j = 1, 2, \ldots$. Then (i, j) refers to a given *user instance*. A user instance (i, j) is told the identity of its partner, i.e., the user it is supposed to connect to (or receive a connection from). An instance is also told its *role* in the session, i.e., whether it is going to *open* itself for connection, or whether it is going to *connect* to another instance.

There is also an *adversary* that may perform certain operations, and a *ring master* that handles these operations by generating certain random variables and enforcing certain global consistency constraints. Some operations result in a record being placed in a *transcript*.

The ring master keeps track of session keys $\{K_{ij}\}$ that are set up among user instances (as will be explained below, the key of an instance is set when that instance starts a session). In addition, the ring master has access to a random bit string R of some agreed-upon length (this string is not revealed to the adversary). We will refer to R as *the environment*. The purpose of the environment is to model information shared by users in higher-level protocols.

Since we deal with password authentication, and since passwords are not cryptographically secure, our system must somehow allow a non-negligible probability of an adversary successfully impersonating an honest user. We do this by including passwords explicitly in our model. We let π denote the function assigning passwords to pairs of users. To simplify notation, we will write $\pi[A, B]$ to mean $\pi[\{A, B\}]$ (i.e., $\pi[A, B]$ is by definition equivalent to $\pi[B, A]$).

The adversary may perform the following types of operations:

initialize user [Transcript: (`"initialize user"`, i, ID_i)]
The adversary assigns identity string ID_i to (new) user i. In addition, a random password $\pi[ID_i, ID_{i'}]$ is chosen by the ring master for each existing user i' (see the discussion below on the distribution from which these passwords are generated). The passwords are not placed in the transcript. This models the out-of-band communication required to set up passwords between users.

set password [Transcript: (`"set password"`, i, ID', π)]
The identity ID' is required to be new, i.e., not assigned to any user. This sets $\pi[ID_i, ID']$ to π and places a record in the transcript.
After ID' has been specified in a *set password* operation, it cannot be used in a subsequent *initialize user* operation.

initialize user instance [Transcript: (`"initialize user instance"`, i, j, $role(i, j), PID_{ij}$)]
The adversary assigns a user instance (i, j) a role (one of $\{\text{open}, \text{connect}\}$) and a user PID_{ij} that is supposed to be its partner. If PID_{ij} is not set to an identity of an initialized user, then we require that a *set password* operation has been previously performed for i and PID_{ij} (and hence there can be no future *initialize user* operation with PID_{ij} as the user ID).

terminate user instance [Transcript: (`"terminate user instance"`,i,j)]
The adversary specifies a user instance (i, j) to terminate.

test instance password
This is called with an instance (i, j) and a password guess π. The returned result is either `true` or `false`, depending on whether $\pi = \pi[ID_i, PID_{ij}]$. If

the result is `true`, then this query is called a *successful guess on* $\{ID_i, PID_{ij}\}$ (note that a successful guess on $\{A, B\}$ is also a successful guess on $\{B, A\}$). This query may only be asked once per user instance. The instance has to be initialized and not yet engaged in a session (i.e., no *start session* operation has been performed for that instance). Note that the adversary is allowed to ask a *test instance password* query on an instance that has been terminated. This query does not leave any records in the transcript.

start session [Transcript: (`"start session"`, i, j)]
The adversary specifies that a session key K_{ij} for user instance (i, j) should be constructed. The adversary specifies which *connection assignment* should be used. There are three possible connection assignments:

open for connection from (i', j')**.** This requires that $role(i, j)$ is "open," (i', j') has been initialized and has not been terminated, $role(i', j')$ is "connect," $PID_{ij} = ID_{i'}$, $PID_{i'j'} = ID_i$, no other instance is open for connection from (i', j'), and no *test instance password* operation has been performed on (i, j). The ring master generates K_{ij} randomly. We now say that (i, j) is open for connection from (i', j').

connect to (i', j')**.** This requires that $role(i, j)$ is "connect," (i', j') has been initialized and has not been terminated, $role(i', j')$ is "open," $PID_{ij} = ID_{i'}$, $PID_{i'j'} = ID_i$, (i', j') was open for connection from (i, j) after (i, j) was initialized and is still open for connection from (i, j), and no *test instance password* operation has been performed on (i, j). The ring master sets $K_{ij} = K_{i'j'}$. We now say that (i', j') is no longer open for connection.

expose. This requires that either PID_{ij} has not been assigned to an identity of an initialized user, or there has been a successful guess on $\{ID_i, PID_{ij}\}$. The ring master sets K_{ij} to the value specified by the adversary.

Note that the connection assignment is not recorded in the transcript.

application [Transcript: (`"application"`, $f, f(R, \{K_{ij}\})$)]
The adversary is allowed to obtain any information she wishes about the environment and the session keys. (This models leakage of session key information in a real protocol through the use of the key in, for example, encryptions of messages.) The function f is specified by the adversary and is assumed to be efficiently computable.

implementation [Transcript: (`"impl"`, *cmnt*)]
The adversary is allowed to put in an "implementation comment" which does not affect anything else in the ideal world. This will be needed for generating ideal world views that are equivalent to real world views, as will be discussed later.

For an adversary $\mathcal{A}^*$, $IdealWorld(\mathcal{A}^*)$ is the random variable denoting the transcript of the adversary's operations.

Discussion (password authentication): Our system correctly describes the ideal world of password authenticated key exchange. If two users successfully complete a key exchange, then the adversary cannot obtain the key or the password. This is modeled by the adversary not being allowed any *test instance password* queries on an instance after a successful key exchange. Our ideal model explicitly uses

(ring master generated) passwords, and an adversary can only obtain information about a password by issuing a *test instance password* query for an instance, signifying an impersonation attempt by the adversary against the key exchange protocol run by that instance. (One may think of this as modeling an adversary who attempts to log in to a server by sending a guessed password.)

We did not specify how the ring master chooses passwords for pairs of users. The simplest model would be to have a dictionary $\mathcal{D}$, which is a set of strings, and let all passwords be chosen uniformly and independently from that dictionary. To achieve the strongest notion of security, though, we can give the adversary all the power, and simply let her specify the distribution of the passwords as an argument to the *initialize user* operation (the specification of the distribution would be recorded in the transcript). The passwords of a user could even be dependent on the passwords of other users. We note that our proofs of security do not rely on any specific distribution of passwords, and would thus be correct even in the stronger model.

We also model the ability of the adversary to set up passwords between any users and herself, using the *set password* query. This can be thought of as letting the adversary set up rogue accounts on any computer she wishes, as long as those rogue accounts have different user IDs from all the valid users.

3.2 Real System with Passwords

Now we describe the real system in which we assume a password-authenticated key exchange protocol runs. Again, this is basically from [27], except that we do not concern ourselves with public keys and certification authorities, since all authentication is performed using shared passwords.

Users and user instances are denoted as in the ideal system. User instances are defined as state machines with implicit access to the user's ID, PID, and password (i.e., user instance (i, j) is given access to $\pi[ID_i, PID_{ij}]$). User instances also have access to private random inputs (i.e., they may be randomized). A user instance starts in some initial state, and may transform its state only when it receives a message. At that point it updates its state, generates a response message, and reports its status, either **"continue"**, **"accept"**, or **"reject"**, with the following meanings:

- **"continue":** the user instance is prepared to receive another message.
- **"accept":** the user instance (say (i, j)) is finished and has generated a session key K_{ij}.
- **"reject":** the user instance is finished, but has not generated a session key.

The adversary may perform the following types of operations:

initialize user [Transcript: $(\texttt{"initialize user"}, i, ID_i)$]
initialize user instance [Transcript: $(\texttt{"initialize user instance"}, i, j, role(i, j), PID_{ij})$]
set password [Transcript: $(\texttt{"set password"}, i, ID', \pi)$]

application [Transcript: ("application", $f, f(R, \{K_{ij}\})$)]
All above as in the ideal system.

deliver message [Transcript: ("impl", "message", i, j, *InMsg*, *OutMsg*, *status*)]
The adversary delivers *InMsg* to user instance (i, j). The user instance updates its state, and replies with *OutMsg* and reports *status*. If *status* is "accept", the record ("start session", i, j) is added to the transcript, and if *status* is "reject", the record ("terminate instance", i, j) is added to the transcript.

random oracle [Transcript: ("impl", "random oracle", $i, x, H_i(x)$)]
The adversary queries random oracle i on a binary string x and receives the result of the random oracle query $H_i(x)$. Note that we do not allow *application* operations to query random oracles H_i. In other words, we do not give higher-level protocols access to the random oracles used by the key exchange scheme (although a higher-level protocol could have its own random oracles). The adversary, however, does have access to all the random oracles.

For an adversary $\mathcal{A}$, *RealWorld*($\mathcal{A}$) denotes the transcript of the adversary's operations. In addition to records made by the operations, the transcript will include the random coins of the adversary in an *implementation* record ("impl", "coins", *coins*).

3.3 Definition of Security

The definition of security for key exchange given in [27] requires

1. **completeness**: for any real world adversary that faithfully delivers messages between two user instances with complimentary roles and identities, both user instances accept; and
2. **simulatability**: for every efficient real world adversary $\mathcal{A}$, there exists an efficient ideal world adversary $\mathcal{A}^*$ such that *RealWorld*($\mathcal{A}$) and *IdealWorld*($\mathcal{A}^*$) are computationally indistinguishable.

We will use this definition for password-authenticated key exchange as well.[2]

4 Explicit Authentication: The PAK Protocol

4.1 Preliminaries

Let κ and ℓ denote our security parameters, where κ is the "main" security parameter and can be thought of as a general security parameter for hash functions and secret keys (say 128 or 160 bits), and $\ell > \kappa$ can be thought of as a security

[2] We can do this because our ideal model includes passwords explicitly. If it did not, we would have to somehow explicitly state the probability of distinguishing real world from ideal world transcripts, given how many impersonation attempts the real world adversary has made.

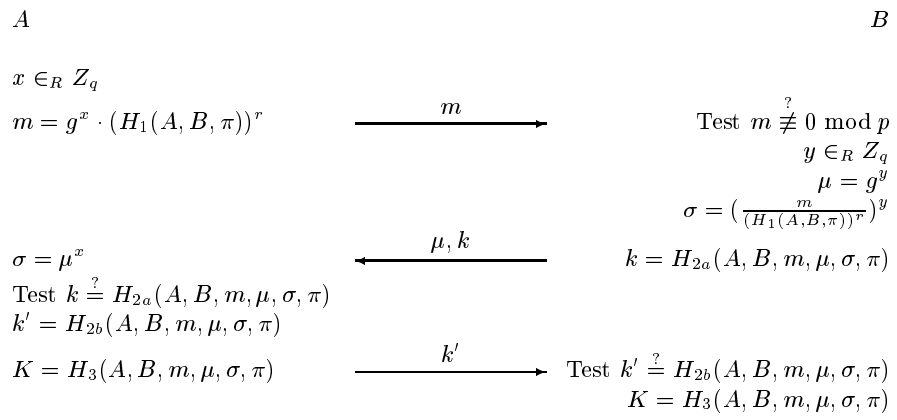


Fig. 1. PAK protocol, with $\pi = \pi[A,B]$. The resulting session key is K. If a "Test" returns false, the protocol is aborted.

parameter for discrete-log-based public keys (say 1024 or 2048 bits). Let $\{0,1\}^*$ denote the set of finite binary strings and $\{0,1\}^n$ the set of binary strings of length n. A real-valued function $\epsilon(n)$ is *negligible* if for every $c > 0$, there exists $n_c > 0$ such that $\epsilon(n) < 1/n^c$ for all $n > n_c$.

Let q of size at least κ and p of size ℓ be primes such that $p = rq + 1$ for some value r co-prime to q. Let g be a generator of a subgroup of Z_p^* of size q. Call this subgroup $G_{p,q}$. We will often omit "$\bmod p$" from expressions when it is obvious that we are working in Z_p^*.

Let $\mathrm{DH}(X,Y)$ denote the Diffie-Hellman value g^{xy} of $X = g^x$ and $Y = g^y$. We assume the hardness of the *Decision Diffie-Hellman problem* (DDH) in $G_{p,q}$. One formulation is that given g, X, Y, Z in $G_{p,q}$, where $X = g^x$ and $Y = g^y$ are chosen randomly, and Z is either $\mathrm{DH}(X,Y)$ or random, each with half probability, determine if $Z = \mathrm{DH}(X,Y)$. Breaking DDH implies a constructing a polynomial-time adversary that distinguishes $Z = \mathrm{DH}(X,Y)$ from a random Z with non-negligible advantage over a random guess.

4.2 The Protocol

Define hash functions $H_{2a}, H_{2b}, H_3 : \{0,1\}^* \to \{0,1\}^\kappa$ and $H_1 : \{0,1\}^* \to \{0,1\}^\eta$ (where $\eta \geq \ell + \kappa$). We will assume that H_1, H_{2a}, H_{2b}, and H_3 are independent random functions. Note that while H_1 is described as returning a bit string, we will operate on its output as a number modulo p.

The PAK protocol is given in Figure 1.

Theorem 1. *The PAK protocol is a secure password-authenticated key exchange protocol in the explicit-authentication model.*

Proof: *(Sketch)* The completeness requirement follows directly by inspection. Here we sketch the proof that the simulatability requirement holds. Complete

details are presented in the full paper [13]. The basic technique is essentially that of Shoup [27]. The idea is to create an ideal world adversary $\mathcal{A}^*$ by running the real world adversary $\mathcal{A}$ against a simulated real system, which is built on top of the underlying ideal system. In particular, $\mathcal{A}^*$ (i.e., the simulator combined with $\mathcal{A}$) will behave in the ideal world just like $\mathcal{A}$ behaves in the real world, except that idealized session keys will be used in the real world simulation instead of the actual session keys computed in the real system.

Thus, our proof consists of constructing a simulator (that is built on top of an ideal system) for a real system so that the transcript of an adversary attacking the simulator is computationally indistinguishable from the transcript of an adversary attacking the real system.

Simulator The general idea of our simulator is to try to detect guesses on the password (by examining the adversary's random oracle queries) and turn them into *test instance password* queries. If the simulator does not detect a password guess, then it either sets up a connection between two instances (if all the messages between them have been correctly relayed), or rejects (otherwise). The main difficulty in constructing the simulator is that we need to respond to the adversary's requests without knowing the actual passwords. This causes us to use random values in place of the results of those random oracle calls that take the password as an argument. We can think of these as "implicit" oracle calls. In handling the adversary's explicit random oracle queries, as well as those protocol operations that use random oracles, we need to make sure that we don't use inconsistent values for the result of a random oracle on a certain input. Specifically, we must make sure the random oracle queries to H_{2a} and H_{2b} are consistent with the k and k' values sent or received by the user instances. This is relatively straightforward (using *test instance password* queries) except when the adversary sends a μ value back to an initiator instance. To be able to determine the password being tested by the adversary in this case, we will make sure the simulator has answered each $H_1(A, B, \pi)$ query using a random value for which it knows the discrete log (after that value is raised to the r).

Indistinguishability The simulation described above is indistinguishable from the real world as long as the simulator does not need to perform a *test instance password* query that is disallowed in the ideal world. Specifically, by the rules of the ideal world, (1) only one of these queries can be made for each user instance, and (2) the query cannot be made at all for any instance that performs a *start session* operation (previously or in the future). So to finish our proof, we need to show that if the adversary can break either rule with non-negligible probability, then we can break the DDH Assumption with non-negligible probability.

The idea of the proof of (2) goes as follows. Say that the offending query is made within the first T queries. (T is bounded by the adversary's running time and must be polynomial.) Take a DDH challenge (X, Y, Z). Run the simulation (playing the ringmaster also, i.e., choosing our own passwords) with the following changes: Choose a random $d \in [0, T]$. On the dth *deliver message* query to initiate a protocol, say for users A and B, set $m = X$. For any B instance that receives

$m = X$, set $\mu = Yg^y$ for some random y. If the adversary makes a query to H_{2a}, H_{2b}, or H_3 with A, B, m, a μ as calculated above, σ, and π, where $\sigma = ZX^y/\mu^{r\alpha_\pi}$ for α_π the discrete log of $(H_1(A, B, \pi))^r$, guess that the DDH challenge is a true DH instance. All other queries are answered in a straightforward way, except that the adversary may make a valid password guess using its own μ and σ, for which the simulator cannot verify the σ value (because the simulator does not know the discrete log of X). In this case we flip a coin to decide whether to accept or not, and continue the simulation. It can be shown that if the adversary is able to break this ideal world rule with probability ϵ, then we will give a correct response to the DDH challenge with probability roughly $\frac{1}{2} + \frac{\epsilon}{4T}$. (The 4 in the denominator comes from the half probability of the DDH challenge being a true DH instance and the half probability of a correct coin flip.)

The idea of the proof of (1) goes as follows. Say that the offending queries occur within the first T queries. Let the DDH challenge be (X, Y, Z). Run the simulation (playing the ringmaster also) with the following changes: Choose a random $d \in [0, T]$. Assume the bad event will occur for the dth pair of users mentioned (either in an $H_1(A, B, \cdot)$ query or an *initialize user instance* with A and partner B) Each time $H_1(A, B, \pi)$ is queried for some π, flip a coin to decide whether to include a factor of X in the return value. For any first message to a B instance with partner A, set $\mu = Yg^y$ for some random y. Note that the σ values used in any pair of H_{2a}, H_{2b}, and H_3 queries for the same A, B, m, μ (where μ was calculated as Yg^y), and using two different password guesses (π_1 and π_2) can be tested against the Z value if exactly one of $H_1(A, B, \pi_1)$ and $H_1(A, B, \pi_2)$ included a factor of X in its calculation. If any of these pairs tests positively for the Z value, guess that the DDH challenge is a true DH instance. All other queries are answered in a straightforward way. It can be shown that if the adversary is able to break this ideal world rule with probability ϵ, then we will give a correct response to the DDH challenge with probability roughly $\frac{1}{2} + \frac{\epsilon}{4T}$. (The 4 in the denominator comes from the half probability of the DDH challenge being a true DH instance and the half probability of the adversary making queries for two passwords in which exactly one included a factor of X in the $H_1()$ calculation.) □

5 Implicit Authentication: The PPK Protocol

We first describe an Ideal System with Implicit Authentication, and then describe the PPK protocol. Note that we still use the Real System from Section 3.2.

5.1 Ideal System with Implicit Authentication

Here we consider protocols in which the parties are *implicitly* authenticated, meaning that if one of the communicating parties is not who she claims to be, she simply won't be able to obtain the session key of the honest party. The honest party (which could be playing the role of either `"open"` or `"connect"`) would still open a session, but no one would be able to actually communicate

with her on that session.[3] Thus, some of the connections may be "dangling." We will allow two new connection assignments:

dangling open. This requires $role(i, j)$ to be "open."
dangling connect. This requires $role(i, j)$ to be "connect."

In both cases, the ring master generates K_{ij} randomly.

To use implicit authentication with passwords, we will make the following rules:

- Dangling connection assignments are allowed even for instances on which the *test instance password* query has been performed.
- A *test instance password* query is allowed on an instance, even if it has already started a session with a dangling connection assignment.

We still restrict the number of *test instance password* queries to at most one per instance. The rules relating to other connection assignments do not change.

The reason for this permissiveness is that an instance with a dangling connection assignment can't be sure that it wasn't talking to the adversary. All that is guaranteed is that the adversary won't be able to get the key of that instance, unless she correctly guesses the password.

In practice, this means that we can't rule out an unsuccessful password guess attempt on an instance until we can confirm that some partner instance has obtained the same key. It follows that if we are trying to count the number of unsuccessful login attempts (e.g., so that we can lock the account when some threshold is reached), we can't consider an attempt successful until we get some kind of confirmation that the other side has obtained the same key. We thus see that key confirmation (which, in our model, is equivalent to explicit authentication) is indeed relevant when we use passwords.

5.2 PPK Protocol

If we don't require explicit authentication, we can make a much more efficient protocol. The PPK protocol requires only two rounds of communication. The protocol is given in Figure 2.

Theorem 2. *The PPK protocol is a secure password-authenticated key exchange protocol in the implicit-authentication model.*

The completeness requirement follows directly by inspection. The proof of simulatability is omitted due to page limits. The basic structure of the proof is very similar to that of the PAK protocol.

[3] In a later version of [27], Shoup also deals with implicit authentication, but in a different way. We feel our solution is more straightforward and intuitive.

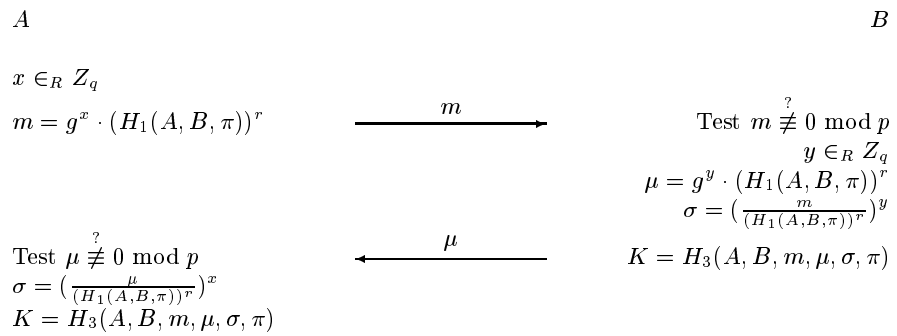


Fig. 2. PPK protocol, with $\pi = \pi[A,B]$. The resulting session key is K.

6 Resilience to Server Compromise—The PAK-X Protocol

6.1 Ideal System with Passwords—Resilience to Server Compromise

Now we define a system in which one party is designated as a server, and which describes the ability of an adversary to obtain information about passwords stored on the server, along with the resultant security. To accomplish this, one role (open or connect) is designated as the *server* role, while the other is designated as the *client* role. We add the *test password* and *get verifier* operations, and change the *start session* operation.

test password
: This query takes two users, say i and i', as arguments, along with a password guess π. If a *get verifier* query has been made on $\{i, i'\}$, then this returns whether $\pi = \pi[ID_i, ID_{i'}]$. If the comparison returns true, this is called a successful guess on $\{ID_i, ID_{i'}\}$. If no *get verifier* has been made on $\{i, i'\}$, then no answer is returned (but see the description of *get verifier* below).
This query does not place a record in the transcript. It can be asked any number of times, as long as the next query after every *test password* is of type *implementation*. (The idea of the last requirement is that a *test password* query has to be caused by a "real-world" operation, which leaves an *implementation* record in the transcript.)

get verifier [Transcript: (`"get verifier"`, i, i')]
: Arguments: users i and i'. For each *test password* query on $\{i, i'\}$ that has previously been asked (if any), returns whether or not it was successful. If any one of them actually was successful, then this *get verifier* query is called a successful guess on $\{ID_i, ID_{i'}\}$. Note that the information about the success or failure of *test password* queries is *not* placed in the transcript.

start session [Transcript: (`"start session"`, i, j)]
: In addition to the rules specified previously, a connection assignment of *ex-*

pose for client instance (i, j) is allowed at any point after a *get verifier* query on users i and i' has been performed, where $ID_{i'} = PID_{ij}$.

The *test password* query does not affect the legality of *open* and *connect* connection assignments.

6.2 Real System—Resilience to Server Compromise

In a real system that has any resilience to server compromise, the server must not store the plaintext password. Instead, the server stores a *verifier* to verify a user's password. Thus, the protocol has to specify a PPT verifier generation algorithm $VGen$ that, given a set of user identities $\{A, B\}$, and a password π, produces a verifier V.

As above for $\pi[A, B]$, we will write $V[A, B]$ to mean $V[\{A, B\}]$.

A user instance (i, j) in the server role is given access to $V[ID_i, PID_{ij}]$. A user instance (i, j) in the client role is given access to $\pi[ID_i, PID_{ij}]$.

The changes to the *initialize user* and *set password* operations are given here:

initialize user [Transcript: $(\texttt{"initialize user"}, i, ID_i)$]
In addition to what is done in the basic real system, $V[ID_i, ID_{i'}] = VGen(\{ID_i, ID_{i'}\}, \pi[ID_i, ID_{i'}])$ is computed for each i'.

set password [Transcript: $(\texttt{"set password"}, i, ID', \pi)$]
In addition to what is done in basic real system, $V[ID_i, ID']$ is set to $VGen(\{ID_i, ID'\}, \pi)$.

We add the *get verifier* operation here:

get verifier [Transcript:$(\texttt{"get verifier"}, i, i')$, followed by $(\texttt{"impl"}, \texttt{"verifier"}, i, i', V[ID_i, ID_{i'}])$]
The adversary performs this query with i and i' as arguments, with $V[ID_i, ID_{i'}]$ being returned.

6.3 PAK-X protocol

In our protocol, we will designate the *open* role as the client role. We will use A and B to denote the identities of the client and the server, respectively. In addition to the random oracles we have used before, we will use additional functions $H_0 : \{0,1\}^* \to \{0,1\}^{|q|+\kappa}$ and $H_0' : \{0,1\}^* \to \{0,1\}^{|q|+\kappa}$, which we will assume to be random functions. The verifier generation algorithm is

$$VGen(\{A, B\}, \pi) = g^{v[A,B]},$$

where we define $v[A, B] = H_0(\min(A, B), \max(A, B), \pi)$ (we need to order user identities, just so that any pair of users has a unique verifier).

The PAK-X protocol is given in Figure 3.

Theorem 3. *The PAK-X protocol is a secure password-authenticated key exchange protocol in the explicit-authentication model, with resilience to server compromise.*

The completeness requirement follows directly by inspection. The proof of simulatability is omitted due to page limits. (The technique that allows us to perform authentication where the server stores a verifier instead of the password itself is similar to the technique developed independently in [17] to obtain an efficient encryption scheme secure against an adaptive chosen-ciphertext attack.)

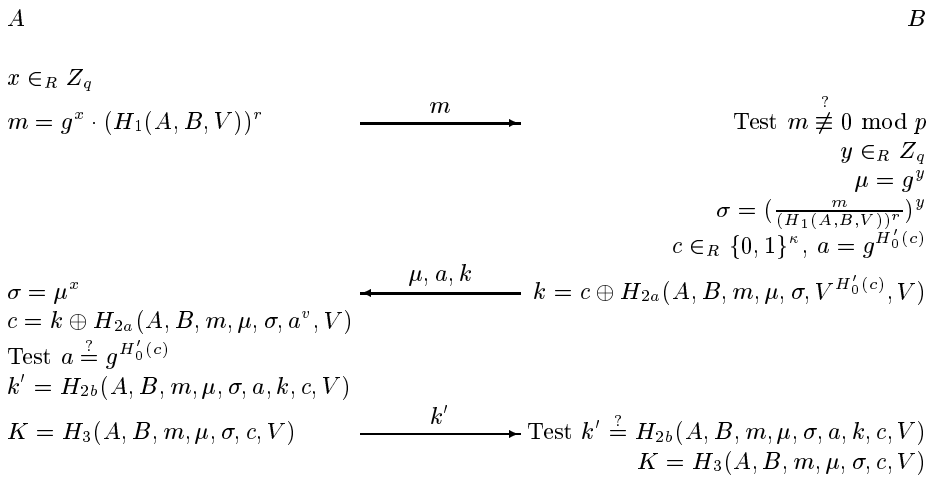


Fig. 3. PAK-X protocol, with $\pi = \pi[A,B]$, $v = v[A,B]$, and $V = V[A,B]$. The resulting session key is K.

Acknowledgements. We would like to thank Daniel Bleichenbacher for an improvement to our method of generating simulated random oracle responses (as shown in the full paper [13]).

References

1. D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *J. of Cryptology*, 4(2):75–122, 1991.
2. M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In STOC'98, pages 419–428.
3. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In EUROCRYPT 2000, pages 140–156.
4. M. Bellare and P. Rogaway. Entity authentication and key distribution. In CRYPTO'93, pages 232–249.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In ACM Security (CCS'93), pages 62–73.
6. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In EUROCRYPT'94, pages 92–111.
7. M. Bellare and P. Rogaway. Provably secure session key distribution—the three party case. In STOC'95, pages 57–66.

8. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In Proceedings of IEEE Security and Privacy, pages 72–84, 1992.
9. S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In ACM Security (CCS'93), pages 244–250.
10. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *Sixth IMA Intl. Conf. on Cryptography and Coding*, 1997.
11. D. Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 1998.
12. M. Boyarsky. Public-key cryptography and password protocols: The multi-user case. In ACM Security (CCS'99), pages 63–72.
13. V. Boyko, P. MacKenzie, and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman (full version). http://www.bell-labs.com/user/philmac/research/pak.ps.gz
14. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In STOC'98, pages 209–218.
15. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, 22(6):644–654, 1976.
16. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithm. *IEEE Trans. Info. Theory*, 31:469–472, 1985.
17. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In CRYPTO'99, pages 537–554.
18. L. Gong. Optimal authentication protocols resistant to password guessing attacks. In *8th IEEE Computer Security Foundations Workshop*, pages 24–29, 1995.
19. L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, June 1993.
20. S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. In ACM Security (CCS'98), pages 122–131.
21. D. Jablon. Strong password-only authenticated key exchange. *ACM Computer Communication Review, ACM SIGCOMM*, 26(5):5–20, 1996.
22. D. Jablon. Extended password key exchange protocols immune to dictionary attack. In *WETICE'97 Workshop on Enterprise Security*, 1997.
23. T. M. A. Lomas, L. Gong, J. H. Saltzer, and R. M. Needham. Reducing risks from poorly chosen keys. *ACM Operating Systems Review*, 23(5):14–18, Dec. 1989. Proceedings of the 12th ACM Symposium on Operating System Principles.
24. S. Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Proceedings of the Workshop on Security Protocols*, 1997.
25. P. MacKenzie and R. Swaminathan. Secure network authentication with password information. manuscript.
26. S. Patel. Number theoretic attacks on secure password schemes. In *Proceedings of IEEE Security and Privacy*, pages 236–247, 1997.
27. V. Shoup. On formal models for secure key exchange. IBM Research Report RZ 3120. April, 1999.
28. M. Steiner, G. Tsudik, and M. Waidner. Refinement and extension of encrypted key exchange. *ACM Operating System Review*, 29:22–30, 1995.
29. T. Wu. The secure remote password protocol. In NDSS'98, pages 97–111.
30. T. Wu. A real world analysis of Kerberos password security. In NDSS'99.

Fair Encryption of RSA Keys

Guillaume Poupard and Jacques Stern

École Normale Supérieure, Département d'informatique
45 rue d'Ulm, F-75230 Paris Cedex 05, France
email: {Guillaume.Poupard,Jacques.Stern}@ens.fr

Abstract. Cryptography is more and more concerned with elaborate protocols involving many participants. In some cases, it is crucial to be sure that players behave fairly especially when they use public key encryption. Accordingly, mechanisms are needed to check the correctness of encrypted data, without compromising secrecy. We consider an optimistic scenario in which users have pairs of public and private keys and give an encryption of their secret key with the public key of a third party. In this setting we wish to provide a publicly verifiable proof that the third party is able to recover the secret key if needed. Our emphasis is on size; we believe that the proof should be of the same length as the original key.
In this paper, we propose such proofs of fair encryption for El Gamal and RSA keys, using the Paillier cryptosystem. Our proofs are really efficient since in practical terms they are only a few hundred bytes long. As an application, we design a very simple and efficient key recovery system.

1 Introduction

In some cryptographic applications it is crucial to be sure that players behave fairly, especially when they use public key encryption. For example, we can consider a voting scheme where each player encrypts the name of his favorite candidate. It can be useful to convince anybody that the encrypted name is indeed in the list of the candidates without revealing any information about this name. Accordingly, mechanisms are needed to check the correctness of encrypted data, without compromising secrecy.

We consider an optimistic scenario in which users have pairs of public and private keys and give an encryption of their secret key with the public key of a third party. In this setting we wish to provide a publicly verifiable proof that the third party is able to recover the secret key if needed. We use the term *fair encryption* for such a verifiable encryption. Note that the third party is not involved during encryption or during verification of the proof. In optimistic systems like [1], the third party is active only in case of dishonest behavior of one participant; it is implicitly assumed that the knowledge that the third party is able to solve any conflict is enough to deter anybody from cheating.

Our emphasis is on size; we believe that the proof should be approximately of the same length as the original key. Consequently, general techniques of zero-knowledge proofs cannot be used and we have to design specific proof systems which are very efficient.

Previous work.
Publicly verifiable encryption is not a new concept and it has been used in applications like secret sharing or key escrow. In 1998, Young and Yung [29] proposed auto-recoverable auto-certifiable public key cryptosystems based on verifiable encryption of secret keys using double decker exponentiation which makes the proofs efficient but certainly not really practical, in a natural sense that is defined below. Furthermore, this system does not separate the recoverability verification from the ability to certify public keys.[1]

Efficient vs Practical protocols.
Following the tradition of complexity theory, cryptographers generally consider that a protocol is *"efficient"* when both its running time and the size of the transmitted data are polynomial in some typical parameters such as the bit length of the integers in use and other security parameters. This approach enables to eliminate non polynomial schemes that are obviously not usable in practice but those which survive cannot necessarily be considered practical. For example we can think about general multi-party computation protocols.

In this paper, we focus on protocols that are efficient but also really practical. As an example, let us consider the Fiat-Shamir identification scheme [10]; if we note k the security parameter and $|N|_{\mathrm{b}}$ the size of used integers, its time complexity is $O(k \times |N|_{\mathrm{b}}{}^2)$ and the communication complexity measuring the size of the exchanged data is $O(k \times |N|_{\mathrm{b}})$. Thus this protocol is efficient but not very practical. As another example, the Schnorr scheme [24] is efficient and even practical since its time complexity is $O(|N|_{\mathrm{b}}{}^2)$ and its communication complexity is $O(k + |N|_{\mathrm{b}})$; the security parameter k may be raised with only a modest increase in size.

Our aim is to design proof systems that are practical at least in terms of communication, i.e. such that the size of the proofs are of the same order than the size of the underlying objects. This is motivated by the scenario that we have in mind since we wish to turn our proofs into non interactive "certificates". In this setting, the optimization of the size of transmitted data is of crucial importance.

Our results.
In this paper, we propose proofs of fair encryption for secret keys of any encryption scheme based on the discrete logarithm problem or on the intractability of the factorization, including RSA and its variants. The asymmetric secret keys are encrypted using any homomorphic public key cryptosystem like those of Naccache-Stern [17], Okamoto-Uchiyama [18] or Paillier [19]. In this paper we only focus on the Paillier scheme but we can immediately adapt the protocols in order to use the Okamoto-Uchiyama cryptosystem which onewayness is based on the well studied factorization problem instead of the new one introduced in [19].

More precisely, we give a protocol to prove that a ciphertext enables a third party to recover the El Gamal secret key related to a public one. Such a proof is very short and the workload of the third party during recovery is very small. We

[1] Those results have been recently improved in [30]. See also [27].

also propose a scheme for fair encrypting the factorization of a public modulus. Such a proof is also very small but the workload of the third party is much more important than for El Gamal keys since, from a theoretical point of view, the recovery time and the cheating time are polynomially related. However, we describe practical parameters to show that actual applications are feasible.

Finally, as an application, we design a very simple and efficient key recovery system that can be used with any kind of keys, including RSA keys. We propose the first non-interactive proof of recoverability of RSA keys short enough (a few hundred bytes) to be appended as a certificate to any ciphertext. A consequence is that the recoverability verification is no longer performed by the certification authority. Consequently, this approach is more flexible than auto-recoverable cryptosystems [29] and more secure than binding cryptography [28].

Those results follow from a careful analysis of previously proposed building blocks: homomorphic cryptosystems based on exponentiation modulo composite integers, the so-called bounded range commitment schemes that tries to prove the knowledge of a discrete logarithm in a given range and short proofs of knowledge for factoring proposed in [23].

Outline of the paper.
In section 2 we describe notations and give a precise description of the three building blocks: trapdoor discrete logarithm cryptosystems, Diophantine commitment and short proof of knowledge for factoring. Security proofs for those two last protocols appear in appendix. Next, in section 3, we describe our fair encryption protocols first for El Gamal and then for RSA. Finally, in section 4 we show how fair encryption enables the design of very simple and efficient key escrow systems.

2 Preliminary tools

Throughout this paper, we use the following notation: for any integer n,
- we use $\varphi(n)$ to denote the Euler totient function, i.e. the cardinality of ${\mathbb{Z}_n}^*$,
- we use $\lambda(n)$ to denote Carmichael's lambda function defined as the largest order of the elements of ${\mathbb{Z}_n}^*$.

It is well known that if the prime factorization of an odd integer n is $\prod_{i=1}^{\eta} {q_i}^{f_i}$ then $\varphi(n) = \prod_{i=1}^{\eta} {q_i}^{f_i-1}(q_i - 1)$ and $\lambda(n) = \mathrm{lcm}_{i=1\ldots\eta}\left({q_i}^{f_i-1}(q_i-1)\right)$.

For any integer x, $|x|_{\mathrm{b}} = (\lfloor \log_2(x) \rfloor + 1)$ is the number of bits of x. Finally, a prime number p is a *strong prime* if $p = 2p' + 1$ and p' is also prime. Our computing model is the probabilistic polynomial time Turing machine (PPTM), whose running time is a polynomial in specified parameters.

2.1 Homomorphic cryptosystems

Various cryptosystems which encrypt a message M by raising a base g to the power M modulo some integer have been proposed [15, 3, 17–19]. Their security is related to the intractability of computing discrete logarithm in the base g. As

usual, the computation becomes easy using a trapdoor. As an important consequence of this encryption technique, those schemes have homomorphic properties that can be informally stated as follows:

$$E(M_1 + M_2) = E(M_1) \times E(M_2) \quad \text{and} \quad E(k \times M) = E(M)^k$$

The early examples of such schemes could only achieve bit by bit encryption [15] or had very limited bandwidth [3]. However, recently, three cryptosystems with significant bandwidth have been proposed: one by Okamoto and Uchiyama [18] based on the exponentiation modulo P^2Q of messages from $\mathbb{Z}_P$ where P and Q are prime numbers, the second by Naccache and Stern [17] based on the exponentiation modulo PQ of messages from $\mathbb{Z}_\sigma$ with σ a smooth divisor of $\varphi(PQ)$ and finally a proposal of Paillier [19] which extends the system of [18] by using exponentiation modulo P^2Q^2 and messages from $\mathbb{Z}_{PQ}$. In the following, we only describe protocols based on the Paillier cryptosystem but we insist on the fact that any of those three cryptosystems could be used.

The Paillier cryptosystem is based on the properties of the Carmichael lambda function in ${\mathbb{Z}_{N^2}}^*$. We recall here the main two properties: for any $w \in {\mathbb{Z}_{N^2}}^*$, $w^{\lambda(N)} = 1 \bmod N$ and $w^{N\lambda(N)} = 1 \bmod N^2$.

Key Generation. Let N be an RSA modulus $N = P \times Q$, where P and Q are prime integers s.t. $\gcd(N, \varphi(N)) = 1$. Let G be an integer of order multiple of N modulo N^2. The public key is (N, G) and the secret key is $\lambda(N)$.

Encryption. To encrypt a message $M \in \mathbb{Z}_N$, randomly choose u in ${\mathbb{Z}_N}^*$ and compute the ciphertext $c = G^M \times u^N \bmod N^2$.

Decryption. To decrypt c, compute $M = \dfrac{L(c^{\lambda(N)} \bmod N^2)}{L(G^{\lambda(N)} \bmod N^2)} \bmod N$ where the L-function takes as input an element equal to 1 modulo N and computes $L(u) = \frac{u-1}{N}$.

The integer $g^{\lambda(N)} \bmod N^2$ is equal to 1 modulo N so there exists $\beta \in \mathbb{Z}_N$ such that $g^{\lambda(N)} = 1 + \beta N \bmod N^2$. Furthermore, we note that $\beta = L(g^{\lambda(N)} \bmod N^2)$. Consequently, $c^{\lambda(N)} = \left(g^M u^N\right)^{\lambda(N)} = \left(g^{\lambda(N)}\right)^M = (1 + \beta N)^M = 1 + M\beta N \bmod N^2$. So $M \times L(g^{\lambda(N)}) = L(c^{\lambda(N)}) \bmod N$.

Security. It is conjectured that the so-called composite residuosity class problem, that exactly consists in inverting the cryptosystem, is intractable. The semantic security is based on the difficulty to distinguish N^{th} residues modulo N^2. We refer to [19] for details.

2.2 Diophantine Commitment

In 1989, Schnorr [24] proposed his famous signature scheme which may be viewed as proof of knowledge of a discrete logarithm modulo a prime number. Since then, many authors have tried to adapt the scheme in order to add control over the size of the secret value. Such a *bounded-range commitment* has many applications and it has been used for group signature by Camenisch and Michels [6], for

electronic cash by Chan, Frankel and Tsiounis [8], for verifiable secret sharing by Fujisaki and Okamoto [12] and finally for proving that a modulus is the product of two safe primes by Camenisch and Michels [7]. However no satisfactory solution has appeared at the moment[2]. Known proposals are only able to prove that the discrete logarithm is not "too far" from a fixed range, their analysis is complex (and sometimes erroneous as in the Eurocrypt '98 version of [8]) and their security is often based on non-standard assumptions such as the so-called "strong RSA assumption" needed to make proofs efficient. In this paper, we adopt a totally different strategy in the analysis of bounded-range commitment schemes.

Let $\mathcal{G}$ be a multiplicative finite group. As an example of such a group, in the next sections we use groups of unknown order $\mathcal{G} = \mathbb{Z}_{N^2}{}^*$ where N is an RSA modulus. Let S be an integer and G be an element of $\mathcal{G}$. We consider a player who has a secret integer x that lies in the range $[0, S[$ and who computes, in $\mathcal{G}$, the public value $\Gamma = G^x$.

We do not know how to prove the knowledge of a discrete logarithm in the range $[0, S[$ of Γ in base G. Consequently we only prove a weaker property. Let A and B be two parameters whose values are analyzed later. We describe a practical statistically zero-knowledge interactive proof of knowledge of $\sigma \in]-A, A[$ and $\tau \in]0, B[$ such that $G^\sigma = \Gamma^\tau$ in $\mathcal{G}$. It should be clear that we do not prove the knowledge of $x \in [0, S[$ such that $\Gamma = G^x$. However, in practice, the prover needs to know such an x in order to be able to perform the proof.

> **Protocol 1**: The following round is repeated ℓ times. At each round, the prover randomly chooses an integer r in $[0, A[$ and computes the commitment $t = G^r$ in $\mathcal{G}$. Then he sends t to the verifier who answers a challenge e randomly chosen in $[0, B[$. The prover computes $y = r + ex$ (an integer in $\mathbb{Z}$) and sends it to the verifier who checks $t = G^y \times \Gamma^{-e}$ in $\mathcal{G}$ and $0 \leq y < A$.

A security analysis of this scheme is proposed in appendix A. Note that this protocol is similar to previous proposals for bounded-range commitment [6, 8, 12, 7] but that the analysis is really different and does not use non-standard hypothesis like the strong-RSA assumption.

Let us summarize the security results. A prover who knows $x \in [0, S[$ is accepted with probability higher than $1 - \ell SB/A$ so A must be much larger than ℓSB in order to make the protocol correct. Furthermore, the protocol is sound, i.e. a prover who convinces a verifier with probability higher that $1/B^\ell$ must know $\sigma \in]-A, A[$ and $\tau \in]0, B[$ such that $G^\sigma = \Gamma^\tau$. Finally, in the complexity theory setting, if we consider a security parameter k and if all the parameters A, B, S and ℓ are viewed as functions of k, the protocol is statistically zero-knowledge if $\ell \times B$ is polynomial in k and if $\ell SB/A$ is negligible.

When S is chosen, the choice of the remaining parameters is directed by those results. From a theoretical point of view, if we consider that the security

[2] Last minute : see F. Boudot's paper [5] in this volume, pp. 437–450.

parameter k is related to the cheating probability $1/2^k$ for an adversary, the soundness implies that $B^\ell \geq 2^k$. Furthermore, the protocol is zero-knowledge if it can be simulated in polynomial time. Since the time complexity of the simulation is $O(\ell \times B)$, the parameters ℓ and B must be polynomial in k. Finally, the correctness and the zero-knowledge property show that A must be such that $\ell SB/A$ is negligible.

From a practical point of view, we can fix the security parameter k to 80 for example. If $|S|_\mathrm{b} = 160$, the following practical values for the other parameters are reasonable: $B = 2^{20}$, $\ell = 4$, $A = 2^{2+160+20+80} = 2^{262}$.

Let ω be the order of G in $\mathcal{G}$. Note that the relation $G^\sigma = \Gamma^\tau$ does not even imply the existence of a discrete logarithm for Γ with base G but, if $\Gamma = G^x \bmod N^2$, we have $x\tau = \sigma \bmod \omega$ and consequently $x = \sigma \times \left(\frac{\tau}{\gcd(\tau,\omega)}\right)^{-1} \bmod \frac{\omega}{\gcd(\tau,\omega)}$.

The Diophantine commitment can be made non-interactive using the Fiat-Shamir heuristic [10]. The verifier's challenge e is replaced with the hash value of the commitment t and of the public data using a collision-resistant hash function H. It is widely believed that such a transformation guarantees an accurate level of security as soon as H is random enough. Furthermore, the security of this approach can be formalized using the random oracle model [20].

2.3 Short proof of knowledge for factoring

Proofs of knowledge for the factorization of an integer n have been known for a long time. But, even if they are claimed efficient according to complexity theoretical arguments, none of them can be considered practical for many applications because of their significant communication complexity: the proof is much longer than the object it deals with.

A new strategy have been used in [23]. The protocol is a proof of knowledge of a small common discrete logarithm of $z^n \bmod n$ for a few randomly chosen elements z modulo n. This scheme is very efficient; when suitably optimized, its communication complexity is only $O(k + |n|_\mathrm{b})$ bits, where k is a security parameter. In this setting, the size of the proof is similar to the size of the integer n. The improvement in comparison with the previously known schemes can therefore be compared with the difference of efficiency between the Fiat-Shamir scheme and the Schnorr one. Furthermore, the computational complexity is proved to be $O((|n|_\mathrm{b}+k)\times k)$ multiplications modulo n both for the prover and the verifier but strong heuristic evidence shows that $O((|n|_\mathrm{b} + k) \times k/\log k)$ is enough. This might appear a small improvement but it has drastic consequences in practical terms: only three modular exponentiations both for the prover and the verifier are needed to obtain a very high level of security.

Protocol 2: First the prover and the verifier agree on mutually randomly chosen integers $z_i \in {\mathbb{Z}_n}^*$ for $i = 1..K$. Then the following elementary round is repeated ℓ times. The prover randomly chooses an integer r in $[0, A[$ and computes, for $i = 1..K$, the commitments $t_i = z_i^r \bmod n$.

Then he sends the t_is to the verifier who answers a challenge e randomly chosen in $[0, B[$. The prover computes $y = r + (n - \varphi(n)) \times e$ (in $\mathbb{Z}$) and sends it to the verifier who checks $0 \leq y < A$ and, for $i = 1..K$, $z_i^{y-n\times e} = t_i \bmod n$.

A security analysis of this scheme appears in [23]. The choice of the parameters ℓ and B must be such that $B^\ell > 2^k$, where k is a security parameter, in order to make the protocol sound. Furthermore, the parameter A must be much larger than $(n-\varphi(n))\ell B$ to guarantee the completeness and the zero-knowledge property but A must also be smaller than n to guarantee the soundness. Consequently, n must verify $(n - \varphi(n))\ell B \ll n$. For the applications we consider in this paper, such a proof is used to prove the knowledge of integers like RSA modulus with large prime factors so this condition is always satisfied. Note that if n has small prime factors, the proof is no longer zero-knowledge but it is still sound so a prover cannot try to cheat choosing an integer n with small factors. Such a choice would only compromise his own security.

Using classical techniques, the commitments t_i can be hashed. This trick makes the communication complexity independent of K. Accordingly, the protocol is really practical in term of communication whatever K may be. Furthermore, the proof can be made non-interactive; the z_i are chosen by means of a hash function repeatedly applied to the integer n and the verifier's challenge e is replaced with the hash value of the commitments and of the public data. The size of such a proof is very small in practice, i.e. similar to the size of n.

3 Fair Encryption of secret keys

We consider a third party who chooses his own private and public keys in the Paillier cryptosystem. Let (N, G) be his public key. We also consider a user who has a pair (SK, PK) of related secret and public keys for any cryptosystem (not necessarily Paillier's one). A fair encryption of SK consists of a ciphertext Γ and of a non-interactive proof of fairness; Γ encrypts some secret data related to SK with the public key of the third party and the proof convinces anybody that the third party is able to recover SK using PK, Γ and his Paillier secret key.

Note that the third party is not involved in the process of fair encryption or during verification of the proof of fairness. As in many "optimistic" systems like [1], the third party is active only in case of dishonest behavior of one participant; it is implicitly assumed that the knowledge that the third party is able to solve any conflict is enough to deter anybody from cheating.

We propose fair encryption scheme for secret keys of all known public key encryption schemes based on the discrete logarithm problem or on the difficulty of factorization. We first give the application to the case of El Gamal keys. Then, we study the case of RSA keys.

3.1 Fair encryption of El Gamal type keys

A fair encryption of an El Gamal secret key x consists of an encryption Γ of x, obtained with the public key of a third party, and of a publicly verifiable non-interactive proof that the third party would be able to recover x from Γ and Y if needed. Such systems have already been proposed (see for example the attempt in [2]) but we explain at the end of this section why previous solutions are not satisfactory. Note that, in order to simplify the presentation of the protocol, we only consider non-randomized Paillier scheme but a semantically secure version can be used as shown in the next section for the case RSA case.

A third party first chooses his public key (N, G) and the related private key to be used with the Paillier cryptosystem, i.e. $N = PQ$ an RSA modulus and G an element of order multiple of N in ${\mathbb{Z}_{N^2}}^*$. Let us further consider a strong prime number $p = 2q + 1$ and a generator g of ${\mathbb{Z}_p}^*$. Each user chooses a private key $x \in \mathbb{Z}_{p-1}$ and computes his public key $Y = g^x \bmod p$. In order to make a fair encryption of his secret key x, he computes the ciphertext $\Gamma = G^x \bmod N^2$ and a non-interactive proof of third party's ability to compute x from Y and Γ. We now describe an interactive version of such a proof that will further be made non-interactive.

We define $S = p - 1$ and $\mathcal{G} = {\mathbb{Z}_{N^2}}^*$. Let A, B and ℓ be Diophantine commitment parameters as described in section 2.2.

Protocol 3:
The following round is repeated ℓ times. At each round, the prover randomly chooses an integer r in $[0, A[$ and sends the commitment $t = (\Gamma^r \bmod N^2, g^r \bmod p)$ to the verifier who answers an integer e randomly chosen in $[0, B[$. The prover computes $y = r + ex$ and sends it to the verifier who checks $t = (G^y \times \Gamma^{-e} \bmod N^2, g^y \times Y^{-e} \bmod p)$ and $0 \leq y < A$.

This protocol runs in parallel the Girault scheme [13] analyzed in [22] and Diophantine commitment. Just as for each of the two schemes separately, we can prove correctness and statistical zero-knowledge property provided $\ell SB/A$ is negligible and $\ell \times B$ is polynomial in the security parameter k. Furthermore, if a prover is accepted with probability $> 1/B^\ell$ then he must know (σ, τ) with $|\sigma| < A$, $0 < \tau < B$, $G^\sigma = \Gamma^\tau \bmod N^2$ and $g^\sigma = Y^\tau \bmod p$. In other words, if an adversary viewed as a probabilistic Turing machine is accepted with probability $> 1/B^\ell$, we can use it in order to make an extractor that computes such a pair (σ, τ).

Theorem 1. *The third party can recover a fair encrypted secret key x from Y and Γ in time $O(|N|)$ if the recoverability proof is valid, provided $N \geq 2\sqrt{2}AB$.*

Proof: First note that the discrete logarithm x of Y in base g modulo p, i.e. the secret key x related to the El Gamal public key Y, exists because g generates ${\mathbb{Z}_p}^*$. The proof associated with the encryption Γ shows that there exists (σ, τ)

such that $g^\sigma = Y^\tau \bmod p$ so we have $\sigma = \tau \log_g Y \bmod p-1$. As $q = (p-1)/2$ is a prime number, we obtain $\sigma/d = \tau/d \times x \bmod q$, where $d = \gcd(\sigma, \tau)$. Consequently, the knowledge of $(\sigma/d, \tau/d)$ enables to recover x because we can compute $x_0 = (\sigma/d) \times (\tau/d)^{-1} = \sigma_0 \times {\tau_0}^{-1} \bmod q$ and the secret key $x \bmod p-1$ is x_0 or $x_0 + q$.

Finally, it is enough to show that a third party can recover $(\sigma_0, \tau_0) = (\sigma/d, \tau/d)$ from Γ and Y. We show that this can be efficiently done, provided $N \geq 2\sqrt{2}AB$.

First, he decrypts Γ and obtains $\gamma = \dfrac{L(\Gamma^{\lambda(N)} \bmod N^2)}{L(G^{\lambda(N)} \bmod N^2)} \bmod N$ so $\Gamma^{\lambda(N)} = G^{\gamma\lambda(N)}$. Since $G^\sigma = \Gamma^\tau \bmod N^2$, the previous equation implies $\sigma - \gamma\tau = 0 \bmod N$. Let us consider the solutions of the equation $x - \gamma y = 0 \bmod N$ where x and y are the unknowns. They are elements of a lattice with basis $((N,0),(\gamma,1))$. Since the dimension of the lattice is 2, we can use Gauss' algorithm [9, p.23] in order to find its shortest vector. When running this algorithm, we need to specify the inner product; for the sake of optimization, we replace the standard inner product by $(x,y).(x',y') = xx' + A^2/B^2 \times yy'$. The corresponding norm is $||(x,y)|| = \sqrt{x^2 + A^2/B^2 \times y^2}$. Receiving basis $((N,0),(\gamma,1))$ as its input, the algorithm outputs the shortest vector (σ_0, τ_0) of the lattice. The (unknown) vector (σ, τ) is also in the lattice so that $||(\sigma_0,\tau_0)|| \leq ||(\sigma,\tau)|| < \sqrt{A^2 + A^2/B^2 \times B^2} = \sqrt{2}A$. This means that $|\sigma_0| < \sqrt{2}A$ and $|\tau_0| < \sqrt{2}B$.

From the equations $\gamma\tau - \sigma = \gamma\tau_0 - \sigma_0 = 0 \bmod N$ we obtain $\sigma_0\tau = \tau\tau_0\gamma = \sigma\tau_0 \bmod N$. But $|\sigma_0\tau - \sigma\tau_0| \leq |\sigma_0||\tau| + |\sigma||\tau_0| < 2\sqrt{2}AB$ so, if $N \geq 2\sqrt{2}AB$, $\sigma_0\tau = \sigma\tau_0$ in $\mathbb{Z}$. Furthermore, (σ_0, τ_0) is the shortest vector of the lattice so $\gcd(\sigma_0, \tau_0) = 1$. Finally, the output of the algorithm leads to the computation of the pair $(\sigma/d, \tau/d)$ where d if the gcd of σ and τ. Furthermore, since $0 < \tau < B$, d is less than B.

Classical results about the complexity of Gauss' algorithm (see for example [25]) prove that the number of repetitions needed to find (σ_0, τ_0) is $O(\log(N))$. □

As a consequence, the key recovery process is efficient from a theoretical point of view. Furthermore, practical experiments confirms very high efficiency since a few milliseconds computation can recover the key, whatever the encryption may be but provided the proof is valid.

In conclusion, the protocol is secure both for the prover and the verifier. An dishonest verifier cannot obtain any extra information about SK and if the proof is accepted the third party can recover SK whatever the encryption Γ, even if the prover is dishonest and have unlimited computation power.

Non-interactive version and Optimizations. Many well known optimizations can be applied to the previous proof. The commitment can be replaced by its hash value as described in [14] and it can be precomputed in order to reduce the on-line computation to a very simple non-modular arithmetic operation. We can also reduce the size of the secret key x to about 160 bits as explained in [26]. Finally, this proof can be made non-interactive in order to obtain a very short certificate of fair encryption.

Comparison with previous proposals. At first sight, the key recovery procedure based on lattice reduction might seem overly intricate. We explain why a simple decryption of Γ (as proposed in [2]) presumably does not always enable to recover the secret key.

Let us consider the following cheating strategy based on the ability to extract f-th root, where f is small, without being able to factor. This is a plausible assumption as explained in [4]. The (dishonest) prover chooses an x, computes Y and $\Gamma = G^x \bmod N^2$. Then he extracts an f-th root $\widetilde{\Gamma}$ of Γ modulo N^2. When f divides the challenge e, which happens with probability $1/f$, the prover answers $z = r + (e/f)x$. The verification is still correct but, when the third party decrypts Γ, he obtains a value that has nothing to do with the the discrete logarithm of Y in base g modulo p.

In order to overcome the difficulty one can use a non-standard intractability assumption, the so-called "strong RSA problem", which appears in several papers [12, 6]. With our system, under standard assumption, the third party would find σ and τ such that $g^\sigma = Y^\tau \bmod p$, since $\sigma = (e - e')/f \times x = (\tau/f)x$, and consequently the correct value of the secret key x as was previously explained.

3.2 Fair encryption of RSA keys

We now turn to fair encryption of RSA keys. Using Diophantine commitment and short proofs of knowledge for factoring, we design a fair encryption system which enables the third party to recover the factorization of the RSA modulus, even if it is not of a correct form, i.e. if it is not the product of two large safe primes of approximately the same length. The originality of our solution, in comparison with other proposals is that it does not include any proof that the RSA modulus has exactly two different prime factors. This has important consequence on efficiency.

We consider a scenario where each user chooses two k'-bit prime numbers p and q and computes his RSA modulus $n = pq$. He also computes $x = n - \varphi(n) = p + q - 1$ and the encryption $\Gamma = G^x \bmod N^2$.

We now describe a scheme that enables the user to convince a verifier that the third party is able to factor n using Γ and is Paillier secret key. Let A, B, ℓ and K be parameters of short proof of knowledge for factoring as exposed in section 2.3.

> **Protocol 4**: First the prover and the verifier agree on mutually randomly chosen integers $z_i \in \mathbb{Z}_n{}^*$ for $i = 1..K$. Then the following basic round is repeated ℓ times. The prover randomly chooses an integer r in $[0, A[$ and sends the commitment $t = (G^r \bmod N^2, \{z_i^r \bmod n\}_{i=1..K})$. Then the verifier answers an integer e randomly chosen in $[0, B[$. The prover computes $y = r + ex$ and sends it to the verifier who checks $t = (G^y \times \Gamma^{-e} \bmod N^2, \{z_i^{y-en} \bmod n\}_{i=1..K})$ and $0 \leq y < A$.

The protocol is a parallel execution of a Diophantine commitment and of a short proof of knowledge for factoring. If a prover is accepted with probability

$> 1/B^\ell$ then, as usual, one can find a round for which he is able to correctly answer y and y' to different challenges e and e' ($e > e'$) following an identical commitment t. Consequently, for all $i = 1..K$, $z_i^{y-y'} = z_i^{n(e-e')} \bmod n$. If we note $\sigma = y - y'$, $\tau = e - e'$ and $L = n\tau - \sigma$, we have

$$\boxed{\sigma \in]-A, A[,\ \tau \in]0, B[,\ G^\sigma = \Gamma^\tau \bmod N^2 \text{ and, for all } i = 1..K,\ z_i^L = 1 \bmod n}$$

If σ and τ, and consequently L, are known, the same technique as for the proof of soundness of protocol 2 shows that the factorization of n can be extracted with $O(|n|_{\mathrm{b}} \times |L|)$ multiplications modulo n.

Theorem 2. *The third party can factor n from the fair encryption Γ in time $O(|N| + \sqrt{B})$ if the recoverability proof is valid, provided $N \geq 2\sqrt{2}AB$.*

Proof: First, using the same procedure as for El Gamal keys, he computes $(\sigma_0, \tau_0) = (\sigma/d, \tau/d)$ with $d = \gcd(\sigma, \tau)$. Let L_0 be $n\tau_0 - \sigma_0$; since $L = n\tau - \sigma = d \times L_0$ and $d = \gcd(\sigma, \tau) \leq |\tau| < B$, the third party recovers L divided by a factor d smaller than B.

This missing information can be computed using an algorithm which finds the order of the z_is as follows. For any i, we know that the order of $y = z_i^{L_0} \bmod n$ is less than B because $z_i^L = 1 \bmod n$. The λ-method of Pollard [21] enables to find this order in time $O(\sqrt{B})$ with memory complexity $O(1)$. The idea is to choose a randomly looking function f and to iteratively compute $y_{i+1} = y_i \times y^{f(y_i)} \bmod n$, with $y_0 = 1$, for $i = 1..M$ where M is a fixed parameter. Then, just remembering this last value, we compute ${y'}_{i+1} = {y'}_i \times y^{f({y'}_i)} \bmod n$, with ${y'}_0 = y^B$, until we find an index M' such that $y_M = {y'}_{M'} \bmod n$ or until M' exceeds a fixed bound. If a collision $y_M = {y'}_{M'} \bmod n$ is found (see [21] for a precise analysis), it leads to $y^{B+\sum_{i=0}^{M'-1} f({y'}_i) - \sum_{i=0}^{M-1} f(y_i)} = 1 \bmod n$ so $L_0 \times \left(B + \sum_{i=0}^{M'-1} f({y'}_i) - \sum_{i=0}^{M-1} f(y_i) \right)$ is a multiple of the order of z_i modulo n.

Finally, in time $O(\sqrt{B})$ and with a small amount of memory, the third party recovers L and then factors n with high probability. □

As a consequence of the time complexity of the algorithm in $O(\sqrt{B})$, if B is exponential in the security parameter k, the extractor is not efficient from a theoretical point of view. However, we show in the next sections that the parameters B and ℓ can be chosen in order to guarantee a high level of security, to make the key recovery process feasible by the third party and to have short proofs.

We insist on the fact that our system does not require the modulus n to have exactly two factors; a cheating user cannot gain any advantage using a modulus with three or more factors. Furthermore, the protocol can be immediately adapted to cryptosystems like Okamoto-Uchiyama's where the modulus is not an RSA modulus (e.g $n = p^2 q$).

Remark about cheating provers. In order to show why we need a key recovery procedure that might seem at first sight overly intricate, we consider

a cheating strategy that enables a dishonest prover to encrypt something different from $n - \varphi(n)$ in Γ and to give a valid proof. Let f be a factor of $\lambda(n)$ and Γ be $G^{n-\alpha\lambda(n)/f} \bmod N^2$, where α is an integer of about the same size as f. The prover follows the protocol but only answers when f divides the challenge e; this happens with probability $1/f$. In this case he returns $y = r + (e/f) \times (n - \alpha\lambda(n))$. Consequently, verifications are correct because $z_i^y = z_i^r \times z_i^{en} \times z_i^{-e\alpha\lambda(n)/f} \bmod N^2$ and the last term is equal to 1 because f divides e but the third party cannot immediately recover the missing factor f. Notice that such a cheating strategy implies a workload $O(f)$ for cheating but only a workload $O(\sqrt{f})$ for the third party to defeat it.

Randomized non-interactive version. In order to prove the semantic security of the Paillier cryptosystem, the encryption has to be probabilistic. This can be done by multiplying with $u^N \bmod N^2$, where u is randomly chosen in ${\mathbb{Z}_N}^*$: $\Gamma = G^{n-\varphi(n)} \times u^N \bmod N^2$. We can easily modify our schemes in order to use this version of the Paillier scheme. Furthermore, when a third party wants to recover a secret key, the randomization does not affect the decryption process so that nothing is changed in the key recovery. Finally, the proof can be made non-interactive. We obtain the following protocol:

Protocol 5:
Encryption. Choose $u \in {\mathbb{Z}_N}^*$ and compute $\Gamma = G^{n-\varphi(n)} \times u^N \bmod N^2$

Proof of Fairness.
Choose $(r_i)_{i=1..\ell} \in_R [0, A[^\ell$ and $(v_i)_{i=1..\ell} \in_R {\mathbb{Z}_N}^{*\ell}$
Compute $t = \left(\left(G^{r_i} v_i^N \bmod N^2 \right)_{i=1..\ell}, \left(z_j^{r_i} \bmod n \right)_{i=1..\ell, j=1..K} \right)$
and $(e_1, \ldots e_\ell) = H\left(t, N, G, (z_j)_{j=1..K}, n\right)$
Compute $y_i = r_i + e_i(n - \varphi(n))$ and ${y_i}' = u^{e_i} \times v_i \bmod N$ for $i = 1..\ell$

A non-interactive proof of fairness is a 3ℓ-tuple $\left(y_i, {y_i}', e_i\right)_{i=1..\ell}$

Verification.
Check $0 \leq y_i < A$ for $i = 1..\ell$
Compute $t' = \left(\left(G^{y_i} \times {y_i}'^N / \Gamma^{e_i} \bmod N^2 \right)_{i=1..\ell}, \left(z_j^{y_i - e_i n} \bmod n \right)_{i=1..\ell, j=1..K} \right)$
Check $(e_1, \ldots e_\ell) = H\left(t', N, G, (z_j)_{j=1..K}, n\right)$

Fair encryption of RSA keys in practice. This section is more practical in character; we consider fair encryption, using protocol 5, for a 1024-bit RSA modulus n with two 512-bit prime factors.

Choice of ℓ and B: the probability of a cheating strategy to succeed during a proof of fairness is smaller than $1/B^\ell$ so $\ell \times |B|_{\mathrm{b}}$ must be large enough, e.g. $\ell \times |B|_{\mathrm{b}} = 80$, in order to guarantee a high level of security. Furthermore, the workload for the third party is $O(\sqrt{B})$ in worst cases so B may not be too large. The choice $\ell = 2$ and $B = 2^{40}$ seems satisfactory.

Choice of A: this parameter must be smaller than n and much larger than $(n-\varphi(n))\ell B$ in order to make proofs of knowledge for factoring secure. Since n has two prime factors of about the same size, $n-\varphi(n) \approx \sqrt{n}$. Consequently, A must satisfy $512+1+40 \ll |A|_{\mathrm{b}} < 1024$; we advise $A = 2^{633}$.

Choice of K: [23] analyzes the choice of K and shows, using heuristic arguments, that $K = 3$ is a good choice. As was already observed, the communication complexity of protocol 5 does not depend of K. Consequently, the use of $K = 80$ in order to reach a high level of provable security does not make proofs longer.

Choice of N: the Paillier modulus N must satisfies $N > 2\sqrt{2}AB$ in order to make the key recovery process possible. With the previously advised values of parameters A and B, this means $|N|_{\mathrm{b}} > 675$. Consequently, in order to guarantee the security of the Paillier cryptosystem, $|N|_{\mathrm{b}} = 1024$ seems to be a good choice.

Choice of H: the function H must be a collision-resistant cryptographic hash function; SHA-1 is a good candidate.

Choice of G: the base G must be an element of order multiple of N modulo N^2. It is very simple to find such an element.

Choice of z_js: the z_js must ideally be mutually randomly chosen in the interactive setting. In practice, they can be pseudo-randomly generated, using a hash function H', with a formula like $z_j = H'(N, G, n, \Gamma, j) \bmod n$.

With those parameters, a complete fair encryption, including the RSA modulus n (1024 bits), the encryption Γ of $n-\varphi(n)$ (2048 bits) and the previously described non-interactive proof (2612 bits) is about only 710 bytes long.

4 Application to Key Recovery systems

As an example of application of fair encryption, we now explain its use in designing very efficient key recovery systems. It must be clear that our aim is not to enter into the controversial debate on the notion of key recovery but to give an application of fair encryption. The general criticisms against such systems are still topical questions. Also, we believe that our notion will find other application, e.g. in the areas of electronic cash, voting schemes or lotteries.

We consider three kinds of participants: users which want to exchange encrypted messages, authorities which are seeking the guaranty that they will obtain the decryption of some messages in specific cases and key recovery agents able to decrypt ciphertexts when requested by the proper authority. Our key recovery systems are designed to be used very easily with any cryptosystem, without adding interaction with authorities, third parties or key recovery agents. The basic idea consists in appending to any ciphertext C a fair encryption Γ of the asymmetric secret key that enables the decryption of C. Γ is encrypted with the Paillier public key of a key recovery agent. The proof of fairness provides a way for anyone (including authorities, proxies, users, ...) to check the correctness of Γ without interaction with any kind of centralized authority and consequently to be convinced that the key recovery agent can actually decrypt C.

Using the Young and Yung setting [29], this leads to the design of auto-recoverable auto-certifiable versions of all the cryptosystems based on discrete logarithm or on factoring. This includes all variants of RSA, the homomorphic schemes [17–19] and many other cryptosystems. But the shortness of our proofs, a few hundred bytes, enables more flexible mechanisms where recoverability verification is separated from the ability to certify public keys. It seems realistic to append short non-interactive proofs to any encrypted message; this leads to a very simple and efficient key recovery system which can be used in conjunction with any common cryptosystem.

We consider a new public key scenario in which each user publicizes is public key PK, a certificate for this key, i.e. a signature of an authority that guarantees the authenticity of PK, and a fair encryption of the secret key related to PK that may enable a key recovery agent to decrypt any ciphertext encrypted with PK. The proof of fairness can be checked by anybody, including people who want to send messages encrypted using PK. In the so-called fair public key scenario, the fair encryption of the secret key related to PK is added to any ciphertext encrypted with PK. Of course, this does not guarantees that it has really been encrypted with PK but the aim of key escrow schemes is only to avoid the use of regular public key infrastructure in dishonest ways; we cannot avoid simple countermeasure like over-encryption or steganography for example. The fair public key scenario can for example be used in a network where servers deliver encrypted messages to Alice only if a fair encryption of her secret key is added to ciphertexts.

Note on shadow public keys. Kilian and Leighton have shown in [16] than many key escrow schemes can be easily misused by dishonest users. The basic idea is to use non-escrowed keys that may be computed from regularly escrowed ones. As a first consequence, the secret keys must be jointly generated by users and authorities. Furthermore, in the more specific case of the system we propose, the proof of fairness should not be used as a subliminal channel to publicize a non-escrowed public key. For example, it is easy to fix a few bits, e.g. in the e_i, but we cannot see any way to increase the bandwidth of such a channel to transmit enough information.

Note on chosen ciphertext attacks. All the known cryptosystems based on a trapdoor discrete log [17–19] are only secure against chosen plaintext attacks but not against chosen ciphertext attacks. As an example, if it is possible to obtain the decryption of a ciphertext in the Okamoto-Uchiyama system, this immediately leads to a multiple of P and consequently to the factorization of N. So a "curious" authority can factor N just asking the recovery of a single key! As a consequence the recovery agent must not reveal recovered keys but only decrypted messages.

With the RSA escrowing scheme, the problem is more subtle because the key obtained by the recovery agent are not meaningless since they are the factorization of a large number. Anyway, an attacker could try to use it as an oracle able to factor a modulus $n = pq$ if and only if $x = p + q - 1 < P$; this binary information can be used to recover the exact value of P. A dichotomic algorithm can

easily bound P in such a way that after $O(|P|_b)$ queries, the attacker recovers the factorization of N.

The Paillier scheme seems much more resistant to such attacks. Of course it is not secure against chosen ciphertext attacks since it is malleable. Furthermore, we cannot use a non-malleable version since we would no longer be able to make proofs. However, we do not know any attack able to recover a Paillier secret key by CCA; this is the main reason why we prefer to use this scheme and not the Okamoto-Uchiyama cryptosystem.

Note on threshold Paillier scheme. The other reason to use the Paillier scheme is that it is the only homomorphic cryptosystem related to be discrete log problem for which a threshold distributed version [11] is known. This may be of crucial importance for practical applications in order to reduce the trust in the recoverability agent.

References

1. N. Asokan, V. Shoup, and M. Waidner. optimistic Fair Exchange of Digital Signatures. In *Eurocrypt '98*, LNCS 1403, pages 591–606. Springer-Verlag, 1998.
2. F. Bao. An Efficient Verifiable Encryption Scheme for Encryption of Discrete Logarithms. In *CARDIS '98*, 1998.
3. J. Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Yale University, 1987. Available from `http://research.microsoft.com/~benaloh`.
4. D. Boneh and R. Venkatesan. Breaking RSA May Not Be Equivalent to Factoring. In *Eurocrypt '98*, LNCS 1403, pages 59–71. Springer-Verlag, 1998.
5. F. Boudot. Efficient Proofs that a Committed Number Lies in an Interval. In *Eurocrypt 2000*, LNCS 1807, pages 437–450. Springer-Verlag, 2000 (this volume).
6. J. Camenisch and M. Michels. A Group Signature Scheme with Improved Efficiency. In *Asiacrypt '98*, LNCS 1514. Springer-Verlag, 1998.
7. J. Camenisch and M. Michels. Proving in Zero-Knowledge That a Number Is the Product of Two Safe Primes. In *Eurocrypt '99*, LNCS 1592, pages 107–122. Springer-Verlag, 1999.
8. A. Chan, Y. Frankel, and Y. Tsiounis. Easy Come – Easy Go Divisible Cash. In *Eurocrypt '98*, LNCS 1403, pages 561–575. Springer-Verlag, 1998. Available as GTE Tech report.
9. H. Cohen. *A Course in Computational Algebraic Number Theory.* Graduate Texts in Mathematics 138. Springer-Verlag, 1993.
10. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, 1987.
11. PA. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In *Financial Cryptography 2000*, LNCS. Springer-Verlag, 2000.
12. E. Fujisaki and T. Okamoto. A Practical and Provably Secure Scheme for Publicly Verifiable Secret Sharing and Its Applications. In *Eurocrypt '98*, LNCS 1403, pages 32–46. Springer-Verlag, 1998.
13. M. Girault. Self-certified public keys. In *Eurocrypt '91*, LNCS 547, pages 490–497. Springer-Verlag, 1992.

14. M. Girault and J. Stern. On the Length of Cryptographic Hash-Values used in Identification Schemes. In *Crypto '94*, LNCS 839, pages 202–215. Springer-Verlag, 1994.
15. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28, 1984.
16. J. Kilian and F.T. Leighton. Fair Cryptosystems Revisited. In *Crypto '95*, LNCS 963, pages 208–221. Springer-Verlag, 1995.
17. D. Naccache and J. Stern. A New Public Key Cryptosystem Based on Higher Residues. In *Proc. of the 5th ACM-CCS*, pages 59–66. ACM press, 1998.
18. T. Okamoto and S. Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. In *Eurocrypt '98*, LNCS 1403, pages 308–318. Springer-Verlag, 1998.
19. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Eurocrypt '99*, LNCS 1592, pages 223–238. Springer-Verlag, 1999.
20. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt '96*, LNCS 1070, pages 387–398. Springer-Verlag, 1996.
21. J. M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143):918–924, July 1978.
22. G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In *Eurocrypt '98*, LNCS 1403, pages 422–436. Springer-Verlag, 1998.
23. G. Poupard and J. Stern. Short Proofs of Knowledge for Factoring. In *Proceedings of PKC2000*, LNCS 1751, pages 147–166. Springer-Verlag, 2000.
24. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
25. B. Vallée. Gauss' Algorithm Revisited. *Journal of Algorithms*, 12:556–572, 1991.
26. P. C. van Oorschot and M. J. Wiener. On Diffie-Hellman Key Agreement with Short Exponents. In *Eurocrypt '96*, LNCS 1070, pages 332–343. Springer-Verlag, 1996.
27. E. Verheul. Certificates of Recoverability with Scaleable Recovery Agent Security. In *Proceedings of PKC2000*, LNCS 1751. Springer-Verlag, 2000.
28. E. Verheul and H. van Tilborg. Binding ElGamal: A Fraud-Detectable Alternative to Key-Escrow Proposals. In *Eurocrypt '97*, LNCS 1233, pages 119–133. Springer-Verlag, 1997.
29. A. Young and M. Yung. Auto-Recoverable Auto-Certifiable Cryptosystems. In *Eurocrypt '98*, LNCS 1403, pages 17–31. Springer-Verlag, 1998.
30. A. Young and M. Yung. RSA-based Auto-Recoverable Cryptosystems. In *Proceedings of PKC2000*, LNCS 1751. Springer-Verlag, 2000.

A Security analysis of Diophantine commitment

In order to prove the exact security of Diophantine commitment, the approach of Feige, Fiat and Shamir is followed, first proving completeness, then soundness and, finally, the zero-knowledge property.

Theorem 3 (Completeness). *The execution of the protocol between an honest prover who knows the secret value* $x \in [0, S[$ *and a verifier is successful with probability higher than* $1 - \ell SB/A$.

Proof: If the prover knows a secret $x \in [0, S[$ and follows the protocol, he fails only if $y \geq A$ at some round of the proof. For any value $x \in [0, S[$ the probability of failure of such an event taken over all possible choices of r is smaller than SB/A. Consequently the execution of the protocol is successful with probability $\geq \left(1 - \frac{SB}{A}\right)^{\ell} \geq 1 - \frac{\ell SB}{A}$. □

Theorem 4 (Soundness). *Assume that some adversary $\widetilde{P}$ is accepted with probability $\varepsilon' = 1/B^{\ell} + \varepsilon$, $\varepsilon > 0$. Then there exists an algorithm which, with probability $> \varepsilon^2/(6\varepsilon'^2)$, outputs a pair (σ, τ) with $-A < \sigma < A$, $0 < \tau < B$ and $G^{\sigma} = \Gamma^{\tau}$ in $\mathcal{G}$. The expected running time is $< 2/\varepsilon \times \tau$, where τ is the average running time of an execution of the proof.*

Proof: Assume that some adversary, modeled as a Turing machine $\widetilde{P}(\omega)$ running on random tape ω, is accepted with probability $\varepsilon' = 1/B^{\ell} + \varepsilon$. We write $Succ(\omega, (e_1, ...e_{\ell})) \in \{true, false\}$ the result (successful of not) of the identification of $\widetilde{P}(\omega)$ when the challenges $e_1, ...e_{\ell}$ are used.

We consider the following algorithm (largely inspired from [24]):

Step 1. Pick a random tape ω and a tuple e of ℓ integers $e_1, ...e_{\ell}$ in $\{0, ..B-1\}$ until $Succ(\omega, e)$. Let u be the number of probes.

Step 2. Probe up to u random ℓ-tuples e' different from e until $Succ(\omega, e')$. If after the u probes a successful e' is not found, the algorithm fails.

Step 3. Let j be one of the indices such that $e_j \neq e_j{}'$; we note y_j and $y_j{}'$ the related correct answers of $\widetilde{P}$. If $e_j > e_j{}'$ the algorithm outputs $(\sigma, \tau) = (y_j - y_j{}', e_j - e_j{}')$, otherwise it outputs $(\sigma, \tau) = (y_j{}' - y_j, e_j{}' - e_j)$.

If this algorithm does not fails, the prover is able to correctly answer two challenges e_j and $e_j{}'$ for the same commitment t_j with the answers y_j and $y_j{}'$. This means that $G^{y_j}/\Gamma^{e_j} = t_j = G^{y_j{}'}/\Gamma^{e_j{}'}$ so $G^{\sigma} = \Gamma^{\tau}$. Furthermore, $|\sigma| < A$ and $0 < \tau < B$ because integers y_i and $y_i{}'$ are smaller than A and integers e_i and $e_i{}'$ are different and smaller than B.

We now analyze the complexity of the algorithm. By assumption, the probability of success of $\widetilde{P}$ is ε' so the first step finds ω and e with the same probability. The expected number E of repetitions is $1/\varepsilon'$ and the number u of probes is equal to N with probability $\varepsilon' \times (1 - \varepsilon')^{N-1}$.

Let Ω be the set of random tapes ω such that $\Pr_{e} \{Succ(\omega, e)\} \geq \varepsilon' - \varepsilon/2 = 1/B^{\ell} + \varepsilon/2$. The probability for the random tape ω found in step 1 to be in Ω conditioned by the knowledge that $Succ(\omega, e) = true$ can be lower bounded:

$$\Pr_{\omega,e} \{\omega \in \Omega | Succ(\omega, e)\} = 1 - \Pr_{\omega,e} \{\omega \notin \Omega | Succ(\omega, e)\} =$$

$$1 - \Pr_{\omega,e} \{Succ(\omega, e) | \omega \notin \Omega\} \times \frac{\Pr_{\omega,e} \{\omega \notin \Omega\}}{\Pr_{\omega,e} \{Succ(\omega, e)\}} \geq 1 - \left(\frac{1}{B^{\ell}} + \frac{\varepsilon}{2}\right) \times 1/\varepsilon' = \frac{\varepsilon}{2 \times \varepsilon'}$$

With probability $> \varepsilon/(2\varepsilon')$, the random tape ω is in Ω and in this case, by definition of the set Ω, the probability for a tuple of challenges $e' \neq e$ to lead

to success is $\geq \varepsilon/2$. The probability to obtain such a tuple e' after less than N probes is $\geq 1-(1-\varepsilon/2)^N$.

Consequently, the probability to obtain a random tape ω in Ω and to find e' is greater than

$$\frac{\varepsilon}{2\varepsilon'} \times \sum_{N=1}^{+\infty} (1-\varepsilon')^{N-1} \times \varepsilon' \times \left[1-(1-\frac{\varepsilon}{2})^N\right] = \frac{\varepsilon^2}{4\varepsilon'(\varepsilon'+\varepsilon/2-\varepsilon\times\varepsilon'/2)} > \frac{\varepsilon^2}{6{\varepsilon'}^2}$$

In conclusion, the algorithm finds (σ,τ) with probability $> \varepsilon^2/(6{\varepsilon'}^2)$ and the total expected number of executions of the proof between $\widetilde{P}$ and a verifier is smaller than $2/\varepsilon'$. □

Finally, in the complexity theory setting, let us consider a security parameter k. All the parameters A, B, S and ℓ are viewed as functions of k.

Theorem 5 (Zero-knowledge). *The protocol is statistically zero-knowledge if $\ell \times B$ is polynomial in k and if $\ell SB/A$ is negligible.*

Proof: We describe the polynomial time simulation of the communication between a prover P and a possibly dishonest verifier $\widetilde{V}$. We assume that, in order to try to obtain information about x, $\widetilde{V}$ does not randomly choose the challenges. If we focus on the ith round, $\widetilde{V}$ has already obtained data, noted $Data_i$, from previous interactions with P. Then the prover sends the commitment t_i and $\widetilde{V}$ chooses, possibly using $Data_i$ and t_i, the challenge $e_i(Data_i, t_i)$.

Here is a simulation of the ith round: choose random values ${e_i}' \in [0, B[$ and ${y_i}' \in [0, A[$, compute ${t_i}' = G^{{y_i}'} \times I^{{e_i}'}$. If $e_i(Data_i, {t_i}') \neq {e_i}'$ then try again with another pair $({e_i}', {y_i}')$, else return $({t_i}', {e_i}', {y_i}')$. It can be formally proved that such a simulation is statistically indistinguishable from the transcript of a real proof as soon as $\ell SB/A$ is negligible:

$$\sum_{(\alpha_i,\varepsilon_i,\beta_i),i\leq\ell} \left|\Pr\{(\alpha_i,\varepsilon_i,\beta_i)=(t_i,e_i,y_i)\} - \Pr\left\{(\alpha_i,\varepsilon_i,\beta_i)=({t_i}',{e_i}',y_i)'\right\}\right| < \frac{4\ell SB}{A}$$

Furthermore, the time complexity of the simulation if $O(\ell \times B)$ so the simulation runs in polynomial time in k if $\ell \times B$ is polynomial. □

Computing Inverses over a Shared Secret Modulus *

Dario Catalano[1], Rosario Gennaro[2], and Shai Halevi[2]

[1] Dipartimento di Matematica e Informatica,
Università di Catania. Viale A. Doria 6, 95125 Catania.
Email: catalano@dmi.unict.it.
[2] IBM T.J.Watson Research Center,
PO Box 704, Yorktown Heights, New York 10598, USA.
Email: {rosario,shaih}@watson.ibm.com

Abstract. We discuss the following problem: Given an integer ϕ shared secretly among n players and a prime number e, how can the players efficiently compute a sharing of $e^{-1} \bmod \phi$. The most interesting case is when ϕ is the Euler function of a known RSA modulus N, $\phi = \phi(N)$. The problem has several applications, among which the construction of threshold variants for two recent signature schemes proposed by Gennaro-Halevi-Rabin and Cramer-Shoup.
We present new and efficient protocols to solve this problem, improving over previous solutions by Boneh-Franklin and Frankel et al. Our basic protocol (secure against honest but curious players) requires only two rounds of communication and a single GCD computation. The robust protocol (secure against malicious players) adds only a couple of rounds and a few modular exponentiations to the computation.

1 Introduction

In this paper we consider the problem of computing a multiplicative inverse of a known prime number over a shared secret modulus. Specifically, given a known prime number e, and an integer ϕ shared secretly among n players, how can the players compute a sharing of $e^{-1} \bmod \phi$, without revealing anything about ϕ. The most interesting case is when ϕ is the Euler function of a known RSA modulus $\phi = \phi(N)$, since in this case the security of the RSA cryptosystem [22] is based on the assumption that $\phi(N)$ remains secret.

The most important applications of distributed modular inversion over a shared modulus are distributed RSA key generation, and distributing the new signature schemes of Gennaro-Halevi-Rabin [17] and Cramer-Shoup [9]. In particular, in the latter applications it is very important to have an efficient inversion

* Extended Abstract. A more complete version is available from http://www.research.ibm.com/security/dist-inv.ps. The first author's research was carried out while visiting the Computer Science Department of Columbia University.

protocol, since in these signature schemes the inversion operation is performed with a different exponent e for each message signed.

We present new and efficient protocols to solve the problem of inversion with a shared modulus. We first present a basic protocol which is only secure against honest but curious players. This protocol is extremely efficient as it requires only two rounds of communication and a single GCD computation on the part of the players. The protocol is also unconditionally secure (given a network of private channels). We then add robustness to the protocol in order to make it secure against malicious players. These modifications add only a couple of rounds and a few modular exponentiations to the computation. To overcome the difficulty of working over an unknown modulus, the protocols use computations over the integers. Some of the techniques developed to prove the security of the protocols may be of independent interest.

Previous work. Although our problem can in principle be solved using generic multiparty computation protocols [19, 3, 8], the resulting solutions would hardly be practical.

BONEH-FRANKLIN. The first to address the issue of an efficient solution for this problem were Boneh and Franklin, who in a breakthrough result show how $n > 3$ parties can jointly generate an RSA key without a trusted dealer [5]. In particular, as part of their solution they show how the parties jointly compute $d = e^{-1} \bmod \phi(N)$, where N, e are the RSA modulus and public exponent, respectively, and $\phi(N)$ is shared among the parties. Our solution improves on some of the features of the Boneh-Franklin protocol. In particular:

1. We only use a *single* invocation of the BGW [3] multiplication protocol, while their protocol needs two of them. Hence the round complexity of our protocol is half that of the protocol in [5].
2. The Boneh-Franklin protocol is based on an n-out-of-n solution where a single crash could prevent the protocol from completing.[1] To obtain a t-out-of-n solution, they suggest using the "share-backup" approach of Rabin [21], but this approach has some known problems. For one thing, it incurs the overhead of multiple layers of (verifiable) secret-sharing. Moreover, it requires that the "good parties" recover the secret information of a party who may simply be temporarily disconnected.
   In contrast, our solution achieves *directly* a t-out-of-n threshold, using polynomial sharings and secret computations over the integers. Some of the most interesting technical contribution of our work are in the security proofs of these secret computations over the integers.
3. The Boneh-Franklin results are presented only in the honest-but-curious model while we are also able to present robust solutions that tolerate malicious players.

[1] In their setting, this is the natural solution, since they also generate the modulus so that it is shared n-out-of-n. Indeed, to use our solution in their setting, one would have to implement also methods for generating and using the modulus in a t-out-of-n fashion.

4. In an updated version of [5], some other solutions are presented. One of them is particularly efficient since it avoids costly increases in the size of the shares. However, to achieve this efficiency, the proposed solution leaks a few bits of information about $\phi(N)$. Although this is acceptable for a protocol that is invoked only once (since those few bits could be guessed anyway by an adversary), it is not clear what happens when the protocol is invoked several times with the same $\phi(N)$ (as in our signature applications). Hence, we designed our protocols so that they do not leak any information about $\phi(N)$, in a strong, statistical, sense. (This requires some increase in the size of the shares, though.)

FRANKEL-MCKENZIE-YUNG. Building on the Boneh-Franklin solution, Frankel, Mc Kenzie and Yung describe in [14] a way to add robustness to the protocols in [5], and in particular how to add robustness to the inversion protocol. The FMY protocol follows the structure of [5], so it also needs two invocations of the BGW multiplication protocol. Moreover in order to achieve a t-out-of-n threshold, the FMY protocol uses *representation changes* for the sharing of the secret data. Namely, data which is shared in a t-out-of-n fashion is converted into a t-out-of-t fashion in order to perform computations, and then re-converted into a t-out-of-n sharing to preserve tolerance of crashing or malicious players. The complexity of the representation change is quite high, making the combined protocol much less efficient. Although the complexity of this protocol is acceptable for the task of distributed RSA key generation, where the protocol is only run once, it is too high for a protocol that must be efficiently run many times, as in the case of the signature applications. We avoid this efficiency cost, by keeping the data always in a t-out-of-n representation.

OTHERS. Some of the techniques that we use in this work originated in papers over robust and proactive RSA. In particular, working over the integers in order to overcome the difficulty of computing modulo an unknown integer was used in several previous papers [13, 18, 12, 21]. Finally, the "dual" problem of computing $x^{-1} \bmod p$ where p is known and x is shared was discussed in [2].

2 Preliminaries

THE NETWORK MODEL. We consider a network of n players, that are connected by point-to-point private channels and by a broadcast channel.[2] We model failures in the network by an adversary $\mathcal{A}$, who can corrupt at most t of the players. We distinguish between the following types of "failures":

- *honest but curious:* the adversary can just read the memory of the corrupted players but not modify their behavior;

[2] The communication assumptions allow us to focus on a high-level description of the protocols, and they can be eliminated using standard techniques for privacy, authentication, commitment and agreement.

- *halting:* an "honest but curious" adversary who may also cause any of the corrupted players to crash and abort the protocol;
- *malicious:* the adversary may cause players to deviate arbitrarily from the protocol.

We assume for simplicity that the adversary is static, i.e. the set of corrupted players is decided at the beginning of the computation of a protocol.[3] We assume communication is synchronous, except that we allow *rushing* adversaries (i.e. adversaries who decide the messages of the bad players at round R after having seen the messages of the good players at the same round).

2.1 Definitions

NOTATIONS. In the following we denote the shared secret modulus by ϕ, and by N we denote an approximate bound on ϕ, which must be known in the protocol (in the typical RSA application, we can use the public modulus N as a bound on $\phi = \phi(N)$). We also denote by L the factorial of n (the number of players), i.e. $L = n!$

A Modular Inversion Protocol is an n-player protocol, where as an input to the protocol the players share a secret modulus ϕ (with player P_i having the share ϕ_i), and all the players know public inputs e (a prime number) and N (an approximate bound on ϕ). At the end of the protocol, each player P_i has a secret output d_i, which would be its share of the modular inverse $d = e^{-1} \bmod \phi$.

CORRECTNESS. We say that a Modular Inversion Protocol is *correct* if the output values $d_1, \ldots, d_n$ constitute a t-out-of-n secret sharing of $d = e^{-1} \bmod \phi$

PRIVACY. We define privacy using the usual simulation approach. That is, we consider the view of the adversary $\mathcal{A}$ during a protocol to be the set of messages sent and received by the bad players during a run of the protocol. We say that a Modular Inversion Protocol is *private* if for any adversary $\mathcal{A}$ there exist a simulator S that runs an execution of the protocol together with $\mathcal{A}$ and produces for it a view that is indistinguishable from the real one.

SECURITY. We say that a Modular Inversion Protocol is *secure* if it is correct and private.

Remark 1 (Trusted dealer) In the above definition and in the presentation of the protocols, we implicitly assume that the modulus ϕ is already shared among the players using an appropriate t-out-of-n scheme. Specifically, for our protocols we always assume that this sharing is done over the integers, with shares from some appropriately large domain. In some cases we also assume that commitments to the shares of all the players are publicly known (see Section 5.2). The exact sharing formats of ϕ that we need are stated explicitly in the description of the protocols.

[3] It is possible to use recent techniques by Canetti et al. [6] to make our protocols secure against adaptive adversaries who corrupt players at any stage during the protocol.

These assumptions can be made formal by including the initialization phase in the protocol definition, and analyzing the protocols under the assumption that this initialization is done by a trusted dealer. However, we feel that such a formulation will only distract attention from the focus of this paper, which is the inversion protocol. In Section 7 we briefly touch on the subject of eliminating the trusted dealer and instead having the n players jointly initialize the system.

3 The Basic Idea

We begin with a very simple protocol which, although doesn't quite solve our problem, is nonetheless useful for illustrating the basic ideas and techniques behind our solution. In particular, this protocol only works for n-out-of-n sharing (i.e. although it tolerates coalitions of $n-1$ honest but curious players, it does not tolerate even a single crashing player).

For this protocol, some multiple of the secret modulus ϕ is shared additively between the players. That is, each player P_i holds a value α_i such that $\sum_i \alpha_i = \lambda\phi$, where λ is a random integer, much larger than ϕ (say, of order $O(N^2)$). In the inversion protocol, each player P_i chooses a "randomizing integer" $r_i \in_R [0..N^3]$, and broadcasts the value $\gamma_i = \alpha_i + r_i e$, and all the players compute $\gamma = \sum_i \gamma_i$. Clearly, we have

$$\gamma = \sum_i \gamma_i = \sum_i \alpha_i + r_i e = \lambda\phi + Re$$

(where $R = \sum_i r_i$). Assuming that $GCD(\gamma, e) = 1$, there exist a, b such that $a\gamma + be = 1$ and thus $d = aR + b = e^{-1} \bmod \phi$. Additive shares of d can be easily obtained by having player P_1 sets $d_1 = ar_1 + b$, and the other players set $d_i = ar_i$. Clearly $d = \sum_i d_i$.

It is not hard to see that the only information leaked by the protocol is the value $\gamma = \lambda\phi + Re$. But it is possible to prove that the distribution of γ is (almost) independent of ϕ. Specifically, it can be shown that when λ and R follow the probability distribution described above, then the distributions $\{\gamma = \lambda\phi + Re\}$ and $\{\gamma' = \lambda N + Re\}$ are statistically close (up to $O(1/N)$).

It should be noted that the above protocol is not secure when it is used more than once with the same λ and different e's. Indeed, for each input e the protocol leaks the value $\lambda\phi \bmod e$, and so after sufficiently many runs with different e's we can then recover the integer $\lambda\phi$ via the Chinese Remainder Theorem. To overcome this, it is necessary to use a "fresh" λ for each input e. In the next section we show how to do this, and at the same time get a t-out-of-n threshold solution (but still in the "honest but curious" model).

4 The honest-but-curious case

The protocol in this section achieves t-out-of-n sharing. It assumes the "honest but curious" model, in which players do not deviate from the protocol but simply pool together their data to try to gain information (in this model we need $n > 2t$).

It also tolerates crashing faults, i.e. players who suspend their participation in the protocol (in this case we need $n > 3t$). In the next section we show how to add robustness to this protocol (i.e. tolerance of maliciously faulty players).

The difference between this protocol and the one in the previous section is that all the secrets are shared via polynomials (rather than sums), and the multiple λ is chosen afresh with each execution. The rest of the protocol is similar to the basic case. The protocol is described in detail in Figure 1. On a high-level the protocol goes as follows:

- Each player starts with input a share of the secret modulus ϕ (multiplied by a factor of $L = n!$ for technical reasons), via a t-degree polynomial $f(z)$ with free term $L\phi$.
- In the first round of the protocol, the players jointly generate two random t-degree polynomials $g(z)$ and $h(z)$ with free terms $L\lambda$ and LR, respectively, and a random $2t$-degree polynomial $\rho(z)$ with free term 0.
- In the second round they reconstruct the $2t$-degree polynomial $F(z) = f(z)g(z) + e \cdot h(z) + \rho(z)$ and recover its free term $\gamma = F(0) = L^2\lambda\phi + LRe$.
- Finally, they use the GCD algorithm to compute a, b such that $a\gamma + be = 1$ and set $d = aLR + b = e^{-1} \bmod \phi$. Each player P_i computes its share of d by setting $d_i = ah(i) + b$.

Theorem 1. *If all the players carry out the prescribed protocol and $n > 2t$ ($n > 3t$ for the case of crashing faults) then the protocol in Figure 1 is a secure Modular Inversion Protocol according to the Definition in Section 2.1.*

The proof follows a standard simulation argument, and is described in the full version of this paper. The crucial part of the proof is to prove that $\lambda\phi + Re$ can be statistically approximated by the simulator without knowing ϕ.

Remark 2 (Size of shares) Note that the shares d_i of $d = e^{-1} \bmod \phi$ have order $O(N^5)$. If the d_i's are used as exponents (as in the threshold signature applications we discuss in Section 6), this results in a factor of five slowdown during the generation of the signature. However, the shares do *not* have to be this large. We chose these bounds to make the presentation and the proof simpler. It is possible to improve (a lot) on those bounds as we discuss in Section 7.

5 A Robust Solution

We show how to deal with a malicious adversary who may corrupt up to t players and make them behave in any arbitrary manner. We use some standard techniques like:

- Replace the simple secret-sharing of the first round with Verifiable Secret Sharing (VSS) a-la-Pedersen [20], to make sure that the players perform correct sharings;

Inversion Protocol for Honest-but-Curious players

Private inputs: Sharing of $L\phi$ using a t-degree polynomial over the integers. Player P_i has private input $f_i = f(i)$, where $f(z) = L\phi + a_1 z + \ldots + a_t z^t$, and $\forall j, a_j \in [-L^2N, L^2N]$.

Public input: prime number $e > n$, an approximate bound N on ϕ.

[**Round 1**] Each player P_i does the following:

1. Choose $\lambda_i \in_R [0 \ldots N^2]$, and $b_{i,1}, \ldots, b_{i,t} \in_R [-L^2N^3 \ldots L^2N^3]$,
   Choose $r_i \in_R [0 \ldots N^3]$, and $c_{i,1}, \ldots, c_{i,t} \in_R [-L^2N^4 \ldots L^2N^4]$
   Choose $\rho_{i,1}, \ldots, \rho_{i,2t} \in_R [-L^2N^5 \ldots L^2N^5]$
2. Set $g_i(z) = L\lambda_i + b_{i,1}z + \ldots + b_{i,t}z^t$, $h_i(z) = Lr_i + c_{i,1}z + \ldots + c_{i,t}z^t$, and $\rho_i(z) = 0 + \rho_{i,1}z + \ldots + \rho_{i,2t}z^{2t}$.
3. Send to each player P_j the values $g_i(j), h_i(j), \rho_i(j)$, computed over the integers.

[**Round 2**] Each player P_j does the following:

1. Set $g_j = \sum_{i=1}^n g_i(j)$, $h_j = \sum_{i=1}^n h_i(j)$, and $\rho_j = \sum_{i=1}^n \rho_i(j)$.
   (These are its shares of the polynomials $g(z) = \sum_i g_i(z)$, $h(z) = \sum_i h_i(z)$, and $\rho(z) = \sum_i \rho_i(z)$.)
2. Broadcast the value $F_i = f_i g_i + e h_i + \rho_i$

[**Output**] Each player P_i does the following:

1. From the broadcast values interpolate the $2t$-degree polynomial $F(z) = f(z)g(z) + e \cdot h(z) + \rho(z)$.
2. Using the GCD algorithm, find a, b such that $aF(0) + be = 1$. If no such a, b exist, go to Round 1.
3. The inverse of e is $d = ah(0) + b$. Privately output the share of the inverse, $d_i = ah(i) + b$.

Fig. 1. Computing inverses in the all-honest case

– Use error-correcting codes or zero-knowledge proofs to combat malicious players who may contribute incorrect shares for the reconstruction of the polynomial $F(z)$ in Round 2.

A few technical complications arise from the fact that we use secret sharing over the integers. Some are solved using known techniques that were developed for robust and proactive RSA [15, 12, 21, 7], others require some new machinery.

5.1 Pedersen's VSS revisited

The problems that we need to tackle is how to ensure that the secrets are shared correctly in Round 1 and recovered correctly in Round 2. For the first problem, we use a variant of Pedersen's Verifiable-Secret-Sharing protocol [20], adjusted to account for the fact that we share these secrets over the integers.

In Pedersen's scheme the secret and the shares are viewed as "indices" for some cyclic group $\langle g \rangle$. Hence, there is an efficient mapping between shares and group elements $x \mapsto g^x$, and the players use the group operation to verify various properties of the shares. There are, however, two problems with using this approach in our setting:

– In our setting, it is imperative that the secrets satisfy some equations over the integers, and not just modulo the order of g. (For example, it would be useless if the shares of $d = e^{-1} \bmod \phi$ would interpolate to $d + ord(g)$ over the integers.)
– Pedersen's protocol does not provide tools to prove that the shared secret is "small enough", whereas the secrecy of our protocol relies on the fact that we know some bound on the size of the secrets. (For example, if the size of λ in $\gamma = \lambda\phi + Re$ is much larger than other terms, then clearly γ reveals information about ϕ.)

Overcoming the second problem is easy. Each player simply checks that its shares are bounded in some interval, and then we show that the secret must also be bounded in some (slightly larger) interval. Solving the first problem is a little harder. We propose two solutions to this problem, each with its own advantages and drawbacks:

– *Work with a group of unknown order.* If the order of g is not known, then it would be potentially hard for the dealer to arrange that some relations hold modulo $ord(g)$ but not over the integers. More specifically, we show that when Pedersen's protocol is executed over an RSA modulus $M = pq$, which is a product of two safe primes ($p = 2p' + 1, q = 2q' + 1$ with p, p', q, q' all primes), then it is indeed a secure VSS under the strong-RSA assumption (see below).
  An advantage of this solution is that the modulus M is independent of the bound on the size of the secrets and shares, and so a smaller M can be used. The drawback is that we must work in a system where such an RSA modulus of unknown factorization is available, and that we use the strong-RSA assumption, which is stronger than, say, plain RSA or discrete-log. Still,

for the main applications of our result (constructing threshold versions of the signature schemes described in [17, 9]), these drawbacks do not matter, since those signature schemes already use these special-form RSA moduli and are based on the strong-RSA assumption.

– *Work with a very large group.* Another option would be to make the order of g much larger than all the other parameters of the system. This way, if the players verify that the size of their shares is "small enough" then any relation that holds modulo $ord(g)$ must also hold over the integers, simply because the numbers involved can never be large enough to "wrap around" $ord(g)$.
It is therefore possible to use Pedersen's original protocol modulo a large prime, provided that all the players check the size of their shares[4] and the prime is large enough. Specifically, if there are n players, and each player verifies that its share is smaller than some known bound B, then it is sufficient to work over a prime $p > tn^t n!B$.

The second solution above is pretty straightforward, and will be described in the full version of the paper. Below we only describe the details of the first solution. For this solution, we have a public modulus M of unknown factorization, which is a product of two safe primes ($M = pq$, $p = 2p' + 1$, $q = 2q' + 1$). For such a modulus, the squares form a cyclic subgroup of Z_M^* of order $p'q'$. We let $G, H \in Z_N^*$ to be two random squares which generate the squares subgroup and we assume that nobody knows the discrete log of H with respect to G. The protocol is spelled out in Figure 2.

THE STRONG-RSA ASSUMPTION. This assumption was introduced in [1] and subsequently used in several other works [15, 17, 9]. It conjectures that given a random square $G \in Z_M^*$ there exists no polynomial time algorithm that can compute $H \in Z_M^*$ and an integer $x \neq 1$ such that $H^x = G \bmod M$.

Lemma 1. *Under the Strong-RSA assumption, the protocol* PedVSS *is a VSS against an adversary who corrupts at most t players when $n > 2t$.*

The reduction from the security of PedVSS (over the integers) to Strong-RSA follows an approach presented first in [15].

Remark 3 (Share size check) The security proof of PedVSS does not require that players check the size of their shares in Step 4. This check however guarantees the good players that the shared secret is bounded by $t^2 n^t L^3 M \beta$ (since the Lagrange interpolation formula tells us that the secret can be written as the linear combination of $t + 1$ shares with coefficients all smaller than L).

Remark 4 (Sharing a known value) In the robust protocol we use the protocol PedVSS to share either a secret unknown value, or the value 0. The latter is used to randomize the product polynomial in the multiplication step.

[4] Note that in Pedersen's protocol, the shares and secrets are committed to by setting $C(x) = g^x h^r \bmod P$ for a random r. In our setting, the players would have to check that the "real share" x is in the allowed interval, but the randomizing element r can be any element in Z_{p-1}.

PedVSS

Dealing Phase

Public Input: RSA modulus M (product of two safe primes), two random squares $G, H \in Z_M^*$, and a bound β.
Input for the dealer: A secret $\lambda \in [0..\beta]$.

1. The dealer chooses $\hat{\lambda} \in_R [0..\beta]$ and $b_1, \ldots, b_t, \hat{b}_1, \ldots, \hat{b}_t \in_R [-L^2 M\beta..L^2 M\beta]$.
   Sets $h(z) = L\lambda + b_1 z + \ldots + b_t z^t$ and $\hat{h}(z) = L\hat{\lambda} + \hat{b}_1 z + \ldots + \hat{b}_t z^t$.
   Sends privately to player P_i the values $h(i)$ and $\hat{h}(i)$ computed over the integers.
   Broadcasts publicly the values $C_0 = G^{\lambda} H^{\hat{\lambda}} \bmod M$ and $C_j = G^{b_j} H^{\hat{b}_j} \bmod M$ for $j = 1, \ldots, t$.
2. Player P_i checks that

$$G^{h(i)} H^{\hat{h}(i)} = \prod_{j=0}^{t} (C_j)^{i^j} \bmod N \tag{1}$$

   If the check fails, P_i complains publicly. If more than t players complain the dealer is disqualified.
3. If the dealer is not disqualified, it reveals the values $h(i), \hat{h}(i)$ satisfying Equation (1) for the players P_i who complained at the previous step. If the dealer does not perform this step correctly it is disqualified.
4. Player P_i checks that the values it received and the values broadcasted by the dealer in the previous steps are integers bounded in absolute value by $tn^t L^2 M\beta$. If the check fails, P_i exposes its share. If an exposed share is larger than $tn^t L^2 M\beta$ and matches Equation (1) then the dealer is disqualified.[a]

Reconstruction Phase

1. Each player P_i reveals $h(i), \hat{h}(i)$. Only the values satisfying Equation 1 will be accepted.
   Interpolate $t+1$ of those values to reconstruct $h(z)$ over the rationals and output the secret $\lambda = h(0)$.

[a] This step is not needed for this protocol to be a "secure VSS protocol", see Remark 3.

Fig. 2. Pedersen's VSS

5.2 The robust solution

The main change from the honest-but-curious to the robust solution is that all the secrets are now shared using our variant of Pedersen's VSS. The full protocol is described in Figure 3. In this description we distinguish between two cases: $n > 4t$ or $3t < n \leq 4t$.

When $n > 4t$ we can use error-correcting codes to interpolate the polynomial $F(z)$ (e.g., using the Berlekamp-Welch algorithm [4] or see for example the appendix in [24]).

For the case of $3t < n \leq 4t$ we do not have enough points to do error-correction, so we identify and sieve out the bad shares by having each player P_i proves in zero knowledge that its value $F(i)$ is the correct one. In the latter case, we need the players to have as public input commitments to the coefficients of the polynomial $f(z)$ (that is used to share $L\phi$), and we use these commitments in the zero-knowledge proofs. The ZK proof (described in detail in Appendix A) is a 3-round, public-coin, honest-verifier statistical ZK proof. When this ZK proof is executed in the distributed protocol above, each player will run it once as the prover. The verifier's challenge will be jointly generated by the other $n-1$ servers. It is shown by Canetti et.al. [6] that it is sufficient that the protocol is only honest-verifier ZK since each prover runs the protocol against a "virtual" verifier which is implemented by the other $n-1$ players. This virtual verifier will be forced to act honestly because a majority of the other players is honest.

Remark 5 (N versus M) If the value N is already an RSA modulus, product of two strong primes, then in Robust Protocol it is possible to set $M = N$. This is indeed the case in most of our applications.

Theorem 2. *Under the Strong-RSA assumption, if the dealer is honest and $n > 3t$, then* Robust Protocol *is a secure Modular Inversion Protocol (according to the Definition in Section 2.1) in the presence of a malicious adversary who corrupts at most t players.*

6 Applications

The main application of our result is the construction of threshold variants for two recently proposed signature schemes [17, 9]. Let us briefly recall the concept of threshold cryptography (which originates in a paper by Desmedt [10]). In a threshold signature scheme n parties hold a t-out-of-n sharing of the secret key SK for a signature scheme. Only when at least $t+1$ of them cooperate they can sign a given message. It is very important however that the computation of such signature is performed without exposing any other information about the secret key; in particular the players cannot reconstruct SK and use the signing algorithm, but must use their shares *implicitly* in a communication protocol which outputs the signature. A large body of research has been done on threshold signature schemes: for lack of space we refer the reader only to two literature surveys [11, 16].

Robust Protocol

Private inputs: Sharing of the number $L\phi$ using a t-degree polynomial over the integers. Player P_i has private input $f_i = f(i)$, where $f(z) = L\phi + a_1 z + \ldots + a_t z^t$, and $\forall j, a_j \in [-L^2N, L^2N]$. If $3t < n \leq 4t$ then P_i also has $\hat{f}_i = \hat{f}(i)$, where $f(z) = \hat{a}_0 + \hat{a}_1 z + \ldots + \hat{a}_t z^t$, and $\forall j, \hat{a}_j \in_R Z_M$.

Public input: prime number $e > n$, and an approximate bound N on ϕ. An RSA modulus M (product of two safe primes), and two random squares $G, H \in Z_M^*$. If $3t < n \leq 4t$ then also commitments $G^{a_j} H^{\hat{a}_j}$.

[**Part 1**] Each player P_i chooses $\lambda_i \in_R [0 \ldots N^2]$, and $r_i \in [0..N^4]$, and does the following:

1. Use PedVSS to share λ_i with bound N^2 and t-degree polys $g_i(z)$ and $\hat{g}_i(z)$.
2. Use PedVSS to share r_i with bound N^4 and t-degree polys $h_i(z)$ and $\hat{h}_i(z)$.
3. Use PedVSS to share 0 with bound N^6 and $2t$-degree polys $\rho_i(z)$ and $\hat{\rho}_i(z)$.

Let A be the set of players who were not disqualified in Round 1, denote $\lambda = \sum_{i \in A} \lambda_i$, $R = \sum_{i \in A} r_i$. Also denote

$$g(z) = \sum_{i \in A} g_i(z),\ h(z) = \sum_{i \in A} h_i(z),\ \rho(z) = \sum_{i \in A} \rho_i(z)$$

$$\hat{g}(z) = \sum_{i \in A} \hat{g}_i(z),\ \hat{h}(z) = \sum_{i \in A} \hat{h}_i(z),\ \hat{\rho}(z) = \sum_{i \in A} \hat{\rho}_i(z)$$

[**Part 2**] Each player P_j does the following

1. Generates its shares of the polynomials $h(z), g(z), \rho(z)$ by summing the shares that were received in Part 1 from players in A.
   If $3t < n \leq 4t$, also generates its shares of the polynomials $\hat{h}, \hat{g}, \hat{\rho}$ similarly.
2. Calculates $F_j = f(j)h(j) + eg(j) + \rho(j)$, and broadcasts F_j as its share of the $2t$-degree polynomial $F(z) = f(z)h(z) + eg(z) + \rho(z)$.

Notice that the free term of $F(z)$ is the integer $F(0) = L^2\lambda\phi + LRe$.

[**Part 3**] We distinguish two cases:

1. If $n > 4t$ then the players interpolate over the rationals, using error-correction, the unique polynomial $F(z)$ of degree $2t$ passing through $n - t$ of the broadcasted points, and set $\gamma = F(0)$.
2. If $3t < n \leq 4t$, each player P_i proves that the value F_i is correct using the subprotocol Prove-Correct described in Appendix A).
   The players interpolate the unique polynomial $F(z)$ of degree $2t$ passing through the broadcasted points which are proven correct, and set $\gamma = F(0)$.

[**Output**]

1. Using the GCD algorithm, each player computes two values a, b such that $aF(0) + be = 1$. If no such a, b exist, return to Part 1.
2. Each player P_i privately compute its share of the inverse, $d_i = ah(i) + b$.

Fig. 3. Computing inverses in the malicious case

THRESHOLD GHR SIGNATURES. In [17] Gennaro, Halevi and Rabin present a new signature scheme which is secure under the Strong-RSA assumption. The scheme works as follows. The public key of the signer is an RSA modulus N, product of two safe primes p, q, and a random element $s \in Z_N^*$. To sign a message m, the signer first hashes it using a suitable hash function H to obtain $e = H(m)$ and then computes $\sigma(m)$ such that $\sigma(m)^e = s \bmod N$. We refer the reader to [17].

Using our Modular Inversion Protocol, we can create a threshold version for the GHR scheme as follows. A trusted dealer can initialize the system by choosing N and sharing $\phi(N)$ as needed in our solution(s) (either the honest-but-curious or the robust one depending on the model). For a reason that will be soon apparent, the dealer also chooses s as follows: pick a random square $s_0 \in Z_N^*$ and compute $s = s_0^{L^2} \bmod N$ and make s_0, s public.

Then for each message m to be signed, the players publicly compute $e = H(m)$ and perform an execution of the inversion protocol, to obtain shares d_i of $d = e^{-1} \bmod \phi(N)$. Recall that each d_i is the point $ah(i) + b$ on a t-degree polynomial $ah(z) + b$ whose free term is d. It follows then that for any subset T of $t + 1$ shares we can write

$$d = \sum_{i \in T} \mu_{i,T} \cdot d_i$$

where $\mu_{i,T}$ are the appropriate Lagrange interpolation coefficients. Notice that the above equation is taken over the rationals, so $\mu_{i,T}$ may be fractions. However because we are always interpolating integer points in the set $\{1, \ldots, n\}$ we have that $L^2 \cdot \mu_{i,T}$ is always an integer. The protocol is concluded by having each player reveal $s_i = s_0^{d_i}$. Then

$$\sigma(m) = s^d = s_0^{L^2 \sum_{i \in T} \mu_{i,T} \cdot d_i} = \prod_{i \in T} s_i^{L^2 \cdot \mu_{i,T}}$$

and the exponents are all integers.

In the case of malicious players, a zero-knowledge proof must be added that s_i is the correct value. Notice that if $n > 4t$ we can still use error-correcting codes inside the inversion protocol, but we do not know how to do error-correction "in the exponent" for the s_i's and thus the ZK proof for this step is required also when $n > 4t$. An efficient ZK proof similar to Prove-Correct (see Appendix A) can be implemented using the public information generated by the inversion protocol. More specifically, the inversion protocol generates public commitments $C_i = G^{d_i} H^{\hat{d}_i}$ to the d_i's. When P_i reveals $s_i = s_0^{d_i}$ it proves that the discrete log of s_i in base s_0 is the same as the opening he knows of the commitment C_i.

A couple of remarks are in order. Because of the way we generate s it is obvious that any message m whose hash value is in the set $\{1, \ldots, n\}$ can be forged, so we need to require that $H(m) > n$ for all messages. This is not a problem as [17] already assumes that $e = \Theta(N)$. Also in one of the variations presented in [17] the hash function is randomized, i.e. $e = H(m, \rho)$ where ρ is a random string which is then attached to the signature for verification purpose.

In this case the inversion protocol must be preceded by a coin flipping protocol by the n players to generate ρ.

THRESHOLD CRAMER-SHOUP SIGNATURES. In [9] Cramer and Shoup presented the following signature scheme. The signer public key is N (the product of two safe primes p, q), two random squares $h, x \in Z_N^*$ and a random prime e' sufficiently long (say 160 bits). To sign a message m, the signer generates a new prime $e \neq e'$ (also of length 160 bits) and a random square $y' \in Z_N^*$. Two values x', y are then computed as

$$x' = \frac{{y'}^{e'}}{h^{H(m)}} \bmod N \quad \text{and} \quad y = \left(x h^{H(x')}\right)^{1/e} \bmod N$$

where H is a collision-resistant hash function. The signature is (e, y, y')

A threshold version of the Cramer-Shoup signature scheme is obtained in the same way as the threshold GHR scheme, since the only part that involves the secret key is the computation of y (here also, for the same reason as above, the dealer must choose h, x as $h = h_0^{L^2} \bmod N$ $x = x_0^{L^2} \bmod N$, and make public the values h_0, x_0). The only difference is that here the prime e must be generated by the players instead of being publicly computed via a hash function, and the requirement is that the signers never use the same prime e for two different messages. This can be done either by having the players together generate randomness and use it for prime generation, or by having one player choose e, and the others just check that it was never used before. (For the latter solution the players need to keep state, and there must be some protocol to keep this state "synchronized" between different players).

7 Conclusions

We presented new protocols to compute a sharing of the inverse of a public integer e modulo a shared secret ϕ. We also presented applications to construction of threshold variants for two newly proposed signature schemes. Our result was constructed with these specific applications in mind, and we focused on protocols which would minimize the round complexity (i.e. the interaction between servers). This is the main improvement with respect to previous solutions from [5, 14].

We conclude with some remarks.

A Note on the Assumptions Used. In this extended abstract we focused on a robust solution to the modular inversion problem which requires the Strong-RSA assumption and the generation of "safe" primes. This solution is the more natural one to use for the applications presented in Section 6 which already have such requirement. We would like to stress however that the Strong RSA assumption and the generation of safe primes is needed only for this variant of the protocol. As we mentioned before, by using Pedersen's VSS over a large prime field it is possible to construct a robust Modular Inversion Protocol based only on the Discrete Log assumption. That is, it is possible to state and prove

an analogous to Theorem 2 assuming only that computing discrete logs is hard. Details will appear in the final paper.

A Note on Efficiency. To simplify the presentation, we did not focus on keeping the size of the integers used in our computations as small as possible. It is however possible to reduce the size of the integers: this is particularly important for the share d_i's which are used as exponents in our threshold signature applications.

The main reason for the increase in size of the integers is that our proofs use $\log N$ as our security parameter (i.e. we define a quantity to be negligible if it is smaller than $1/N$). If instead we were to choose a different security parameter k (and define negligible anything smaller than 2^{-k}), then part of the growth in the size of the shares would be in multiplicative factors of 2^k rather than N. In particular the real bound on the size of the shares d_i is $O(N^2 2^{3k})$ for the honest-but-curious case, and $O(N^2 2^{4k})$ for the malicious adversary case. For reasonable choices of the parameters (say $k = 100$ and $\log N = 1000$) this is even less that $O(N^3)$, so the threshold signature protocols proposed in Section 6 are slower by less than a factor of 3 than the centralized one.

It would be interesting to come up with different protocols (or proof techniques for our protocol) that further reduce this size.

On the Trusted Dealer. Throughout the paper we implicitly assumed that the input for our protocols (i.e., the sharing of ϕ) was generated by a trusted dealer. In some cases this assumption can be eliminated by having the players generate ϕ cooperatively. For example, for the applications in which $\phi = \phi(N)$ for an RSA modulus N we can use the first part of the Boneh-Franklin result [5] to have the players jointly generate N and share $\phi(N)$ among them. Notice that [5] cannot be used to generate a product of two safe primes, so in this case we must use the discrete-log based robust solution.

Acknowledgment. We would like to thank Don Coppersmith for helpful discussions. We also thank the Eurocrypt committee for their suggestions and comments.

References

1. N. Barić, and B. Pfitzmann. Collision-free accumulators and Fail-stop signature schemes without trees. In *Advances in Cryptology - Eurocrypt '97*, LNCS vol. 1233, Springer, 1997, pages 480-494.
2. J. Bar-Ilan, and D. Beaver. Non-Cryptographic Fault-Tolerant Computing in a Constant Number of Rounds. In *Proceedings of the ACM Symposium on Principles of Distributed Computation*, pp.201–209, 1989.
3. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-cryptographic Fault-Tolerant Distributed Computations. *20th ACM Symposium on the Theory of Computing*, pp.1–10, ACM Press, 1988.
4. E. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent 4,633,470.

5. D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. In *Advances in Cryptology - Crypto '97*, LNCS vol. 1294, Springer, 1997, pages 425-439. Extended version available from `http://crypto.stanford.edu/~dabo/pubs.html`
6. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Adaptive Security for Threshold Cryptosystems. In *Advances in Cryptology - Crypto '99*, LNCS vol. 1666, Springer, 1999, pages 98-115.
7. D. Catalano and R. Gennaro. New Efficient and Secure Protocols for Verifiable Signature Sharing and Other Applications. In *Advances in Cryptology - Crypto '98*, LNCS vol. 1462, Springer, 1998, pages 105-120.
8. D. Chaum, C. Crepeau, and I. Damgard. Multiparty Unconditionally Secure Protocols. *20th ACM Symposium on the Theory of Computing*, pp.11–19, ACM Press, 1988.
9. R. Cramer and V. Shoup. Signature Schemes Based on the Strong RSA Assumption. To appear in the Proceedings of the *6th ACM Conference in Computer and Communication Security*, 1999.
10. Y. Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *Advances in Cryptology–CRYPTO'87*, Lecture Notes in Computer Science Vol. 293, pp. 120–127, Springer-Verlag, 1988.
11. Y.G. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–457, July 1994.
12. Y. Frankel, P. Gemmell, P. Mackenzie, and M. Yung. Optimal Resilience Proactive Public-Key Cryptosystems. *38th IEEE Symposium on the Foundations of Computer Science*, pp.384–393, IEEE Computer Society Press, 1997.
13. Y. Frankel, P. Gemmell, and M. Yung. Witness-based Cryptographic Program Checking and Robust Function Sharing. *28th ACM Symposium on the Theory of Computing*, pp.499–508, ACM Press, 1996.
14. Y. Frankel, P. Mackenzie, and M. Yung. Robust Efficient Distributed RSA-Key Generation. In *STOC* 1998, pp.663–672.
15. E. Fujisaki and T. Okamoto. Statistical Zero-Knowledge Protocols to Prove Modular Polynomial Relations. In *Advances in Cryptology - Crypto '97*, LNCS vol. 1294, Springer, 1997, pages 16-30.
16. P. Gemmell. An Introduction to Threshold Cryptography. *RSA Laboratories CryptoBytes*, Vol.2, No.3, Winter 1997.
17. R. Gennaro, S. Halevi and T. Rabin. Secure Hash-and-Sign Signatures without the Random Oracle. In *Advances in Cryptology - Eurocrypt '99*, LNCS vol. 1592, Springer, 1999, pages 123-139.
18. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. *Crypto'96*, pp.157–172, Lecture Notes in Computer Science vol.1109, Springer-Verlag, 1996.
19. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. *19th ACM Symposium on Theory of Computing*, pp.218–229, ACM Press, 1987.
20. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Crypto'91*, pp.129-140, Lecture Notes in Computer Science vol.576, Springer-Verlag, 1992.
21. T. Rabin. A Simplified Approach to Threshold and Proactive RSA. *Crypto'98*, pp.89–104, Lecture Notes in Computer Science vol.1462, Springer-Verlag, 1998.
22. R. Rivest, A. Shamir and L. Adelman. A Method for Obtaining Digital Signature and Public Key Cryptosystems. *Comm. of ACM*, 21 (1978), pp. 120–126
23. A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons. 1986.

24. M. Sudan. Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems. Lecture Notes in Computer Science, vol.1001, Springer-Verlag, 1995.

A The proof of share correctness

The problem facing the players in Part 3, Step 2 of Robust Protocol can be abstracted as follows. We have public values $A = G^a H^{\hat{a}}$, $B = G^b H^{\hat{b}}$, $C = G^c H^{\hat{c}}$ and e. A player P knows $a, \hat{a}, b, \hat{b}, c, \hat{c}$, it publishes a value F, and needs to prove that $F = ab + ec$ (in Robust Protocol each player P_i has to perform this task with $a = f(i), \hat{a} = \hat{f}(i)$, $b = g(i), \hat{b} = \hat{g}(i)$, $c = g(i), \hat{c} = \hat{g}(i)$; we are not considering the randomizers $\rho(i), \hat{\rho}(i)$ for simplicity.)

Notice that the problem arises because P has to open a value that contains the product ab of two committed values. We solve the problem by having P publish a new commitment $D = G^{ab} H^{\tau}$ to ab and prove in zero-knowledge that it is correct, and then open the commitment $DC^e = G^{ab+ec} H^{\tau + e\hat{c}}$.

The protocol described in Figure 4 works for the case in which we use the robust solution based on the Strong-RSA assumption and assumes that M is the product of two safe primes. For the other version of the robust protocol (the one based on discrete-log), a similar, simpler, protocol can be used as described in the final version.

Prove-Correct

Private input for P: $a, \hat{a}, b, \hat{b}, c, \hat{c}$.
Public Input: RSA modulus M, $G, H \in Z_M^*$ as above. $A = G^a H^{\hat{a}}$, $B = G^b H^{\hat{b}}$, $G = G^c H^{\hat{c}}$, and F.
Goal: Prove that $F = ab + ec$.

1. P chooses a random $\tau \in [-M^2, M^2]$ and publishes $D = G^{ab} H^{\tau}$.
2. P proves in zero-knowledge (to a verifier V) that D is correct w.r.t. A, B as follows
   (a) P chooses $\alpha, \hat{\alpha}, \beta, \hat{\beta}, \hat{\gamma}$ at random in $[-M^6, M^6]$, and send to V the values $M_1 = G^{\alpha} H^{\hat{\alpha}}$, $M_2 = G^{\beta} H^{\hat{\beta}}$, $M_3 = B^{\alpha} H^{\hat{\gamma}}$.
   (b) V chooses a random d in $[0, M]$ and sends it to P.
   (c) P answers with the following values $x = \alpha + da$, $\hat{x} = \hat{\alpha} + d\hat{a}$, $z = \hat{\gamma} + d(\tau - \hat{b}a)$, $y = \beta + db$, $\hat{y} = \hat{\beta} + d\hat{b}$.
   (d) V accepts if $G^x H^{\hat{x}} = M_1 A^d$, $B^x H^z = M_3 D^d$, $G^y H^{\hat{y}} = M_2 B^d$
3. P reveals $f = ab + ec$ and $\hat{f} = \tau + e\hat{c}$. The value is accepted if and only if $G^f H^{\hat{f}} = DC^e \bmod M$

Fig. 4. How to prove that $F = ab + ec$

The protocol in step 2 of Prove-Correct is a honest-verifier, statistical ZK proof of knowledge of the openings of the commitments A, B, D and simultaneously proves that the opening of D is the product of the opening of A and B.

The extraction works using a technique due to Fujisaki and Okamoto [15] and it assumes that the prover is not able to solve the Strong-RSA assumption.

The proof is statistical ZK for the following reason. Notice that in our application the product ab is $O(N^4)$. By choosing the original randomizers in the set $[-N^6..N^6]$ we make sure that the Prover's answers in step 2c are statistically indistinguishable from random numbers in that interval. Details will appear in the final paper.

Practical Threshold Signatures

Victor Shoup

IBM Zurich Research Lab, Säumerstr. 4, 8803 Rüschlikon, Switzerland
sho@zurich.ibm.com

Abstract. We present an RSA threshold signature scheme. The scheme enjoys the following properties:

1. it is unforgeable and robust in the random oracle model, assuming the RSA problem is hard;
2. signature share generation and verification is completely non-interactive;
3. the size of an individual signature share is bounded by a constant times the size of the RSA modulus.

1 Introduction

A k out of l threshold signature scheme is a protocol that allows any subset of k players out of l to generate a signature, but that disallows the creation of a valid signature if fewer than k players participate in the protocol. This *non-forgeability* property should hold even if some subset of less than k players are corrupted and work together. For a threshold scheme to be useful when some players are corrupted, it should should also be *robust*, meaning that corrupted players should not be able to prevent uncorrupted players from generating signatures.

The notion of a threshold signature scheme has been extensively studied. However, all previously proposed schemes suffer from at least one of the following problems:

1. the scheme has no rigorous security proof, even in the random oracle model;
2. signature share generation and/or verification is interactive, moreover requiring a synchronous communications network;
3. the size of an individual signature share blows up linearly in the number of players.

To correct this situation, we present a new threshold RSA signature scheme that enjoys the following properties:

1. it is unforgeable and robust in the random oracle model, assuming the RSA problem is hard;
2. signature share generation and verification is completely non-interactive;
3. the size of an individual signature share is bounded by a small constant times the size of the RSA modulus.

We stress that the resulting signature is a completely standard "hash and invert" RSA signature, in the sense that the format of the public key and verification algorithm are the same as for ordinary RSA signatures. We do, however, place some restrictions on the key; namely, the public exponent must be a prime exceeding l, and the modulus must be the product of two "strong" primes.

Our scheme is exceedingly simple, and it is truly amazing that such a scheme has apparently not been previously proposed and analyzed.

We also consider a more refined notion of a threshold signature scheme, where there is one threshold t for the maximum number of corrupt players, and another threshold k for the minimum quarum size. The fact that a particular message has been signed means that at least $k - t$ uncorrupted players have authorized the signature.

Previous investigations into threshold signature schemes have always assumed (explicitly or implicitly) that $k = t + 1$. We also investigate the more general setting where $k \geq t + 1$. This generalization is useful in situations where the uncorrupted parties do not necessarily agree on what they are signing, but one wants to be able to prove that a large number of them have authorized a particular signature. In particular, threshold signatures with $k = l - t$ and $t < l/3$ can be exploited to reduce the sizes of the messages sent in Byzantine agreement protocols in an asynchronous network. This is explored in detail in [CKS00].

The application to asynchronous Byzantine agreement was actually our original motivation for studying this problem, and is the main reason for our requirement that the signing protocol is non-interactive. Almost all previous work on threshold signatures assumes a model with a synchronous network, and where all players somehow simultaneously agree to start the signing protocol on a given message. Clearly, we can not work in such a model if we want to implement asynchronous Byzantine agreement.

We stress that our notion of a "dual-parameter" threshold scheme provides stronger security guarantees than single parameter threshold schemes, and such schemes are in fact more challenging to construct and to analyze. Our notion of a dual-parameter threshold scheme should not be confused with a weaker notion that sometimes appears in the threshold cryptography literature (e.g., [MS95]). For this weaker notion, there is a parameter $k' > t$ such that the reconstruction algorithm requires k' shares, but the security guarantee is lost if just a *single* honest party reveals a share. In our notion, no security is lost unless $k - t$ honest parties reveal their shares.

We work with a "static corruption model": the adversary must choose which players to corrupt at the very beginning the attack. This is in line with previous investigations into threshold signatures, which also (explicitly or implicitly) assume static corruptions.

Our basic scheme, Protocol 1, can be proven secure when $k = t + 1$ in the random oracle model under the RSA assumption.

We present another scheme, Protocol 2, for use in the more general setting $k \geq t+1$. Protocol 2 can be be proven secure—again, in the random oracle model—when $k = t+1$ under the RSA assumption, and when $k > t+1$ under an additional assumption, namely, an appropriate variant of the Decision Diffie-Hellman assumption.

As already mentioned, our proofs of security are valid in the so-called "random oracle model," where cryptographic hash functions are replaced by a random oracle. This model was used informally by Fiat and Shamir [FS87], and later was rigorously formalized and more fully exploited in Bellare and Rogaway [BR93], and thereafter used in numerous papers.

For Protocol 1, we only need random oracles for robustness, if we assume that ordinary RSA signatures are secure. In fact, Gennaro *et al.* [GJKR96a] present a non-interactive share verification scheme that can be analyzed without resorting to random oracles. One could use their verification scheme in place of the one we suggest, thus avoiding random oracles in the analysis, but this would have certain practical drawbacks, requiring a special relationship between the sender and recipient of a share of a signature. Alternatively, one could use a simple interactive share verification scheme. The resulting signature scheme would no longer be truly non-interactive, but it would still not require any coordination or synchronization among the players. We do not explore these alternatives in any detail here, as they are quite straightforward.

The analysis of Protocol 2 makes use of the random oracle model in a more fundamental way. Since this seemed inevitable, we took several liberties in the design of Protocol 2, so that it is actually a bit simpler and more efficient than Protocol 1. Thus, even if $k = t+1$, Protocol 2 may be an attractive practical alternative to Protocol 1.

We view a proof of security in the random oracle model as a heuristic argument that provides strong evidence that a system cannot be broken. All things being equal, a proof of security in the random oracle model is not as good as a proof of security in the "real world," but is much better than no proof at all. Anyway, it does not seem unreasonable to use the random oracle model, since that is the only way we know of to justify the security of ordinary RSA signatures.

Previous Work

Desmedt [Des87] introduces the more general notion of threshold signatures. Desmedt and Frankel [DF89] present a non-robust threshold ElGamal scheme [ElG85] based on "secret sharing," [Sha79] i.e., polynomial interpolation over a finite field. Their scheme has small share size, but requires synchronized interaction. Harn [Har94] presents a robust threshold ElGamal scheme with small share size, but again requires synchronized interaction. It seems that the security of both of the above schemes can be rigorously analyzed in a satisfactory way, although neither paper does this. Gennaro *et al.* [GJKR96b] present a robust threshold DSS scheme with small share size that again requires synchronized interaction; they also give a rigorous security analysis.

All of the above-mentioned schemes are interactive. Indeed, any threshold signature scheme based on discrete logarithms appears doomed to be interactive, since all such signature schemes are randomized, and so the signers have to generate random values jointly, which apparently requires interaction.

In [DF89], Desmedt and Frankel also briefly address the problem of designing a threshold RSA [RSA78] signature scheme, noting that there are some technical obstructions to doing this arising from the fact that polynomial interpolation over the coefficient ring $\mathbf{Z}_{\phi(n)}$, where n is the RSA modulus and ϕ the Euler totient function, is somewhat awkward. Later, Desmedt and Frankel [DF91] return again to the problem of threshold RSA, and present a non-robust threshold RSA scheme that is non-interactive and with small share size, but with no security analysis. Frankel and Desmedt [FD92] present results extending those in [DF91], giving a proof of security for a non-robust threshold RSA scheme with small share size, but which requires synchronized interaction. Later, De Santis *et al.* [DDFY94] present a variation (also non-robust) on the scheme in [FD92] that trades interaction for large share size (growing linearly in the number of players). Both [FD92] and [DDFY94] avoid the problems of polynomial interpolation over $\mathbf{Z}_{\phi(n)}$ by working instead with over $\mathbf{Z}_{\phi(n)}[X]/(\Phi_q(X))$, where where $\Phi_q(X)$ is the qth cyclotomic polynomial (taken mod $\phi(n)$), and q is a prime greater than l. This is convenient, as standard secret sharing techniques can then be directly applied, but it leads to a much more complicated schemes that also require either interaction or large share sizes.

Gennaro *et al.* [GJKR96a] give a few general techniques that allow one to make RSA threshold systems robust.

Later, Frankel *et al.* [FGMY97b,FGMY97a] and Rabin [Rab98] propose and rigorously analyze robust threshold RSA schemes that have small share size, but require synchronized interaction. These papers take a different approach to the "interpolation over $\mathbf{Z}_{\phi(n)}$ problem," sidestepping it by introducing an extra layer of "secret sharing" and *much* more interaction and complexity. These schemes have other features as well, namely they provide a type of security known as "pro-active security," a topic we do not address here at all.

As we shall see, the "interpolation over $\mathbf{Z}_{\phi(n)}$ problem" is not really a problem at all—it is entirely trivial to work around the minor technical difficulties to obtain an extremely simple and provably secure threshold RSA scheme. We do not even need a random oracle if we do not require robustness and we are willing to assume that the RSA signature scheme is itself secure.

Organization

In §2 we describe our system model and security requirements for threshold signatures. In §3 we describe Protocol 1. In §4 we analyze Protocol 1 in the case $k = t + 1$. In §5 we present Protocol 2, and analyze it in the more general case $k \geq t + 1$.

2 System model and security requirements

The participants. We have a set of l players, indexed $1, \ldots, l$, a *trusted dealer*, and an *adversary*. There is also a *signature verification algorithm*, a *share verification algorithm*, and a *share combining algorithm*.

There are two parameters:

t—the number of corrupted players;
k—the number of signature shares needed to obtain a signature.

The only requirements are that $k \geq t+1$ and $l - t \geq k$.

The action. At the beginning of the game, the adversary selects a subset of t players to *corrupt*.

In the *dealing phase*, the dealer generates a public key PK along with secret key shares $SK_1, \ldots, SK_l$, and verification keys $VK, VK_1, \ldots, VK_l$. The adversary obtains the secret key shares of the corrupted players, along with the public key and verification keys.

After the dealing phase, the adversary submits signing requests to the uncorrupted players for messages of his choice. Upon such a request, a player outputs a *signature share* for the given message.

Robustness and combining shares. The signature verification algorithm takes a input a message and a signature, along with the public key, and determines if the signature is valid. The signature share verification algorithm takes as input a message, a signature share on that message from a player i, along with PK, VK, and VK_i, and determines if the signature share is valid. The share combining algorithm takes as input a message and k valid signature shares on the message, along with the public key and (perhaps) the verification keys, and outputs a valid signature on the message.

Non-forgeability. We say that the adversary *forges a signature* if at the end of the game he outputs a valid signature on a message that was not submitted as a signing request to at least $k - t$ uncorrupted players. We say that the threshold signature scheme is *non-forgeable* if it is computationally infeasible for the adversary to forge a signature.

Discussion. Notice that our model explicitly requires that the generation and verification of signature shares is completely non-interactive.

Also notice that we have two independent parameters t and k. As mentioned in the introduction, previous investigations into threshold signatures have only dealt with the case $k = t+1$. In this case, the non-forgeability requirement simply says that a signature is forged if no uncorrupted player was asked to sign it. As we shall see, achieving non-forgeability when $k > t+1$ is harder to do than when $k = t+1$. For simplicity, we shall start with the case $k = t+1$.

3 Protocol 1: a very simple RSA threshold scheme

We now describe Protocol 1, which will be analyzed in the next section when $k = t+1$.

The dealer. The dealer chooses at random two large primes of equal length (512 bit, say) p and q, where $p = 2p' + 1$, $q = 2q' + 1$, with p', q' themselves prime. The RSA modulus is $n = pq$. Let $m = p'q'$. The dealer also chooses the RSA public exponent e as a prime $e > l$.

The public key is $PK = (n, e)$.

Next, the dealer computes $d \in \mathbf{Z}$ such that $de \equiv 1 \bmod m$. The dealer sets $a_0 = d$ and chooses a_i at random from $\{0, \ldots, m-1\}$ for $1 \leq i \leq k-1$. The numbers $a_0, \ldots, a_{k-1}$ define the polynomial $f(X) = \sum_{i=0}^{k-1} a_i X^i \in \mathbf{Z}[X]$.

For $1 \leq i \leq l$, the dealer computes

$$s_i = f(i) \bmod m. \tag{1}$$

This number s_i is the secret key share SK_i of player i.

We denote by Q_n the subgroup of squares in $\mathbf{Z}_n^*$.

Next, the dealer chooses a random $v \in Q_n$, and for $1 \leq i \leq l$ computes $v_i = v^{s_i} \in Q_n$. These elements define the verification keys: $VK = v$, and $VK_i = v_i$.

Some preliminary observations. Note that $\mathbf{Z}_n^* \simeq \mathbf{Z}_m \times \mathbf{Z}_2 \times \mathbf{Z}_2$. If we let J_n denote the subgroup of elements $x \in \mathbf{Z}_n^*$ with Jacobi symbol $(x|n) = 1$, then we have $Q_n \subset J_n \subset \mathbf{Z}_n^*$; moreover, Q_n is cyclic of order m and J_n is cyclic of order $2m$. Also, $-1 \in J_n \backslash Q_n$.

Generally speaking, we shall ensure that all group computations are done in Q_n, and corresponding exponent arithmetic in $\mathbf{Z}_m$. This is convenient, since $m = p'q'$ has no small prime factors.

Since the dealer chooses $v \in Q_n$ at random, we may assume that v generates Q_n, since this happens with all but negligible probability. Because of this, the values v_i completely determine the values $s_i \bmod m$.

For any subset of k points in $\{0, \ldots, l\}$, the value of $f(X)$ modulo m at these points uniquely determines the coefficients of $f(X)$ modulo m, and hence the value of $f(X)$ modulo m at any other point modulo in $\{0, \ldots, l\}$. This follows from the fact the corresponding Vandermonde matrix is invertible modulo m, since its determinant is relatively prime to m.

From this, it follows that for any subset of $k-1$ points in $\{1, \ldots, l\}$, the distributions of the value of $f(X)$ modulo m at these points are uniform and mutually independent.

Let $\Delta = l!$. For any subset S of k points in $\{0, \ldots, l\}$, and for any $i \in \{0, \ldots, l\} \backslash S$, and $j \in S$, we can define

$$\lambda_{i,j}^S = \Delta \frac{\prod_{j' \in S \setminus \{j\}} (i - j')}{\prod_{j' \in S \setminus \{j\}} (j - j')} \in \mathbf{Z}. \tag{2}$$

These values are derived from the standard Lagrange interpolation formula. They are clearly integers, since the denominator divides $j!(l-j)!$ which in turn divides $l!$. It is also clear that these values are easy to compute. From the Lagrange interpolation formula, we have:

$$\Delta \cdot f(i) \equiv \sum_{j \in S} \lambda_{i,j}^S f(j) \bmod m. \tag{3}$$

Valid signatures. We next describe what a valid signature looks like. We need a hash function H mapping messages to elements of $\mathbf{Z}_n^*$. If $x = H(M)$, then a valid signature on M is $y \in \mathbf{Z}_n^*$ such that $y^e = x$. This is just a classical RSA signature.

Generating a signature share. We now describe how a signature share on a message M is generated. Let $x = H(M)$. The signature share of player i consists of

$$x_i = x^{2\Delta s_i} \in Q_n, \tag{4}$$

along with a "proof of correctness."

The proof of correctness is basically just a proof that the discrete logarithm of x_i^2 to the base

$$\tilde{x} = x^{4\Delta} \tag{5}$$

is the same as the discrete logarithm of v_i to the base v. For this, we can easily adapt a well-known interactive protocol, due to Chaum and Pedersen [CP92]. We "collapse" the protocol, making it non-interactive, by using a hash function to create the challenge—this is where the random oracle model will be needed. We also have to deal with the fact that we are working in a group Q_n whose order is not known. But this is trivially dealt with by just working with sufficiently large integers.

Now the details. Let $L(n)$ be the bit-length of n. Let H' be a hash function, whose output is an L_1-bit integer, where L_1 is a secondary security parameter ($L_1 = 128$, say). To construct the proof of correctness, player i chooses a random number $r \in \{0, \ldots, 2^{L(n)+2L_1} - 1\}$, computes

$$v' = v^r,\ x' = \tilde{x}^r,\ c = H'(v, \tilde{x}, v_i, x_i^2, v', x'),\ z = s_i c + r.$$

The proof of correctness is (z, c).

To verify this proof of correctness, one checks that

$$c = H'(v, \tilde{x}, v_i, x_i^2, v^z v_i^{-c}, \tilde{x}^z x_i^{-2c}).$$

The reason for working with x_i^2 and not x_i is that although x_i is supposed to be a square, this is not easily verified. This way, we are sure to be working in Q_n, where we need to be working to ensure soundness.

Combining shares. We next describe how signature shares are combined. Suppose we have valid shares from a set S of players, where $S = \{i_1, \ldots, i_k\} \subset \{1, \ldots, l\}$.

Let $x = H(M) \in \mathbf{Z}_n^*$, and assume that $x_{i_j}^2 = x^{4\Delta s_{i_j}}$. Then to combine shares, we compute

$$w = x_{i_1}^{2\lambda_{0,i_1}^S} \cdots x_{i_k}^{2\lambda_{0,i_k}^S},$$

where the λ's are the integers defined in (2). From (3), we have $w^e = x^{e'}$, where

$$e' = 4\Delta^2. \tag{6}$$

Since $\gcd(e', e) = 1$, it is easy to compute y such that $y^e = x$, using a standard algorithm: $y = w^a x^b$ where a and b are integers such that $e'a + eb = 1$, which can be obtained from the extended Euclidean algorithm on e' and e.

4 Security analysis of Protocol 1

Theorem 1. *For $k = t + 1$, in the random oracle model for H', Protocol 1 is a secure threshold signature scheme (robust and non-forgeable) assuming the the standard RSA signature scheme is secure.*

We show how to simulate the adversary's view, given access to an RSA signing oracle which we use only when the adversary asks for a signature share from an uncorrupted player.

Let $i_1, \ldots, i_{k-1}$ be the set of corrupted players. Recall $s_i \equiv f(i) \bmod m$ for all $1 \leq i \leq l$, and $d \equiv f(0) \bmod m$.

To simulate the adversary's view, we simply choose the s_{i_j} belonging to the set of corrupted players at random from the set $\{0, \ldots, \lfloor n/4 \rfloor - 1\}$. We have already argued that the the corrupted players' secret key shares are random numbers in the set $\{0, \ldots, m-1\}$. We have

$$n/4 - m = (p' + q')/2 + 1/4 = O(n^{1/2}),$$

and from this a simple calculation shows that the statistical distance between the uniform distribution on $\{0, \ldots, \lfloor n/4 \rfloor - 1\}$ and the uniform distribution on $\{0, \ldots, m-1\}$ is $O(n^{-1/2})$.

Once these s_{i_j} values are chosen, the values s_i for the uncorrupted players are also completely determined modulo m, but cannot be easily computed. However, given $x, y \in \mathbf{Z}_n^*$ with $y^e = x$, we can easily compute $x_i = x^{2\Delta s_i}$ for an uncorrupted player i as

$$x_i = y^{2(\lambda^S_{i,0} + e(\lambda^S_{i,i_1} s_{i_1} + \cdots + \lambda^S_{i,i_{k-1}} s_{i_{k-1}}))},$$

where $S = \{0, i_1, \ldots, i_{k-1}\}$. This follows from (3).

Using this technique, we can generate the values $v, v_1, \ldots, v_l$, and also generate any share x_i of a signature, given the standard RSA signature.

This argument shows why we defined the share x_i to be $x^{2\Delta s_i}$, instead of, say, x^{2s_i}. This same idea was used by Feldman [Fel87] in the context of the different but related problem of verifiable secret sharing.

With regard to the "proofs of correctness," one can invoke the random oracle model for the hash function H' to get soundness and statistical zero-knowledge. This is quite straightforward, but we sketch the details.

First, consider soundness. We want to show that the adversary cannot construct, except with negligible probability, a proof of correctness for an incorrect share. Let x and x_i be given, along with a valid proof of correctness (z, c). We have $c = H'(v, \tilde{x}, v_i, x_i^2, v', x')$, where

$$\tilde{x} = x^{4\Delta},\ v' = v^z v_i^{-c},\ x' = \tilde{x}^z x_i^{-2c}.$$

Now, $\tilde{x}, v_i, x_i^2, v', x'$ are all easily seen to lie in Q_n, and we are assuming that v generates Q_n. So we have

$$\tilde{x} = v^{\alpha},\ v_i = v^{s_i},\ x_i^2 = v^{\beta}, v' = v^{\gamma},\ x' = v^{\delta},$$

for some integers $\alpha, \beta, \gamma, \delta$. Moreover,

$$z - cs_i \equiv \gamma \bmod m \quad \text{and} \quad z\alpha - c\beta \equiv \delta \bmod m.$$

Multiplying the first equation by α and subtracting the second, we have

$$c(\beta - s_i\alpha) \equiv \alpha\gamma - \delta \bmod m. \tag{7}$$

Now, a share is correct if and only if

$$\beta \equiv s_i\alpha \bmod m. \tag{8}$$

If (8) fails to hold, then it must fail to hold mod p' or mod q', and so (7) uniquely determines c modulo one of these primes. But in the random oracle model, the distribution of c is uniform and independent of the inputs to the hash function, and so this even happens with negligible probability.

Second, consider zero-knowledge simulatability. We can construct a simulator that simulates the adversary's view without knowing the value s_i. This view includes the values of the random oracle at those points where the adversary has queried the oracle, so the simulator is in complete charge of the random oracle. Whenever the adversary makes a query to the random oracle, if the oracle has not been previously defined at the given point, the simulator defines it to be a random value, and in any case returns the value to the adversary. When an uncorrupted player is supposed to generate a proof of correctness for a given x, x_i, the simulator chooses $c \in \{0, \ldots, 2^{L_1} - 1\}$ and $z \in \{0, \ldots, 2^{L(n)+2L_1} - 1\}$ at random, and for given values x and x_i, defines the value of the random oracle at $(v, \tilde{x}, v_i, x_i^2, v^z v_i^{-c}, \tilde{x}^z x_i^{-2c})$ to be c. With all but negligible probability, the simulator has not defined the random oracle at this point before, and so it is free to do so now. The proof is just (z, c). It is straightforward to verify that the distribution produced by this simulator is statistically close to perfect.

From soundness, we get the robustness of the threshold signature scheme. From zero-knowledge, and the above arguments, we get the non-forgeability of the threshold signature scheme, assuming that the standard RSA signature scheme is secure, i.e., existentially non-forgeable against adaptive chosen message attack. This last assumption can be further justified (see [BR93]): in the random oracle model for H, this assumption follows from the RSA assumption—given random $x \in \mathbf{Z}_n^*$, it is hard to compute y such that $y^e = x$.

5 Protocol 2: a modification and security analysis when $k \geq t + 1$

We now present Protocol 2 and analyze its security when $k \geq t + 1$. In our analysis of Protocol 2, we need to make use of the random oracle model in a

fundamental way. As such, we fully exploit the random oracle model to get a scheme that is a bit simpler and more efficient that Protocol 1.

Protocol 2 is obtained by modifying Protocol 1 as follows.

Instead of computing the secret key share s_i as in (1), the dealer computes it as

$$s_i = f(i)\Delta^{-1} \bmod m.$$

We add to the verification key VK an element $u \in \mathbf{Z}_n^*$ with Jacobi symbol $(u|n) = -1$. Note that the Jacobi symbol can be efficiently computed, and such a u can be found just by random sampling.

We then modify the share generation algorithm as follows. Let $\hat{x} = H(M)$. We set

$$x = \begin{cases} \hat{x} & \text{if } (\hat{x}|n) = 1; \\ \hat{x}u^e & \text{if } (\hat{x}|n) = -1. \end{cases}$$

This forces the Jacobi symbol of x to be 1. The share generation, verification, and combination algorithms then run as before, using this new value of x, except that we make the following simplifications: we redefine x_i, $\tilde{x}$, and e' (defined in (4), (5), and (6)) as

$$x_i = x^{2s_i}, \ \tilde{x} = x^4, \ e' = 4.$$

Thus, we eliminate the somewhat "artificial" appearances of Δ in the share generation and combination algorithms.

The original share combination algorithm produces y such that $y^e = x$. If $x = \hat{x}u^e$, then we can divide y by u, obtaining an eth root of $H(M)$, so we still obtain a standard RSA signature.

That completes the description of Protocol 2.

To analyze the security of Protocol 2, we will need to work in the random oracle model for H. The intractability assumptions we will need to make are then as follows:

- The RSA assumption—it is hard to compute y such that $y^e = x$, given random $x \in \mathbf{Z}_n^*$;
- The Decision Diffie-Hellman (DDH) assumption—given random $g, h \in Q_n$, along with g^a and h^b it is hard to decide if $a \equiv b \bmod m$.

We make our DDH assumption a bit more precise. For $h \in Q_n$, $a, b \in \mathbf{Z}_m$, and $c \in \{0, 1\}$, define

$$F(h, a, b, c) = \begin{cases} h^a \text{ if } c = 0; \\ h^b \text{ if } c = 1. \end{cases}$$

The DDH assumption states that for random $g \in Q_n$, and random h, a, b, c as above, it is hard to compute—with negligible error probability—c given $g, h, g^a, F(h, a, b, c)$.

Note that this is an average-case complexity assumption. It is equivalent to a worst-case complexity assumption, by a standard "random self reduction" argument [Sta96,NR97], provided the inputs are restricted in the following way: g and h should generate Q_n, and $\gcd(a - b, m) \notin \{p', q'\}$.

Note that the DDH is a reasonable assumption here, since the group Q_n has no small prime factors [Sho97].

By a standard "hybrid" argument (see [NR97]), the above DDH assumption is equivalent to the following: the distributions

$$(g, g^{a_1}, \ldots, g^{a_s};\ h, h^{a_1}, \ldots, h^{a_s})$$

and

$$(g, g^{a_1}, \ldots, g^{a_s};\ h, h^{b_1}, \ldots, h^{b_s})$$

are computationally indistinguishable. Here s is any (small) number, g and h are random elements of Q_n, and the a_i's and b_i's are random numbers modulo m. Note that it is possible to prove the same equivalence using the random self-reducibility property of the DDH (see [Sho99] or [BBM00]).

Theorem 2. *In the random oracle model for H and H', under the RSA and DDH assumptions Protocol 2 is a secure threshold signature scheme (robust and non-forgeable) for $k \geq t+1$; moreover, when $k = t+1$, the same holds under the RSA assumption alone.*

The proof of the robustness property goes through as before. We focus here in the proof of non-forgeability.

The reason we need the DDH assumption is the following: when $k > t+1$, some honest players may have to generate shares for the "target" message, and we need the DDH to allow us to generate "dummy" shares in this case.

The random oracle model for H will allow the simulator to choose the outputs of H as it wishes, so long as these outputs have the right distribution.

We now describe a series of simulators.

The first simulator. The simulator chooses the shares for the corrupted players $s_{i_1}, \ldots, s_{i_t}$ as random numbers chosen from $\{0, \ldots, \lfloor n/4 \rfloor - 1\}$, just as it did in the previous section.

Let $g, g_{i_{t+1}}, \ldots, g_{i_{k-1}}$ be random elements in Q_n. Here, $i_{t+1}, \ldots, i_{k-1}$ are arbitrary indices of uncorrupted players. We assume that all of these group elements are generators for Q_n, which is the case with all but negligible probability. The values $g, g_{i_{t+1}}, \ldots, g_{i_{k-1}}$ implicitly define $s_{i_{t+1}}, \ldots, s_{i_{k-1}}$ modulo m by the equation $g_{i_j} = g^{s_{i_j}}$.

We next show how to sample from the distribution

$$\hat{x}, x_1, \ldots, x_l.$$

We choose $r \in \{0, \ldots, \lfloor n/4 \rfloor - 1\}$ at random, and $b_1, b_2 \in \{0, 1\}$ at random. We set $\hat{x} = (g^r)^{\Delta^2 e} u^{-b_1 e} (-1)^{b_2}$, thus defining the corresponding value x to be $(g^r)^{\Delta^2 e}(-1)^{b_2}$. For one of the uncorrupted players $i_j \in \{i_{t+1}, \ldots, i_{k-1}\}$, we have $x_{s_j} = (g_{i_j}^r)^{2\Delta^2 e}$. For other uncorrupted players i, we can compute x_i as

$$x_i = (g^r)^{2(\lambda_{i,0}^S + \Delta e(\lambda_{i,i_1}^S s_{i_1} + \cdots + \lambda_{i,i_t}^S s_{i_t}))} (g_{i_{t+1}}^r)^{2\Delta e \lambda_{i,i_{t+1}}^S} \cdot \cdots \cdot (g_{i_{k-1}}^r)^{2\Delta e \lambda_{i,i_{k-1}}^S},$$

where $S = \{0, i_1, \ldots, i_{k-1}\}$. Again, this follows from (3).

We can generate values in this way so that $\hat{x}$ is the output of the random oracle H. We can also generate the verification keys $v, v_1, \ldots, v_l$ in basically the same way.

This simulator generates $\hat{x}$ in this way for *every* random oracle query, so we will not be able to break the RSA problem with this simulator (this is only the first step).

It is easy to see that this simulation is statistically close to perfect. The one thing to notice is that $\hat{x}$ is nearly uniformly distributed in $\mathbf{Z}_n^*$. The proof of this utilizes the fact that every element in $\mathbf{Z}_n^*$ can be expressed uniquely as $g^a u^{-eb_1} (-1)^{b_2}$, for $a \in \{0, \ldots, m-1\}$, and $b_1, b_2 \in \{0, 1\}$.

The second simulator. This simulator is the same as the first, except as follows. Let $g, g_{i_{t+1}}, \ldots, g_{i_{k-1}}$ and $h, h_{i_{t+1}}, \ldots, h_{i_{k-1}}$ be random elements in Q_n. This simulator "guesses" which message will be forged by the adversary; that is, we can assume that the forged message is an input to the random oracle, and the simulator just guesses one of these queries is the "target" message.

Everything is the same as before, except that when generating $\hat{x}, x_1, \ldots, x_l$ for the target message, the simulator performs the same calculations using the values $h, h_{i_{t+1}}, \ldots, h_{i_{k-1}}$ instead of $g^r, g^r_{i_{t+1}}, \ldots, g^r_{i_{k-1}}$ in the calculation.

This simulation is no longer statistically indistinguishable from from the real game, but this is where we use the DDH assumption. On this assumption, with non-negligible probability, the adversary will still forge a message, and that message will be the selected target.

Notice that the "correctness proofs" of the shares can be still be simulated using the random oracle model for H' just as before—the fact that the statement being "proved" is false is interesting, but irrelevant.

The third simulator. This simulator is the same as the first, except as follows. Let z be a random element in $\mathbf{Z}_n^*$. For the target message hash value, the simulator sets $\hat{x} = z$. Also, whenever the adversary asks for a signature share x_i on the target message from any uncorrupted player, the adversary simply outputs a random quadratic residue. The "correctness proofs" can still be simulated, just as before. If the adversary ever asks for more than $k - t - 1$ signature shares on the target message, the simulator simply halts and reports an error.

It is easy to see that the distribution of this simulation is identical to that of the second simulation, provided the adversary does not ask for too many shares of the target message. Indeed, because of the way the second simulator constructs the signature shares x_i from the uncorrupted players on the target message, any subset of $k - t - 1$ of them is uniformly distributed in Q_n, and independent of all other variables in the adversary's view. So with non-negligible probability, the adversary will forge a signature on the target message, which means, in particular, the he does not ask for too many shares. Moreover, if he forges this signature, then he has computed an eth root of z in $\mathbf{Z}_n^*$, thus contradicting the RSA assumption.

To complete the proof of the theorem, we simply note that when $k = t + 1$, the DDH is not needed at all in the above arguments.

Acknowledgments

Thanks to Rosario Gennaro for suggesting improvements to a previous version of the paper.

References

[BBM00] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: security proofs and improvements. In *Advances in Cryptology–Eurocrypt 2000,* pages 262–278, 2000.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[CKS00] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in Constantinople: practical asynchronous Byzantine agreement using cryptography. Manuscript, 2000.

[CP92] D. Chaum and T. Pedersen. Wallet databases with observers. In *Advances in Cryptology–Crypto '92*, pages 89–105, 1992.

[DDFY94] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *26th Annual ACM Symposium on Theory of Computing*, pages 522–533, 1994.

[Des87] Y. Desmedt. Society and group oriented cryptography: a new concept. In *Advances in Cryptology–Crypto '87*, pages 120–127, 1987.

[DF89] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology–Crypto '89*, pages 307–315, 1989.

[DF91] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In *Advances in Cryptology–Crypto '91*, pages 457–569, 1991.

[ElG85] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.

[FD92] Y. Frankel and Y. Desmedt. Parallel reliable threshold multisignature. Technical Report TR-92-04-02, Univ. of Wisconsin–Milwaukee, 1992.

[Fel87] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science*, pages 427–437, 1987.

[FGMY97a] Y. Frankel, P. Gemmall, P. MacKenzie, and M. Yung. Optimal-resilience proactive public-key cryptosystems. In *38th Annual Symposium on Foundations of Computer Science*, 1997.

[FGMY97b] Y. Frankel, P. Gemmall, P. MacKenzie, and M. Yung. Proactive RSA. In *Advances in Cryptology–Crypto '97*, 1997.

[FS87] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology–Crypto '86, Springer LNCS 263*, pages 186–194, 1987.

[GJKR96a] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In *Advances in Cryptology–Crypto '96*, pages 157–172, 1996.

[GJKR96b] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS. In *Advances in Cryptology–Eurocrypt '96*, pages 354–371, 1996.

[Har94] L. Harn. Group-oriented (t, n) threshold digitial signature scheme and digital multisignature. *IEE Proc.-Comput. Digit. Tech.*, 141(5):307–313, 1994.

[MS95] S. Micali and R. Sidney. A simple method for generating and sharing pseudo-random functions, with applications to Clipper-like key escrow systems. In *Advances in Cryptology–Crypto '95*, pages 185–196, 1995.

[NR97] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, 1997.

[Rab98] T. Rabin. A simplified approach to threshold and proactive RSA. In *Advances in Cryptology–Crypto '98*, 1998.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, pages 120–126, 1978.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology–Eurocrypt '97*, 1997.

[Sho99] V. Shoup. On formal models for secure key exchange. IBM Research Report RZ 3120, April 1999.

[Sta96] M. Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology–Eurocrypt '96*, pages 190–199, 1996.

Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures*

(Extended Abstract)

Stanisław Jarecki and Anna Lysyanskaya**

MIT Laboratory for Computer Science, Cambridge, MA 02139, USA
{stasio, anna}@theory.lcs.mit.edu

Abstract. We put forward two new measures of security for threshold schemes secure in the adaptive adversary model: security under concurrent composition; and security without the assumption of reliable erasure. Using novel constructions and analytical tools, in both these settings, we exhibit efficient secure threshold protocols for a variety of cryptographic applications. In particular, based on the recent scheme by Cramer-Shoup, we construct adaptively secure threshold cryptosystems secure against adaptive chosen ciphertext attack under the DDH intractability assumption. Our techniques are also applicable to other cryptosystems and signature schemes, like RSA, DSS, and ElGamal.
Our techniques include the first efficient implementation, for a wide but special class of protocols, of secure channels in erasure-free adaptive model.
Of independent interest, we present the notion of a *committed proof*.

1 Introduction

Overview. The idea of threshold cryptography [Des87,DF89] is that a highly sensitive operation such as decryption or signing, can be performed by a group of cooperating servers in such a way that no minority of servers are able to perform the operation by themselves, nor are they be able to prevent the other servers from performing the operation when it is required. Thus, threshold protocols implement trusted entities, based on the assumption that only a fraction of a given set of dedicated servers can become corrupted. However, it is a challenging task to design protocols that are secure in the face of realistic attacks against the

* This extended abstract is a concise presentation of two independent results by Lysyanskaya [Lys00] and Jarecki and Lysyanskaya [JL00]. Lysyanskaya [Lys00] introduces the concurrent model, presents the notion of a committed proof, and constructs threshold schemes secure against the adaptive adversary in the concurrent model; Jarecki and Lysyanskaya [JL00] introduce the erasure-free model, and present threshold schemes secure against the adaptive adversary in this model, including the efficient implementation of secure channels.

** Part of this research was carried out while the author was visiting IBM Zurich Research Laboratory.

servers. In the present extended abstract we consider two such attacks for which no previous threshold schemes can be proven secure. The two attacks correspond to two quite limiting assumptions which were necessary for the previously known solutions, and which consequently hindered their applicability.

We consider a concurrent attack where the adversary tries to get an advantage by participating in several concurrent executions of the threshold protocol. Since previous schemes were not provably secure in this adversarial model, they were limited to sequential execution synchronized among all servers. We also consider an attack in which the entire history of a server's computation is recorded and becomes available to an adversary that corrupts this server. Since no schemes were provable in this model, they had to be executed on servers that could reliably erase their data. For both of these adversarial models, we devise novel techniques that allow us to implement efficient protocols that withstand them. We exemplify these techniques with threshold implementations of the Cramer-Shoup cryptosystem [CS98], which achieves the highest known level of security: security against adaptive chosen ciphertext attack. Furthermore, our techniques also yield efficient concurrent or erasure-free adaptively secure solutions to other schemes like RSA, DSS, and ElGamal.

History. For a long time, we knew only how to design threshold protocols secure in the so-called *static* adversary model where the adversary fixes the players that will be corrupted before the protocol starts. Recently, Canetti et al. [CGJ+99a] and Frankel et al. [FMY99a-b] exhibited the first threshold schemes secure and robust against the stronger and more realistic *adaptive* adversary, who chooses which players to corrupt at any time and based on any information he sees during the protocol. These results are important since it is known that the adaptive adversary is strictly stronger than the static one [CFGN96,Can98,CDD+99]. However, none of these adaptively secure protocols remained secure under concurrent composition, and they all required erasures. In addition, the cryptosystems and signature schemes implemented by these threshold schemes are not known to be provably secure under adaptive chosen ciphertext attack/adaptive chosen message attack. We remark that even though general multi-party computation results guarantee adaptive erasure-free distributed function evaluation [BGW88,CCD88,CDD+99,CFGN96], implementing threshold cryptography via these general techniques is impractical.

General model. We consider a network of n players and an adaptive adversary that can corrupt up to a minority $t < n/2$ of the players. The players have access to a reliable broadcast channel, there are insecure point-to-point links between each pair of them, and the message delivery is partially synchronous.

Concurrent model. We consider the *concurrent* setting, where many invocations of possibly the same threshold cryptosystem or signature scheme can be executed at the same time, and each of them must remain secure. This previously unexplored setting models an important property of threshold systems: the possibility of executing several protocols at the same time.

Erasure-free model. All of the threshold systems mentioned so far are implemented using erasures. That is to say, they are only secure if honest players can erase local data once it is no longer needed. However, secure erasure is hard to achieve in practice. On the hardware level, it is difficult to permanently erase information from hardware storage devices. On the system maintenance level, the need to erase data complicates standard computer system bookkeeping and backup procedures. Most serious problems arise on the operating systems level, since in order to securely erase the data, one needs to erase it from all the caches and from the part of the hard drive that was used for page swaps, etc. Di Crescenzo et al. [CFIJ99] discuss this problem and suggest a solution that enables erasures based on the assumption that some area of memory can indeed be securely erased. In contrast, we show that in the adaptively secure threshold setting it is possible to get rid of the need of secure data erasure altogether. We thus examine an *erasure-free* model, in which the adversary is effectively allowed to examine the entire history of the computation of a party it corrupts.

Techniques of independent interest. We introduce the notion of a *committed proof*, i.e. a zero-knowledge proof of an unknown statement [Lys00]. It was not known before that it was possible to prove a statement without revealing it to the verifier until the very last round of communication. Here we use such committed proofs to achieve threshold cryptosystems adaptively secure in the concurrent model. Another useful technique of independent interest that we put forward as an implementation of secure channels in the erasure-free model is our *receiver-non-committing encryption* scheme [JL00]. A non-committing encryption scheme has a property that there is a way to generate messages that look like ciphertexts but do not commit the players to any particular plaintext. We give a simple and efficient encryption scheme that is non-committing for the receiver under the decisional Diffie-Hellman intractability assumption.

Organization. In Section 2 we give an overview of our results and the most important techniques which allow us to achieve them. In Section 3 we present the notion and an implementation of a committed proof. Section 4 presents our non-committing encryption scheme. We then present our adaptive threshold protocols: Section 5 describes the basic building blocks of our solutions; sections 6 and 7 exemplify our techniques with two threshold implementations of the Cramer-Shoup cryptosystem: (1) an erasure-enabled protocol secure in concurrent composition; (2) an erasure-free protocol which is concurrently secure only under certain restrictions. For a more thorough treatment of our results pertaining to the concurrent model and to committed proofs, see the work of Lysyanskaya [Lys00]. For a more thorough treatment of our results pertaining to the erasure-free model and non-committing encryption, see Jarecki and Lysyanskaya [JL00].

2 Overview of our concurrent and erasure-free protocols

Definitions and goals. A threshold cryptosystem or signature scheme implemented by n players with threshold t is said to be *secure* if the view of the adversary that corrupts up to t players does not enable him to compute decryptions or signatures on his own. A threshold scheme is said to be *robust* if, no matter what the corrupted t players do, the remaining (i.e. honest) players still output a valid decryption or signature. (For formal definitions of security and robustness, see previous work [SG98,CG99,CGJ+99b].)

A standard technique of proving security of a threshold cryptosystem (or a signature scheme) is to exhibit a *simulation* algorithm which, without access to any secret information but with an oracle access to the single-server realization of the underlying cryptosystem, furnishes the adversary with the correct view of the execution of the threshold protocol. Thus, by exhibiting such simulator, we *reduce* the security of the threshold version of a cryptosystem to the security of its single-server counterpart.

A corresponding standard technique for proving robustness of a threshold scheme is to exhibit a knowledge *extractor* which plays the part of the honest players in the protocol, and in case the adversary succeeds in inducing the honest players into producing an invalid output, it extracts from the adversary's behavior a solution to some hard problem. Thus again, by exhibiting such extractor, we *reduce* the robustness of our threshold protocol to some standard hardness assumption.

Previous adaptively secure solutions. The task of strengthening statically-secure protocols to handle an adaptive adversary contains a following difficulty: To compute an instance of a certain function robustly (we abstract from whether the function is a signature or a decryption), say an exponentiation function $A{=}m^a$ on instance m, where a is secret-shared, the players must publish some partial results of this function, say values $A_i{=}m^{\alpha_i}$ where α_i's are the polynomial shares of a. In the static model, since the group of corrupted players is fixed, without knowing a, on the input $A{=}m^a$ received from the function oracle (see the *definitions* paragraph above), the simulator can produce the view of the protocol that outputs A by picking the shares of the corrupted players and using them to interpolate values A_i of the honest players. However, in the adaptive model, such simulation fails because the simulator cannot publish A_i's for the honest players and then be able to open the values α_i s.t. $m^{\alpha_i}{=}A_i$ for any t-sized group of players that the adaptive adversary chooses to corrupt: That would imply the knowledge of more than t shares α_i, and hence the knowledge of a.

Recent adaptively secure protocols [CGJ+99,FMY99a-b] have overcome this difficulty with the following ideas: i) Value A can be reconstructed if every player publishes $A_i = m^{a_i}$ where a_i is its *additive* share of a, i.e. $\sum a_i{=}a$; ii) Robustness, previously guaranteed via "interpolation in the exponent" of values A_i, is achieved via generation of Pedersen's commitments $g^{a_i}h^{\hat{a}_i}$ along each share a_i, and with zero-knowledge proofs that show that m^{a_i} corresponds to the committed value. Because the shares are additive, the simulator can reveal a consistent

internal state a_i for *all but one* of the players it controls. If that single *inconsistent* player is corrupted, which happens with at most 1/2 probability, the simulation is rewound to the beginning of this instance of the function application. Thus the simulation of a single instance succeeds after expected two trials. However, since such rewinding must be constrained to within the single instance of the function application (see the overview discussion of Canetti et al. [CGJ+99a]), the additive shares a_i used in this protocol must be erased (resharing must be performed to enable the application of the function on a new instance), so that the simulator will know again the proper internal state of that player: He simply no longer needs to show the information that he cannot produce.

Concurrent adaptive security with committed proofs. Our first observation about the above reasoning is that there might be *no* inconsistent player during the simulation at all, if the "compromising" share a_i can be erased *before* the partial result A_i is published. Since there would be no inconsistent players, the simulator would never have to rewind, and hence *concurrent* executions of such threshold protocol can be simulated and thus proven secure. However, how can we achieve robustness if a player is to erase its share a_i before publishing A_i? We show that it is indeed possible by devising a novel tool of a *committed* zero-knowledge proof (see Sec. 3), where a statement that needs to be proven, e.g. "A_i and $g^{a_i}h^{\hat{a}_i}$ contain the same value a_i", is revealed only after the proof ends. In particular, it can be revealed after the witness a_i needed to prove the above statement is erased. This committed proof technique can thus be applied to transform, with negligible increase in communication complexity, the adaptive DSS and RSA solutions [CGJ+99,FMY99a-b], as well as other protocols like threshold ElGamal, to concurrently secure adaptive solutions.

We further observe that by providing robustness while eliminating all inconsistent players in the above way, the committed proof technique can actually transform, in the erasure-enabled setting, a very general class of *statically* secure threshold protocols into adaptively and concurrently secure ones (see Lysyanskaya [Lys00] for more discussion). In Section 6 we exemplify the generality of these techniques with a threshold Cramer-Shoup cryptosystem.

Erasure-free adaptive security with persistently inconsistent players. Our second observation is that in the above simulation [CGJ+99,FMY99a-b] a random inconsistent player need not be picked in a simulation of *each instance* of the function application protocol. Instead, it can pick some player at the beginning of the simulation process, and use that player as a single *persistently inconsistent* player in a simulation of each instance of the function application. If that player is ever corrupted, the simulation fails, but since that happens only with at most 1/2 probability, such simulation still establishes a reduction from the security of the threshold protocol to the security of the underlying cryptosystem or signature scheme. If we can show that indeed this single player is the only player whose internal state held by the simulator is inconsistent with the adversary's view of the protocol, then our protocols do not have to resort to erasure, and hence they are secure in the erasure-free model.

We achieve this goal in two steps: First we remove the need to erase on the protocol level, by which we mean that the resulting scheme is secure in erasure-free model if it is implemented over *secure channels*. We do this, in general, by using additive sharing instead of polynomial sharing throughout the threshold protocols. Secondly, for the threshold protocols that need secure channels, we need to devise an encryption that implements the secure channels abstraction in the adaptive erasure-free model. This is an intriguing and non-trivial task, and the solution of *non-committing encryption for the receiver* which we provide in Section 4 is better than the available solutions [CFGN96,Bea97] of general non-committing encryption because it does not introduce any non-negligible communication overhead. The reason why non-committing encryption for the receiver only is sufficient is because not only is our simulator able to reveal, at the time of corruption, the history of computation of all players it controls except the persistently inconsistent one, but he already knows the values of all messages sent by these players at the time the messages are sent. These techniques yield efficient adaptive non-erasing protocols for DSS, RSA, and ElGamal (additionally, our methods lead to a dramatic reduction in the cost of adaptive RSA [JL00]). In Section 7 we exemplify them with a threshold Cramer-Shoup cryptosystem.

Finally, we remark that since the simulators in our erasure-free protocols are also non-rewinding (although they have $1/2$ probability of faiure), a concurrent execution of any number of instances of such protocols is secure if, for example, they are executed by dedicated players (see [JL00] for more details).

3 Adaptive, concurrent security via committed proofs

In this section, we present the notion of a committed proof [Lys00], which is a zero-knowledge proof that is carried out *in a committed form*. The verifier does not learn the statement that is being proven until the very last round of the protocol. As discussed in section 2, this technique gives a general tool that transforms statically secure threshold protocols to adaptively secure ones, with the additional property that their security is preserved under concurrent composition.

Suppose we are given a following three-step public-coin honest-verifier zero-knowledge proof of knowledge system Z [BG92] for language L:

1. The proof system has perfect completeness and soundness $2^{-\Omega(k)}$.
2. The prover's input is $x \in L$, a witness w, and some randomness r.
3. The random coins R are tossed after the prover issues the first message.
4. Algorithms $P_1(x, w, r)$, and $P_2(x, w, r, R)$ generate the first and second messages of the prover.
5. The verifier runs algorithm $Ver(x, m_1, R, m_2)$ to determine whether to accept or reject.
6. The simulator algorithm SIM used for proving the zero-knowledge property of Z, has the property that for all inputs $R \in \{0,1\}^k$, it generates an accepting transcript (m_1, R, m_2) indistinguishable from a transcript of a conversation with the real prover.

7. The knowledge extractor algorithm KE for Z has the property that, for some constant c, on input $(x, m_1, R, R', \ldots, R^{(c)}, m_2, m_2', \ldots, m_2^{(c)})$ such that $R \neq R' \neq \ldots \neq R^{(c)}$ and Ver accepts all transcripts (x, m_1, R, m_2), (x, m_1, R', m_2'), $\ldots$, $(x, m_1, R^{(c)}, m_2^{(c)})$, KE outputs a witness w with probability $1 - neg(k)$.

Such proof systems exist for all languages in NP, by a witness-preserving reduction to Hamiltonian cycles [Gol95]. In particular, for proving knowledge or equality of discrete logarithms or representations, such proof systems have perfect simulations and are well-studied and efficient [Bra99,Cam98].

Suppose that x for which the prover is demonstrating membership in L is unknown to the verifier. However, the verifier knows the distribution $\mathcal{D}$ from which x has been sampled. Moreover, $\mathcal{D}$ has the property that there is an efficiently samplable joint distribution $(\mathcal{W}, \mathcal{D})$ from which pairs (w, x) are sampled, such that w is a witness for the statement $x \in L$. For example, x can be a tuple (G_q, g, h, y) and statement $x \in L$ means that y is an element in G_q that can be represented in bases g and h. When we sample $\mathcal{D}$, we can first generate a random $\alpha, \beta \in \mathbb{Z}_q$, then and then set $w = (\alpha, \beta)$, and $y = g^\alpha h^\beta$.

Suppose we are given a *trapdoor* commitment scheme, i.e. a commitment scheme that has the property that for any instance of the commitment scheme, there exists a trapdoor σ the knowledge of which enables to open any commitment to an arbitrary value within some given domain.

For example, consider Pedersen commitment: an instance is a group G_q of order q in which the discrete logarithm problem is hard, with generators g and h and a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q^*$. The trapdoor $\sigma = \log_g h$. To commit to x, choose a random r and output $g^{\mathcal{H}(x)} h^r$. To open the commitment, reveal x and r. If σ is known, it is easy to see that a commitment can be opened to any x. Note that if we are not given a collision-resistant hash function, then the prover can still commit to his input x and the first message of the proof, but this commitment will have to use some special encoding of x and will be larger.

How can we create a simulator such that σ is known to it? In multi-party systems, we can have an instance of the commitment scheme generated as part of the set-up for the system; then it will follow from the properties of multi-party computation that a simulator will know σ. We will discuss such a protocol in section 5.1. In two-party protocols, σ can be a value known to the verifier, but not the prover; the simulator with black-box access to the verifier will then have to *extract* σ from the verifier.

Using trapdoor commitments, the prover can execute the proof *without revealing x to the verifier until the very end of the proof.* Consider the protocol in figure 1 between a prover and a verifier. The protocol uses Pedersen commitment, but any trapdoor commitment can be used instead.

Note (Completeness): We get completeness for free from proof system Z.

Lemma 1. (Zero-knowledge) *This protocol is zero-knowledge for any verifier.*

Proof: The lemma follows from the fact that for a simulator that knows $\log_g h$, the commitments M_1 and M_2 are not binding, and so the simulator can reveal

Common inputs: (G_q, g, h): an instance of Pedersen commitment.
Prover's inputs: statement $x \in \mathcal{D}$, witness w, random input r.
Verifier's goal: obtain x s.t. prover knows a witness to "$x \in L$."

$P \longrightarrow V$ Prover computes $m_1 = P_1(x, w, r)$, chooses random r_1 and sends $M_1 = g^{\mathcal{H}(x,m_1)} h^{r_1}$.
$P \longleftarrow V$ Verifier tosses random coins R and sends them to the prover.
$P \longrightarrow V$ Prover computes $m_2 = P_2(x, w, r, R)$, chooses random r_2 and sends $M_2 = g^{\mathcal{H}(m_2)} h^{r_2}$. Prover erases w.
$P \longrightarrow V$ Prover sends x, m_1, m_2, r_1, r_2, i.e. opens commitments M_1, M_2.
Acceptance: The verifier accepts if M_1 is a valid commitment to x and m_1, M_2 is a valid commitment to m_2, and $Ver(x, m_1, R, m_2)$ accepts.

Fig. 1. Committed proof

x, message m_1 and response m_2 in the very end, when it already knows the challenge R, by property 6 of proof system Z. □

Note: Notice that the original proof system Z was zero-knowledge for the public-coin model only, while the proof system we obtain is zero-knowledge for *any* verifier. (We achieve this because of a preprocessing step that generates h.)

Lemma 2. (Concurrent composition) *This protocol remains secure when executed concurrently (i.e. with an arbitrary interleaving of steps) with arbitrarily many invocations of itself or of any other concurrently composable protocols.*

Proof: The lemma follows from the fact that the above simulator that exhibits the zero-knowledge property does not need to rewind the verifier. □

Lemma 3. (Soundness and knowledge extraction) *If the discrete logarithm problem is hard, and the hash function $\mathcal{H} : \{0,1\}^* \to {\mathbb{Z}_q}^*$ is collision-resistant, then for this protocol there exists a polynomial-time knowledge extractor such that if the verifier accepts with non-negligible probability, then with probability $1 - neg(k)$ the knowledge extractor learns the witness w for x that the prover possesses.*

Proof: We will exhibit a knowledge extractor which, with black-box access to the prover that induces the verifier to accept with non-negligible probability, either extracts a witness for x or computes the discrete logarithm of h to the base g, or finds a collision in $\mathcal{H}$. Clearly this is sufficient to prove the lemma.

The extractor runs the prover and obtains the x, as well as m_1, R, m_2 and M_1, r_1, M_2, r_2. Now the extractor rewinds the prover to step 3 of the protocol and issues a challenge $R' \neq R$. Running the protocol to the end allows the verifier to obtain x', as well as m_1', m_2', r_1', r_2' and M_2'. Note that since the prover replies with non-negligible probability, with enough rewindings, we will get as many replies from him as the knowledge extractor KE of proof system Z may need.

Suppose $x \neq x'$. Then either $x = \mathcal{H}(x, m_1) \neq \mathcal{H}(x', m_1') = x'$ or we have found a collision in the hash function. If the latter, we have the desired contradiction. Otherwise, $g^x h^{r_1} = M_1 = g^{x'} h^{r_1'}$, and so we can compute $\log_g h$.

Now suppose $x = x'$. Then, by the same argument as above, $m_1 = m_1'$ or we find a collision or compute discrete log. Then since m_2 is a valid response to challenge R and so is m_2' to challenge R', it follows from the fact that Z is a proof of knowledge that we can extract a witness for x by using KE. □

Finally, lemma 4 below is the key to why a committed proof is instrumental for designing protocols that are secure against the adaptive adversary. It captures the counter-intuitive fact that the prover can be attacked in the middle of the proof, but the adversary still learns nothing, i.e. the zero-knowledge property of the whole game is retained! The only condition required is that the distribution $(\mathcal{W}, \mathcal{D})$ that captures the adversary's *a priori* information about the distribution that x and witness w come from, be efficiently samplable.

Lemma 4. (Security against corruption) *If the prover is corrupted by the adversary in the middle of the proof,everything that the adversary learns can be accomplished either by revealing x, or by sampling $(\mathcal{W}, \mathcal{D})$.*

Proof: We prove the claim by exhibiting a simulator $\mathcal{S}$ which generates the adversary's view of the corruption. Suppose the adversary decides to corrupt the prover just before the end of step 3. Then $\mathcal{S}$ samples $(\mathcal{W}, \mathcal{D})$ and obtains a witness w' for an x'. $\mathcal{S}$ the generates a random r and, using trapdoor $\sigma = \log_g h$ computes $m_1' = P_1(x, w, r)$ and r_1' such that $M_1 = g^{\mathcal{H}(x', m_1')} h^{r_1'}$, as well as $m_2' = P_2(x, w, r, R)$ and r_2' such that $M_2 = g^{\mathcal{H}(m_2')} h^{r_2'}$. Reveal w', x', r, r_1', r_2' to the adversary. These values are distributed correctly since w' and x' come from distribution $(\mathcal{W}, \mathcal{D})$ and r, r_1', r_2' are all random values.

Suppose the adversary decides to corrupt the prover at some step before the end of step 3. Then it is clear that $\mathcal{S}$ will just have to reveal a subset of the values above (depending on whether M_1 and M_2 have been issued yet).

Suppose the adversary corrupts the prover after the end of step 3, i.e. after w was erased. Since w is erased, the adversary learns nothing more than what the verifier can learn. Thus, $\mathcal{S}$ just runs the simulator we have constructed for proving the zero-knowledge property. □

As we will see in section 6, this property of a committed proof allows us to create a perfect and never failing simulation of the adversary's view, which implies full concurrency of the erasure-enabled threshold cryptosystems we propose.

4 Implementing secure channels without erasures

In erasure-enabled adaptive threshold cryptosystems (for example our threshold Cramer-Shoup of Sec. 6) we can assume secret communication between players because they can be implemented in that model with an inexpensive technique due to Beaver and Haber [BH92]. However, if erasures are not allowed, implementing secure channels is more complicated. The problem arises because the adversary can tap all the channels and see all the ciphertexts passed between players. When the adaptive adversary corrupts a party, he expects to see cleartexts that correspond to the ciphertexts he has seen. Thus the adaptive adversary

can potentially open any generated ciphertext. When instead of the honest players, we have a simulator attempting to simulate the adversary's view (recall that such simulator is needed to prove security), we cannot easily argue why the adversary does not learn anything from, paradoxingly, the ciphertexts that he does *not* get to open. This subtle problem, known as *selective decommitment problem* (see Dwork et al. [DNRS99]), arises, from our inability to reduce an adversary that *does* learn something from such view to semantic security of encryption. This problem can be solved with a non-committing encryption, i.e. an encryption with an additional property that the ciphertext-looking messages sent by the simulator can be opened as any cleartexts, and hence contain no information.

A general solution to this problem, due to Canetti et al. [CFGN96], requires $O(k^2)$ communication for secure transmission of a single bit, where k is the security parameter. A less expensive technique under the decisional Diffie-Hellman requires $O(k)$ overhead and is due to Beaver [Bea97].

We present a conceptually simpler but less general encryption scheme E which, under the DDH assumption, is non-committing for the receiver only [JL00]. Such encryption is custom-made for the *persistently inconsistent player* paradigm. Namely, a simulator who sends the ciphertext-looking messages on behalf of the inconsistent player is able to open them freely if the adversary attacks any receiver of these messages, i.e. anybody but the inconsistent player. Since our simulation assumes that the adversary never corrupts that player anyway (which gives us 1/2 probability of success), such encryption is good enough for simulatability of our protocols. The non-committing encryption we propose has only negligible communication overhead.

E is a non-committing encryption scheme in the following sense: On the one hand, any properly encrypted message has a unique decryption. On the other, there is a procedure which, given a sender's key and some trapdoor σ, can produce special type of *invalid* ciphertexts, which, for any $a \in \mathbb{Z}_q$, can be opened as an encryption of $m = g^a$. This is achieved because there are q possible secret keys that this procedure can reveal. Moreover, under DDH, it is impossible to distinguish the regular ciphertexts and the invalid ones produced by this special procedure. The ideas we use to implement this encryption E are similar to those of Cramer and Shoup [CS98].

Common system parameters: Group G_q, generators g and h.
Bob selects: $x, y \in \mathbb{Z}_q$, **Bob sends to Alice:** $P = g^x h^y$.
Alice: To transmit message $m \in G_q$ to Bob, Alice chooses $r \in Z_q$ and sends
$A = g^r$, $B = h^r$, $C = P^r m$ to Bob.
Bob: Computes $m = C/(A^x B^y)$.

Fig. 2. Non-committing encryption scheme E

Lemma 5. *Under DDH, E is non-committing for the receiver.*

Proof: Suppose that Alice (the sender) and Bob (the receiver) are on the same side and both know $\sigma = \log_g h$ and $z = \log_g P$. Then they can compute an *invalid* ciphertext as follows: Pick $r_1 \neq r_2 \neq r_3$ at random, and let $A^* = g^{r_1}$, $B^* = g^{r_2}$, $C^* = g^{r_3}$. (A^*, B^*, C^*) is not a valid ciphertext because $r_1 \neq r_2$. If Bob is infiltrated, then for any $m_a = g^a$, he can claim that this triple is an encryption of m_a, by showing a secret key (x^*, y^*) such that the decryption algorithm outputs m_a. He can do that by solving a system of linear equations: $x^* + y^*\sigma = z \bmod q$ and $r_3 = r_1 x^* + r_2 y^* \sigma + a \bmod q$. If $r_1 \neq r_2$ this system must have a solution. Therefore, as long as a, σ and z are known to Alice and Bob, they are not committed to the plaintext.

We must now show that whether the ciphertext sent is valid or invalid as above the view of the adversary who is observing the conversation and may infiltrate Bob remains the same. Let us call the distribution that produces the tuples (P, A^*, B^*, C^*) of the invalid form, $\mathcal{E}^*(G_q, g, h)$. By $\mathcal{E}(G_q, g, h)$, we will denote the distribution that produces the tuples (P, A, B, C) where (A, B, C) is a valid ciphertext under key P. We will now show that $\mathcal{E}$ and $\mathcal{E}^*$ are computationally indistinguishable under the DDH assumption.

Suppose a DDH instance (g, u, v, w) is given. Our goal is to decide whether it was sampled according to distribution $\mathcal{D} = \{g, g^s, g^t, g^{st}\}_{\{s,t\}}$ or according to distribution $\mathcal{D}^* = \{g, g^s, g^t, g^z\}_{\{s,t,z\}}$. Create the common information for the encryption scheme as follows: Choose values α and β such that $h = g^\alpha u^\beta$. Choose x and y and create $P = g^x h^y$. Choose some random a, b, r. Send (A, B, C) where $A = (g^a v^b)^r$, $B = A^\alpha ((u^a w^b)^r)^\beta$, and $C = A^x B^y m$. Note that if $\log_g w = \log_g u \log_g v$ (i.e. the DDH instance is from $\mathcal{D}$), then the view the adversary gets is from distribution $\mathcal{E}$; otherwise the adversary's view is from distribution $\mathcal{E}^*$. Thus, the adversary that distinguishes between $\mathcal{E}$ and $\mathcal{E}^*$ can be used to distinguish between $\mathcal{D}$ and $\mathcal{D}^*$. Therefore, under DDH, no such polynomial-time adversary exists. □

Lemma 6. *If a multi-party protocol is secure against the adaptive adversary in the secure channel erasure-free model, and the simulator algorithm SIM^* used to prove security produces a perfect simulation and is such that all but a constant number of players controlled by this simulator (i.e. the inconsistent players) follow the protocol exactly, and all messages sent by all honest players can be prepared by the simulator at send-time such that (1) the inconsistent player's messages are selected uniformly at random and (2) other players' messages are distributed correctly in full consistency with whatever the simulator will open as this player's internal state, then using encryption E results in a secure multi-party protocol in insecure channels (under the DDH assumption).*

Proof Sketch: First we notice that, since the messages of the honest and consistent players are known to SIM^*, the erasure-free simulator SIM that we need to construct just uses E to encrypt the right message from them all the time. Second, we note that since the messages of the inconsistent player can also be prepared at send-time, the simulator can prepare sender's key, receiver's key, ciphertext tuples that would decrypt to these messages.

Next, we notice that if SIM uses scheme $\mathcal{E}^*$ for the inconsistent players, then, whether the simulator knows the secret inputs and follows the protocol (call that $View_1$) or simulates it as SIM^* would (call that $View_2$), the adversary sees no difference in the view because the "ciphertexts" produced by $\mathcal{E}^*$ are independent of the messages sent on the part of the sender. Now, assume that the simulator knows the players' inputs and follows the protocol, but embeds an instance of DDH into the common system parameter h, as described in lemma 5, into the ciphertext-looking messages produced on the part of the inconsistent players. This construction creates information-theoretically independent samples of $\mathcal{E}$ or $\mathcal{E}^*$ based on the same instance of the DDH (call the view of the first distribution $View_3$, and note that the second view is $View_1$ discussed above). Therefore, the adversary that differentiates these two distributions can be used to solve the DDH. Hence $View_3$ is the view of the protocol over the insecure channels, and $View_2$ is a view of a simulation, this protocol is secure. □

We note that this implementation of secure channels can only work for a special class of multi-party protocols, namely, the ones that satisfy the conditions of lemma 6. Thus, although it does not replace Beaver's elegant scheme in general, it allows us to create efficient erasure-free adaptive protocols for many schemes that are important in practice, like RSA, DSS, ElGamal, and Cramer-Shoup.

5 Common building blocks

5.1 Set-up: Generating an Instance of a Trapdoor Commitment

Our protocols rely heavily on a discrete-log based trapdoor commitment scheme due to Pedersen: On instance (p, q, g, h), where $h \in G_q$, a commitment to $x \in \mathbb{Z}_q$ is $C = g^x h^{\hat{x}}$, where $\hat{x}$ is picked at random in $\mathbb{Z}_q$. The value h that defines the commitment instance is generated jointly once and for all at the beginning of our protocols in such a way so that (1) the simulator can learn the trapdoor $\log_g h$ of the chosen commitment; and (2) the simulator can embed another instance of the discrete log problem into the generated commitment by learning the representation of h in bases g, $\tilde{g}$ of its choice. Option i) is used for proving secrecy, when knowledge of the trapdoor enables the simulator to always open the commitments of the players it controls in the way it chooses, which leads to efficient simulation of the protocols. Option ii) is used to prove robustness: If the adversary cheats in protocols that follow, the simulator can use such adversary to break an instance of the hard problem embedded in the trapdoor. When secure channels are present, h can be obtained by using general techniques of multi-party computation [BGW88,CDD+99]. When secure channel are not there, and implementing them by erasure is not an option, we can use another protocol, where each player generates his share h_i of h, and then all players, in parallel, prove knowledge of $\log_g h_i$ to each other. This is in some respect similar to the solution of Frankel et al. [FMY99a-b]. Please see Jarecki and Lysyanskaya [JL00] for the details.

5.2 Joint Random VSS and Distributed Coinflip

In Figure 3, we include the well-known protocol Joint-RVSS [Ped91,GJKR99] for joint verifiable sharing of a random secret, which is a basic building block of our protocols. We give it here anew using notation that is useful for the presentation of the protocols that follow.

Protocol: (on inputs group G_q, generators g, h)
1. Each player P_i performs a Pedersen VSS of a random value a_i:
(a) P_i picks t-deg. polynomials $f_{a_i}(z) = \sum_{k=0}^{t} c_{ik} z^k$, $f_{\hat{a}_i}(z) = \sum_{k=0}^{t} \hat{c}_{ik} z^k$
Let $a_i = f_{a_i}(0)$ and $\hat{a}_i = f_{\hat{a}_i}(0)$ be the values shared by these polynomials
P_i broadcasts $C_{ik} = g^{c_{ik}} h^{\hat{c}_{ik}}$ for $k = 0..t$. Set $F_{a_i}(z) = \prod_{k=0}^{t} (C_{ik})^{z^k}$
P_i sends to P_j shares $\alpha_{ij} = f_{a_i}(j)$, $\hat{\alpha}_{ij} = f_{\hat{a}_i}(j)$ for each $j = 1..n$
(b) Each P_j verifies if $g^{\alpha_{ij}} h^{\hat{\alpha}_{ji}} = F_{a_i}(j)$ for $i = 1..n$
If the check fails for any i, P_j broadcasts a *complaint* against P_i
(c) If P_j complained against P_i, P_i broadcasts α_{ij}, $\hat{\alpha}_{ij}$; everyone verifies it.
If P_i fails this test or receives more than t complains, exclude P_i from $Qual$
2. P_i sets his polynomial share of the generated secret a as
$\alpha_i = \sum_{P_j \in Qual} \alpha_{ji}$, and their associated randomness as $\hat{\alpha}_i = \sum_{P_j \in Qual} \hat{\alpha}_{ji}$

We label the data structure created by this protocol as RVSS-data$_{t,g,h}[a]$:
Secret Information of each player P_i: (well-defined for $P_i \in Qual$)
– $a_i, \hat{a}_i$ his additive shares of the secret and its associated randomness
– $f_{a_i}, f_{\hat{a}_i}$ t-degree polynomials he used in sharing his additive share
– $\alpha_i, \hat{\alpha}_i$ his polynomial share of the secret and its associated randomness
– $\alpha_{ji}, \hat{\alpha}_{ji}$ his polynomial shares (and assoc. rand.) of $f_{a_j}, f_{\hat{a}_j}$ for $j = 1..n$
Public Information:
– the set $Qual \subseteq \{P_1, .., P_n\}$
– verification function $F_a : \mathbb{Z}_q \to \mathbb{Z}_p^*$ (see the implicit information below)
– verification functions $F_{a_i}(z) = g^{f_{a_i}(z)} h^{f_{\hat{a}_i}(z)}$ for $P_i \in Qual$
Secret Information Defined *Implicitly* (not stored by any player):
– secret sharing t-degree polynomials $f_a(z), f_{\hat{a}}(z)$ s.t. $\alpha_i = f_a(i)$, $\hat{\alpha}_i = f_{\hat{a}}(i)$, $f_a(z) = \sum_{P_i \in Qual} f_{a_i}(z)$, $f_{\hat{a}}(z) = \sum_{P_i \in Qual} f_{\hat{a}_i}(z)$, and $F_a(z) = g^{f_a(z)} h^{f_{\hat{a}}(z)}$
– secret shared value $a = f_a(0)$ and its associated randomness $\hat{a} = f_{\hat{a}}(0)$

Fig. 3. Joint-RVSS creates a sharing RVSS-data[a] of random secret $a \in \mathbb{Z}_q$

Notation: We say that players generate RVSS-data$_{t,g,h}[a]$ if they execute this protocol with generators g, h and polynomials of degree t. We index the data produced with labels a, α, using the associated Greek letter for polynomial shares.

One use of Joint-RVSS is in a distributed coinflip protocol (Fig.4), whose security properties are formalized in Lemma 7. This lemma is useful also for other uses of Joint-RVSS, where unlike in the coinflip protocol, the generated secret is not explicitly reconstructed.

Lemma 7. *In secure channels model, the distributed coinflip protocol of Fig. 4 (1) does not use erasures and (2) simulator* SIM *simulates it without rewinding.*

Proof: The simulator for the security proof is contained in figure 4. The simulator knows $\log_g h$, thus it need not decide on a_i's for players P_i it controls until it learns a_j for each player P_j that the adversary controls. (Note that the simulator can determine the adversary's value a_j by interpolating $f_{a_j}(i)$.) After that, the simulator assigns values a_i to the players in such a way that $\sum_{P_i \in Qual} a_i = a^*$. □

Note: If the simulator is allowed to have one player $P_i \in Qual$ whose internal state is inconsistent, then it can decide on the values a_k in advance for all $P_k \neq P_i$, and only leave a_i undefined until it is able to set $a_i = a^* - \sum_{P_k \in Qual, k \neq i} a_k$. This observation will be useful for erasure-free protocols.

Protocol: (on inputs group G_q, generators g, h)
1. Players generate RVSS-data[a] (i.e. perform Joint-RVSS, Fig.3)
2. Each $P_i \in Qual$ broadcasts his additive shares $a_i, \hat{a}_i$
3. For $P_i \in Qual$ s.t. $g^{a_i} h^{\hat{a}_i} \neq F_{a_i}(0)$, the players reconstruct P_i's additive share a_i by broadcasting their shares $\alpha_{ij}, \hat{\alpha}_{ij}$ and verifying them with F_{a_i}
4. A public random value a is reconstructed as $a = \sum_{P_i \in Qual} a_i$

Simulation: (on SIM's inputs G_q, g, h and $\sigma = \log_g h$)
1. SIM performs Joint-RVSS on the part of the honest players
2. SIM receives random $a^* \in \mathbb{Z}_q$. For some P_i among the players it controls: SIM broadcasts $a_i^* = a^* - \sum_{P_j \in Qual \setminus \{P_i\}} a_j$, $\hat{a}_i^*$ s.t. $a_i + \sigma \hat{a}_i = a_i^* + \sigma \hat{a}_i^*$
   For all other players P_j it controls, SIM broadcasts correct values $a_j, \hat{a}_j$
3. SIM performs Step 3 on the part of the honest players
4. Note that the public random value is reconstructed as a^*

Fig. 4. Erasure-Free Distributed Coinflip Protocol using Joint-RVSS

5.3 Simultaneous Zero-Knowledge Proofs of Knowledge

Our adaptive protocols, following the protocols of Canetti et al. [CGJ+99a], use *simultaneous* zero-knowledge proofs of knowledge to enable robustness efficiently. We describe this technique here in full generality.

Consider any honest-verifier public-coin zero-knowledge proof of knowledge system (ZKPK) [BG92]. Say that the prover shows knowledge of witness w of a public relation $A = (y, x)$ for some value y. Let (p, q, g) be a discrete-log instance and assume that the random coins in the proof system are picked in $\mathbb{Z}_q$. Assume that the simulator that exhibits the zero-knowledge property proceeds by first choosing any value for the random coin and then generating the rest of the proof transcript, and that it has zero probability of failure. Three-round ZKPKs of this form exist for, in particular, proving knowledge of discrete

logarithm, i.e. $A = \{g^x, x\}$ (e.g. Schnorr's protocol [Sch91]), or knowledge of representations, e.g $A = \{(g, h, g^x h^{\hat{x}}), (x, \hat{x})\}$ (see the work of Brands [Bra99] or Camenisch [Cam98] and the references therein). In a simultaneous proof using a three-round ZKPK, each player P_i proves knowledge of its witnesses w_i for some statement (y_i, w_i) in A in parallel, by executing the steps of the prover as in the original ZKPK protocol, while for the verifier's part, they all use a single common public coin generated with a distributed coinflip protocol. In our protocols, such simultaneous proof is preceded by h-generation and the coinflip is implemented with the protocol in Fig.4. This method generalizes to ZKPK protocols with any number of rounds: Every time a public coin is needed, it is picked via a distributed coinflip.

The following lemma is purely technical, but it isolates a convenient property of the simultaneous proof that allow us to concisely argue the security of the protocols that use it as a building block.

Lemma 8. *In the secure channels model, the simultaneous proof protocol has the following two properties: (1) It can be simulated without rewinding as long as the simulator has a consistent internal state for every player the adversary corrupts; (2) There is a simulator that can extract all the witnesses from the players controlled by the adversary.*

See Jarecki and Lysyanskaya [JL00] for the proof. From the lemma above and lemma 4 we immediately get:

Corollary 1. *In the erasure-enabled model, if the ZKPK proof used in the above simultaneous proof protocol is a committed proof of Fig.1, this protocol can be successfully simulated without rewinding even if the simulator does not know any witnesses to the statements it reveals for the players it controls.*

Lemma 8 also implies a corollary useful for our erasure-free protocols:

Corollary 2. *In the secure channels erasure-free model, the simultaneous proof protocol can be simulated if the simulator does not know the witnesses for a constant number of players it controls, as long as these players are not corrupted by the adversary.*

5.4 Shared Exponentiation Protocol

Another useful building block of our threshold cryptosystems is a protocol that computes m^a for any input element $m \in G_q$ if value $a \in \mathbb{Z}_q$ is secret-shared with RVSS-data$[a]$. This protocol has two variants, an "additive" and "polynomial" exponentiation (Figs. 5 and 6), which refers to the two methods of extracting value m^a from the sharing RVSS-data$[a]$ of a: Every player P_i broadcasts either value m^{a_i} for its additive share a_i, or value m^{α_i} for its polynomial share α_i.

The additive exponentiation protocol, which generalizes and removes erasure from the distributed key generation protocol of [CGJ+99a], is a basis of the key generation for our threshold Cramer-Shoup cryptosystems, and it is used in our threshold Cramer-Shoup decryption in the erasure-free setting. The polynomial

exponentiation is used in our concurrent erasure-enabled Cramer-Shoup decryption. Since the polynomial exponentiation protocol erases the polynomial shares α_i of a at the end, in that model we must always use a one-time randomization of the polynomial secret-sharing of a as inputs to this protocol. We omit the proofs of the two lemmas below and send the reader to Jarecki and Lysyanskaya [JL00] and Lysyanskaya [Lys00] for them.

Input: $m \in G_q$, secret sharing RVSS-data$[a]$, $g, h \in G_q$
1. Each P_i broadcasts $A_i = m^{a_i}$
2. With a simultaneous proof of Sec. 5.3, using ZKPK proof of equality of representation, each P_i proves knowledge of (equal) representation of m_i in bases $m, 1$ and $F_{a_i}(0)$ in bases g, h.
   If some P_i fails, a_i and $A_i = m^{a_i}$ are reconstructed publicly using F_{a_i}
3. Everyone computes $m^a = \prod_{i=1}^{n} A_i$

Fig. 5. Erasure-Free Additive Exponentiation with RVSS-data$[a]$

Lemma 9. *In the secure channels erasure-free model, as long as the adversary does not corrupt the designated persistently inconsistent player, the additive exponentiation protocol can be simulated such that (1) for all honest and consistent players, the simulator can provide correct messages they send at the time of sending and (2) for the honest inconsistent player, the simulator can provide messages such that if any t of them are revealed they look correct.*

Input: $m \in G_q$, secret sharing RVSS-data$[a]$, $g, h \in G_q$
1. With a simultaneous proof of Sec. 5.3, using *committed* ZKPK proof (Fig.1) of equality of representation, each P_i proves knowledge of (equal) representation of $A_i = m^{\alpha_i}$ in bases $m, 1$ and $F_a(i)$ in bases g, h.
   Note that at the end of the proof, value A_i is published and α_i erased.
2. Value m^a is interpolated in the exponent from A_i's that passed the proof

Fig. 6. Erasure-Enabled Polynomial Exponentiation with RVSS-data$[a]$

Lemma 10. *In the erasure-enabled model, the polynomial exponentiation protocol can be simulated without rewinding.*

6 Concurrent threshold Cramer-Shoup cryptosystem

The Cramer-Shoup cryptosystem. Recall the Cramer-Shoup [CS98] cryptosystem. The setting is as follows: a group G_q in which the decisional Diffie-Hellman problem is assumed to be hard, and a universal one-way family of

hash functions $\mathcal{H} : \{0,1\}^* \to {\mathbb{Z}_q}^*$ are given [BR97,Sho99a]. The secret key consists of five values, a, b, c, d, e, selected from ${\mathbb{Z}_q}^*$ uniformly at random. The public key consists of two random bases, $g_1, g_2 \in G_q$, such that the discrete logarithm that relates them is unknown, and the group elements $C = g_1^a g_2^b$, $D = g_1^c g_2^d$ and $W = g_1^e$. To encrypt a message m from a message space M (M is assumed to have an efficiently computable and invertible mapping into G_q, and so we write $m \in G_q$), Alice chooses $r \in {\mathbb{Z}_q}^*$ uniformly at random, computes $x = g_1^r$, $y = g_2^r$, $w = W^r m$, $\sigma = \mathcal{H}(x, y, w)$, and $v = C^r D^{r\sigma}$. The ciphertext is the 4-tuple (x, y, w, v). And now for decryption, we will use the Canetti-Goldwasser method [CG99]: Bob selects uniformly at random $s \in {\mathbb{Z}_q}^*$ and outputs $w/(x^e(v/v')^s)$, where $v' = x^{a+c\sigma} y^{b+d\sigma}$. Recall that, under the assumption that the decisional Diffie-Hellman problem is hard, the Cramer-Shoup cryptosystem, as well as the Canetti-Goldwasser variant thereof, has been shown to be secure against adaptive chosen ciphertext attack which is the strongest notion of security known for public-key cryptosystems [CS98,Sho99b,CG99].

Key generation. In figure 7, we present the key generation protocol for the concurrent Cramer-Shoup cryptosystem. We assume that the group G_q with a generator g and the universal one-way hash function $\mathcal{H}$ have been generated already. Indeed we may allow one server to set up these parameters and have the others verify that his computation was performed correctly. We also assume that $h \in G_q$ was generated using correct h-generation protocol.

Input: G_q, g, h, $\mathcal{H}$
Goal: Generate the Cramer-Shoup public key (g_1, g_2, C, D, W).

1. Run the joint coinflip protocol and generate random bases g_1, g_2, h_1, h_2.
2. Run Joint-RVSS five times in parallel and obtain $\mathsf{RVSS\text{-}data}_{t,g_1,h_1}[a, c, e]$ and $\mathsf{RVSS\text{-}data}_{t,g_2,h_2}[b, d]$.
3. P_i performs, in parallel, committed simultaneous proofs of knowledge of repr. in bases g_1, g_2 of values $C_i = g_1^{a_i} g_2^{b_i}$, $D_i = g_1^{c_i} g_2^{d_i}$ and $W_i = g_1^{e_i}$; and repr. in bases h_1, h_2 of values $\hat{C}_i = h_1^{\hat{a}_i} h_2^{\hat{b}_i}$, $\hat{D}_i = h_1^{\hat{c}_i} h_2^{\hat{d}_i}$, and $\hat{W}_i = g_1^{\hat{e}_i}$; P_i erases f_{a_i}, f_{b_i}, f_{c_i}, f_{d_i}, f_{e_i} and $f_{\hat{a}_i}$, $f_{\hat{b}_i}$, $f_{\hat{c}_i}$, $f_{\hat{d}_i}$, $f_{\hat{e}_i}$; P_i opens the committed proofs.
4. Verify (1) validity of other players' proofs; and (2) for all $P_k \in Qual$, $C_k \hat{C}_k = F_{a_k}(0) F_{b_k}(0)$, $D_k \hat{D}_k = F_{c_k}(0) F_{d_k}(0)$, and $W_k \hat{W}_k = F_{e_k}(0)$. For any player who failed the test, reconstruct all his secrets using backup information stored in $\mathsf{RVSS\text{-}data}[a, b, c, d, e]$.
5. Compute the public key: $C = \prod_{P_i \in Qual} C_i$, $D = \prod_{P_i \in Qual} D_i$ and $W = \prod_{P_i \in Qual} W_i$.

Fig. 7. Erasure-Enabled Key Generation for Cramer-Shoup

See Lysyanskaya [Lys00] for proofs of security and robustness for this protocol. We note that the simulator for the security proof is easy to construct. The key step here is that we generate two auxiliary bases, h_1 and h_2, such that if this is a simulation, the simulator will get to know $\log_{g_1} h_1$ and $\log_{g_2} h_2$. As a result of this and of the committed proof technique, at no step of this protocol will the simulator be committed to a particular player's internal state (see lemma 1 and lemma 4). The additive share of the public key published at the end is non-committing to any current internal state either because it is distributed independently from any non-erased information that the adversary will ever have a chance to see.

We also note that if a corrupted player deviates from the protocol but still succeeds in carrying out the committed proof so that the honest players accept, then, since these proofs are proofs of knowledge of representation, we can exhibit an extractor which will compute two different representations of some value in two different bases, and will therefore solve the discrete logarithm problem.

Decryption. In figure 8, we present the decryption protocol for the Cramer-Shoup cryptosystem. For full proofs of security and robustness of this protocol, see Lysyanskaya [Lys00].

Let us only show correctness of the decryption protocol in figure 8: if all the players behave as prescribed by the protocol, the output is valid decryption. To see this, let us look at the values O_i: $O_i = m_i m_i' = (v_i/v)^{s_i} g^{r_i s_i} g^{z_i} x^{e_i} g^{-r_i s_i} g^{o_i} g^{-z_i}$ $= (v_i/v)^{s_i} x^{e_i} g^{r_i s_i + z_i - r_i s_i + o_i - z_i} = (v_i/v)^{s_i} x^{e_i} g^{o_i}$ Since $o(i)$ is a degree $2t$ share of 0, the interpolation of these shares will yield $(v'/v)^{s(0)} u_1^z$, as in Canetti and Goldwasser [CG99].

The decryption protocol is secure because all the information that one sees before the committed proofs are opened does not commit the simulator to the internal state of any of the players (by lemma 4), and, since the simulator knows the values $\log_g h$, $\log_{g_1} h_1$ and $\log_{g_2} h_2$, the simulator can exhibit the internal state of any player at the adversary's request. The information revealed after the committed proof is information-theoretically independent of the internal state of a player who published this information, since by the time he publishes it, any secrets pertaining to it have been erased; and the whole process is perfect zero-knowledge by corollary 1. Therefore, owing to the committed proof technique we get a perfect simulation for the adversary's view. Robustness follows from lemma 3.

Key refresh. Notice that, using standard techniques [HJJ+97], the above implementation of the threshold Cramer-Shoup cryptosystem can be made *proactive* i.e. secure against mobile attackers who, over time, lose control over some of the servers, but attack new ones.

Taking the decryption off-line. Note that, as in the Canetti-Goldwasser implementation [CG99], we can precompute and store the randomizers. When a ciphertext needs to be decrypted, a user can talk to each server individually and have each server, using committed proofs, prove to the user that its share

of the decryption is valid. By lemma 2, these committed proofs can be executed concurrently. Such a method can tolerate up to $n/3$ corruptions.

7 Erasure-free threshold Cramer-Shoup cryptosystem

We exemplify our erasure-free threshold cryptography techniques with a threshold protocol for the Cramer-Shoup cryptosystem (Fig.10). We assume that the key generation was done similarly to Sec. 6, except that all sharings are of the type RVSS-data$_{t,g,h}[a, b, c, d, e]$. Since this protocol essentially exponentiates elements v^{-1}, x, y to values that are held with additive sharing, the security of this protocol can be shown in an argument similar to Lemma 9. For full analysis, as well as the key generation protocol, see Jarecki and Lysyanskaya [JL00].

This protocol uses an "additive multiplication" sub-protocol MULT (Fig.9), which creates ADD-data[c], an *additive sharing with polynomial backups* of value $c = ab \bmod q$ from sharings RVSS-data[a] and RVSS-data[b]. Note that if α_i's and β_i's are shares of t-degree polynomials f_a, f_b s.t. $f_a(0) = a$, $f_b(0) = b$ then $c = \sum_{i=1}^{n} v_i$ where $v_i = \lambda_i \alpha_i \beta_i \bmod q$ for some interpolation coefficients λ_i (assuming, for simplicity, that $n = 2t + 1$). Therefore, the players already hold additive shares v_i of c, but they are not independently distributed. Protocol MULT essentially re-randomizes this additive sharing, as the "2sum-to-2sum" protocol of [FMY99a-b] (except that here all players participate). In the future use of the newly created additive shares c_i of c, the polynomial sharings RVSS-data[a,b] can serve as backups: If any player P_j misuses or withholds its c_j in the future, these shares are used to reconstruct $v_j = \lambda_j a_j b_j$, and the values c_i's of all other players are adjusted so that $\sum_{i \neq j} c_i = c$.

Acknowledgements

Anna Lysyanskaya is immensely grateful to Victor Shoup who guided much of this research during her stay at IBM Zurich Research lab. She is also very grateful to Christian Cachin for helpful thoughts and discussions. Both authors acknowledge the help and encouragement of Ron Rivest, Shafi Goldwasser, Silvio Micali, Rosario Gennaro, Hugo Krawczyk, Tal Rabin, Klaus Kursawe and Leonid Reyzin. We also thank the anonymous referees for their numerous helpful comments.

In addition, Stanisław Jarecki acknowledges an NTT research grant, and Anna Lysyanskaya acknowledges the support of an NSF graduate fellowship and of the Lucent Technologies GRPW program.

References

[Bea97] Donald Beaver. Plug and play encryption. In *Advances in Cryptology—CRYPTO 97*. Springer-Verlag, 1997.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology—CRYPTO 92*. Springer-Verlag, 1992.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Widgerson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *ACM Symposium on Theory of Computing*, pages 1–10, 1988.

[BH92] D. Beaver and S. Haber. Cryptographic protocols provably secure againts dynamic adversaries. In *Advances in Cryptology—EUROCRYPT 92*, 1992.

[BR97] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: towards making uowhfs practical. In *Advances in Cryptology—CRYPTO 97*, 1997.

[Bra99] Stefan Brands. Rethinking public-key infrastructures and digital certificates–building in privacy. *Ph.D. dissertation, Technical University of Eindhoven*, 1999.

[Cam98] Jan Camenisch. Group signature schemes and payment systems based on the discrete logarithm problem. *ETH Series in Information Security and Cryptography, vol.2*, 1998.

[Can98] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Theory of Cryptography Library, http://philby.ucsd.edu/cryptolib/1998.html*, 1998.

[CCD88] David Chaum, Claude Crepeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 11–19, 1988.

[CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proof for finite field arithmetics, or: Can zero-knowledge be for free. In *Advances in Cryptology—CRYPTO 98*, pages 424–441. Springer-Verlag, 1998.

[CDD+99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology—EUROCRYPT 99*, 1999.

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 639–648, 1996.

[CFIJ99] Giovanni Di Crescenzo, Niels Ferguson, Russell Impagliazzo, and Markus Jakobsson. How to forget a secret. In *Proceedings of STACS'99*, 1999.

[CG99] Ran Canetti and Shafi Goldwasser. An efficient threshold public key cryptosystem secure against adaptice chosen ciphertext attack. In *Advances in Cryptology—EUROCRYPT 99*, pages 90–106. Springer-Verlag, 1999.

[CGJ+99a] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *Advances in Cryptology—CRYPTO 99*. Springer-Verlag, 1999.

[CGJ+99b] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. *http://theory.lcs.mit.edu/~cis/cis-publications.html*, 1999.

[CS98] Ronald Cramer and Victor Shoup. A practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology—CRYPTO 98*. Springer-Verlag, 1998.

[Des87] Yvo Desmedt. Society and group oriented cryptography. In *Advances in Cryptology—CRYPTO 87*. Springer-Verlag, 1987.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology—CRYPTO 89*, pages 307–315. Springer-Verlag, 1989.

[DNRS99] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. In *40th IEEE Symp. on Foundations of Comp. Science*, 1999.

[FMY99a] Yair Frankel, Philip MacKenzie, and Moti Yung. Adaptively-secure distributed threshold public key systems. In *Proceedings of ESA 99*, 1999.

[FMY99b] Yair Frankel, Philip MacKenzie, and Moti Yung. Adaptively-secure optimal-resilience proactive RSA. In *Advances in Cryptology—ASIACRYPT 99*. Springer-Verlag, 1999.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete–log based cryptosystems. In *Advances in Cryptology—EUROCRYPT 99*, pages 295–310, 1999.

[Gol95] Oded Goldreich. Foundations of cryptography: Fragments of a book. *http://theory.lcs.mit.edu/~oded*, 1995.

[HJJ+97] Amir Herzberg, Markus Jakobsson, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *4th ACM Conf. on Comp. and Comm. Security*, pages 100–110, 1997.

[JL00] Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography without erasures. *Theory of Cryptography Library*, 2000.

[Lys00] Anna Lysyanskaya. Threshold cryptography secure against the adaptive adversary, concurrently. *Theory of Cryptography Library*, 2000.

[Ped91] T. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology—EUROCRYPT 91*, pages 522–526, 1991.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[SG98] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology—EUROCRYPT 98*. Springer-Verlag, 1998.

[Sho99a] Victor Shoup. A composition theorem for universal one-way hash functions. *IBM Research Report RZ3147*, 1999.

[Sho99b] Victor Shoup. Why chosen ciphertext security matters. *IBM Research Report RZ3076*, 1999.

Input: Values obtained from the key generation protocol.
Goal: Decrypt ciphertext (x, y, w, v).
Notation: In this protocol, indexed Latin letters (e.g. a_i) denote *polynomial* shares of the corresponding values. (Unlike the rest of this extended abstract where they denote additive shares.)

1. Run Joint-RVSS five times and obtain $\mathsf{RVSS\text{-}data}_{t,g,h}[s, r, p]$ and $\mathsf{RVSS\text{-}data}_{2t,g,h}[o, z, u]$.
2. P_i computes the following values:
   (a) $l_i = x^{a_i + c_i\sigma} y^{b_i + d_i\sigma} g^{r_i} = v_i g^{r_i}$, where $v_i = x^{a_i + c_i\sigma} y^{b_i + d_i\sigma}$.
   (b) $l'_i = g^{r_i} h^{p_i}$.
   (c) $l''_i = g^{r_i s_i} h^{p_i s_i + u_i} = (l'_i)^{s_i} h^{u_i}$.
   (d) $m_i = (l_i/v)^{s_i} g^{z_i} = (v_i/v)^{s_i} g^{r_i s_i} g^{z_i}$.
   (e) $m'_i = x^{e_i} g^{-r_i s_i} g^{o_i} g^{-z_i}$.
3. Prove in committed simultaneous form:
   (a) Eq. of repr. of l_i, $F_a(i)$, $F_b(i)$, $F_c(i)$, $F_d(i)$, $F_r(i)$ in bases $(x,\ x^\sigma,\ y,\ y^\sigma,\ g,\ 1,\ 1,\ 1,\ 1,\ 1)$, $(g_1,\ 1,\ 1,\ 1,\ 1,\ h_1,\ 1,\ 1,\ 1,\ 1)$, $(1,\ 1,\ g_2,\ 1,\ 1,\ 1,\ 1,\ h_2,\ 1,\ 1)$, $(1,\ g_1,\ 1,\ 1,\ 1,\ 1,\ h_1,\ 1,\ 1,\ 1)$, $(1,\ 1,\ 1,\ g_2,\ 1,\ 1,\ 1,\ 1,\ h_2,\ 1)$, $(1,\ 1,\ 1,\ 1,\ g,\ 1,\ 1,\ 1,\ 1,\ h)$, correspondingly.
   (b) Eq. of repr. of l'_i, $F_r(i)$, $F_p(i)$ in bases $(g,\ 1,\ h,\ 1)$, $(g,\ h,\ 1,\ 1)$, $(1,\ 1,\ g,\ h)$.
   (c) Eq. of repr. of l''_i, $F_s(i)$, $F_u(i)$ in bases $(l'_i,\ h,\ 1,\ 1)$, $(g,\ 1,\ h,\ 1)$, $(1,\ g,\ 1,\ h)$.
   (d) Eq. of repr. of m_i,$F_s(i)$, $F_z(i)$ in bases $((l_i/v),\ g,\ 1,\ 1)$, $(g,\ 1,\ h,\ 1)$, $(1,\ g,\ 1,\ h)$.
   (e) Eq. of repr. of m'_i, $F_e(i)$, l''_i, $F_o(i)$, $F_z(i)$ in bases $(x, g^{-1}, g, g^{-1}, 1, 1, 1, 1)$, $(g_1, 1, 1, 1, h_1, 1, 1, 1)$, $(1, g, 1, 1, 1, h, 1, 1)$, $(1, 1, g, 1, 1, 1, h, 1)$, $(1, 1, 1, g, 1, 1, 1, h)$.
4. Erase the one-time secrets generated in step 1.
5. Open the committed proofs and reveal l_i, l'_i, l''_i, m_i, and m'_i.
6. Verify the committed proofs of other players.
7. Set a players output share $O_i = m_i m'_i$. Determine the output O by Lagrange interpolation in the exponent; the resulting decryption is w/O.

Fig. 8. Erasure-enabled decryption for Cramer-Shoup

Input: Sharings RVSS-data[a], RVSS-data[b], values p, q, g, h
Goal: Additive sharing ADD-data[c] of $c = ab = \sum_{i=1}^{n} \lambda_i \alpha_i \beta_i \bmod q$

1. Each player P_i computes its additive share $v_i = \lambda_i \alpha_i \beta_i$ of c, picks $\hat{v}_i \in Z_q$, broadcasts value $F_{v_i}(0) = g^{v_i} h^{\hat{v}_i}$, and proves that v_i in V_i is the product of α_i and $\lambda_i \beta_i$ committed to in $F_a(i)$ and $(F_b(i))^{\lambda_i}$. This is done using a simultaneous proof of Sec.(5.3) with a 3-move public-coin zero-knowledge proof of [CD98].
2. Players perform the "2sum-to-2sum" protocol of [FMY99a-b] for additive re-sharing of shares $v_1, .., v_n$. At the end each P_i computes its new additive share $c_i, \hat{c}_i$ of c, and there are public verification values $F_{c_i}(0) = g^{c_i} h^{\hat{c}_i}$.

Fig. 9. Multiplication MULT : (RVSS-data[a], RVSS-data[b]) $\rightarrow$ ADD-data[ab]

Input: Ciphertext x, y, w, σ, v, public key g_1, g_2, C, D, W, values p, q, g, h
Sharings RVSS-data[a,b,c,d,e,s] (i.e. RVSS-data[a],RVSS-data[b],etc.)
Goal: Decrypt cleartext $m = w(v^{-1})^s x^{s(a+c\sigma)-e} y^{s(b+d\sigma)} \bmod p$

1. Each player locally obtains its part of RVSS-data[$a+c\sigma$] and RVSS-data[$b+d\sigma$] from RVSS-data[a,b,c,d] and σ.
2. Let $r = s(a + c\sigma)$ and $z = s(b + d\sigma)$. Players perform two parallel MULT instances to get ADD-data[r] and ADD-data[z] from RVSS-data[s,$a+c\sigma$,$b+d\sigma$]
3. Each P_i broadcasts $m_i = w(v^{-1})^{s_i} x^{r_i - e_i} y^{z_i}$ and proves, using simultaneous proof of Sec.(5.3) with a 3-move public-coin zero-knowledge proof of equality of representation of m_i/w, $F_{s_i}(0)$, $F_{r_i}(0)/F_{e_i}(0)$, and $F_{z_i}(0)$ in appropriate bases made of elements $1, g, h, v^{-1}, x, y$ in G_q.
4. If any player fails, their secret inputs are reconstructed.

Fig. 10. Adaptive Erasure-Free Cramer-Shoup Protocol

Confirmer Signature Schemes Secure against Adaptive Adversaries

(Extended Abstract)

Jan Camenisch
IBM Research
Zurich Research Laboratory
CH–8803 Rüschlikon
jca@zurich.ibm.com

Markus Michels
Entrust Technologies (Switzerland)
Glatt Tower
CH–8301 Glattzentrum
Markus.Michels@entrust.com

Abstract. The main difference between confirmer signatures and ordinary digital signatures is that a confirmer signature can be verified only with the assistance of a semitrusted third party, the confirmer. Additionally, the confirmer can selectively convert single confirmer signatures into ordinary signatures.
This paper points out that previous models for confirmer signature schemes are too restricted to address the case where several signers share the same confirmer. More seriously, we show that various proposed schemes (some of which are provably secure in these restricted models) are vulnerable to an adaptive *signature-transformation* attack. We define a new stronger model that covers this kind of attack and provide a generic solution based on any secure ordinary signature scheme and public key encryption scheme. We also exhibit a concrete instance thereof.

1 Introduction

To limit the information dispersed by digital signatures, Chaum and van Antwerpen introduced the concept of *undeniable* signatures [10]. Undeniable signatures can only be verified with the help of the original signer. Of course, the signer should be able to deny invalid signatures but must not be able to deny valid signatures. Thereby the signer is able to control who gets to know the validity of a signature. To overcome this concept's shortcoming that signers might be unavailable or unwilling to cooperate and hence signatures would no longer be verifiable, Chaum suggested the notion of *confirmer signatures* [9]. Here, the ability to verify/deny signatures is transferred to a semitrusted third party, the confirmer. The confirmer is also given the power to convert a confirmer signature into an ordinary (i.e., publicly verifiable) signature. Of course, the confirmer should not be involved in the signing process. It is understood that the confirmer follows some policy for deciding to whom he confirms signatures or which signatures he can convert and under which circumstances (e.g., such a policy could be included in the signed message). For instance, a policy could state that confirmation is only allowed during a certain time period, only to a certain group of people, or simply that the confirmer must log all requests.

Chaum also presented a concrete scheme but neither a formal model nor a proof of security [9]. Later, Okamoto presented a formal model and proved

that confirmer signature schemes are equivalent to public-key encryption [23]. Okamoto further presented a practical solution. However, Okamoto's model explicitly enables not only the confirmer but also the signer to assist in verification of a confirmer signature. A drawback of this approach is that a coercer could force the signer to cooperate in confirming or denying a signature. Although a signer is in principle always able to prove that a confirmer signature he generated is valid (e.g., by proving knowledge of all inputs to the signing algorithm), the signer can always claim that he did not generate an alleged confirmer signature and thus is unable to prove anything, if confirmer signatures are "invisible", i.e., if it is undecidable for everybody apart from the confirmer whether a confirmer signature is valid or not. This coercer problem is (partially) overcome in the model of Michels and Stadler [21], which does not explicitly enable the signer to deny invalid signatures. They also showed that Okamoto's practical scheme is insecure because the confirmer can fake signatures. Moreover, they proposed new schemes and proved them secure in their model. Finally, all realizations proposed so far [9, 11, 21–23] incorporate the feature that the confirmer could convert confirmer signatures into ordinary but proprietary signatures (i.e., not standard signatures such as RSA PKCS#1 or DSS). However, this convertibility is not included in any of their models and it is hence uncertain whether the schemes remain secure if this feature is activated.

The contribution of this paper is to point out that various proposed confirmer schemes are insecure when many signers share the same confirmer. The latter seems to be natural in an e-commerce environment where playing the role of a confirmer is offered as a trusted third party service and signers decide on a per-signature basis which confirmer to use (possibly considering requirements from the signature receiver). More precisely, these schemes are vulnerable to an adaptive *signature-transformation* attack, where the attacker transforms a confirmer signature with respect to given signing keys into a confirmer signature with respect to other signing keys such that the resulting confirmer signature is valid only if the original signature is valid. With this new signature the attacker can enter the confirmation protocol thus circumvent the policy of the original signature. For instance, such attacks are possible against the schemes in [21] that were proved secure with respect to the model given there and applies also to some of the schemes presented in [9, 23]. We argue that the formal models [21, 23] proposed so far are too restrictive, e.g., as this kind of attack is not incorporated.

This paper exhibits a new model that fully incorporates adaptive adversaries. The model also explicitly includes the convertibility of confirmer signatures into ordinary signature schemes and excludes the signer's ability to deny invalid signatures. We present a generic solution based on any signature scheme that is secure against an adaptive chosen-message attack and on any encryption scheme that is secure against an adaptive chosen-ciphertext attack and prove its security in our model. This solution enjoys *perfect convertibility*, i.e., converted signatures are signatures with respect to the signature scheme we use as a building block. This property is unmet by all previously proposed schemes. We also provide a concrete instance based on any deterministic RSA signature scheme and the

Cramer-Shoup encryption scheme. An adaption to other signature schemes such as DSS is easily possible using techniques from [1]. Moreover, we outline how the scheme of Michels and Stadler can be adapted to be secure in our model and how scenarios such as fair contract signing and verifiable signature sharing can be addressed.

2 Confirmer Signature Model

This section provides a formal definition of confirmer signatures. After having defined our model, we discuss the differences to previously suggested models in detail and point out why various previously proposed schemes fail in our model.

2.1 Formal Model

Definition 1. *The players in a confirmer signature scheme are signers S, confirmers C, and verifiers V. A confirmer signature scheme consists of the following procedures:*

Key generation: *Let $CKGS(1^\ell) \to (x_S, y_S)$ and $CKGC(1^\ell) \to (x_C, y_C)$ be two probabilistic algorithms. The parameter ℓ is a security parameter, (x_S, y_S) is a secret/public key pair for the signer, and (x_C, y_C) is a secret/public key pair for the confirmer.*

Signing: *A probabilistic signature generation algorithm $CSig(m, x_S, y_S, y_C) \to \sigma$ for signing a message $m \in \{0,1\}^*$.*

Confirmation and disavowal: *A signature verification protocol $(CVerC, CVerV)$ between a confirmer and a verifier. The private input of the confirmer is x_C and their common input consists of m, σ, y_S, and y_C. The output of the verifier is either 1 (true) and 0 (false).*

Selective convertibility: *An algorithm $CConv(m, \sigma, y_S, x_C, y_C) \to s$ that allows a confirmer to turn a confirmer signature σ into an ordinary signature. If the conversion fails, the algorithm's output is $\perp$.*

Signature verification (ordinary): *An algorithm $COVer(m, s, y_S) \to \{0,1\}$ that allows everybody to verify signatures and takes as input a message m, a signature s, and the public key y_S of the signer.*

Before formally stating the security requirements, we try to describe the intuition behind them. Correctness and validity of confirmation/disavowal, and correctness of conversion are obvious. Security for the signer guarantees that confirmer signatures as well as converted signatures are unforgeable under an adaptive chosen-message attack (cf. [20]). Security for the confirmer/invisibility of signatures guarantees that the scheme is secure for the confirmer against adaptive chosen-confirmer-signature attacks (this is similar to security against chosen-ciphertext attacks for encryption schemes, in fact, CSig can be regarded as an encryption scheme for a single bits). This requirement also assures that no one apart from the confirmer can distinguish between valid and invalid confirmer

signatures. This ensures for instance that the signer is not coercible. Finally, non-transferability says that one cannot get more information out of the confirmation/disavowal protocol than whether a signature is valid or not.

By $\{A(u)\}$ we denote the set of all possible output values of a probabilistic algorithm A when input u.

Correctness of confirmation/disavowal: *If the confirmer and the verifier are honest, then for all ℓ, all $(x_S, y_S) \in \{CKGS(1^\ell)\}$, all $(x_C, y_C) \in \{CKGC(1^\ell)\}$, all $m \in \{0,1\}^*$, and all $\sigma \in \{0,1\}^*$,*

$$CVerV_{CVerC}(m, \sigma, y_S, y_C) = \begin{cases} 1 & \text{if } \sigma \in \{CSig(m, x_S, y_S, y_C)\} \\ 0 & \text{otherwise .} \end{cases}$$

Validity of confirmation/disavowal: *For all $CVerC^*$, all sufficiently large ℓ, all $(x_S, y_S) \in \{CKGS(1^\ell)\}$, all $(x_C, y_C) \in \{CKGC(1^\ell)\}$, all $m \in \{0,1\}^*$, all $\sigma \in \{0,1\}^*$, and for every polynomial $p(\cdot)$ we require that*

$$Prob\,[CVerV_{CVerC^*}(m, \sigma, y_S, y_C) = 0] < 1/p(\ell)$$

if $\sigma \in \{CSig(m, x_S, y_S, y_C)\}$ and

$$Prob\,[CVerV_{CVerC^*}(m, \sigma, y_S, y_C) = 1] < 1/p(\ell)$$

otherwise. The probability is taken over the coin tosses of $CVerV$ and $CVerC^*$.

Correctness of conversion: *For all ℓ, all $(x_S, y_S) \in \{CKGS(1^\ell)\}$, all $(x_C, y_C) \in \{CKGC(1^\ell)\}$, all $m \in \{0,1\}^*$, and for all $\sigma \in \{CSig(m, x_S, y_S, y_C)\}$, it holds that $COVer(m, CConv(m, \sigma, y_S, x_C, y_C), y_S) = 1$.*

Security for the signer: *Consider the following game against an adversary A. First the key generators for the signer and the confirmer are run on input 1^ℓ. Then A is given as input the public key of the signer and the confirmer, y_S and y_C, and the secret key x_C of the confirmer. A is further allowed oracle access to the signer (i.e., it may ask confirmer signatures of polynomially many messages $\{m_i\}$). Finally, A halts and outputs a pair of strings (m, u) where $m \neq m_i$ for all i. Then, for all such A and all sufficiently large ℓ we require that A's output satisfies $COVer(m, u, y_S) = 1$ with negligible probability only. The probability is taken over the coin tosses of the signer, A, and the key generators. (Note that the adversary can convert confirmer signatures itself as it is given x_C.)*

Security for the confirmer / Invisibility of Signatures: *Consider the following game against an adversary A. First the key generators for the signer and the confirmer are run on input 1^ℓ. The adversary is given the public keys of the signer and the confirmer, in addition to the secret key of the signer. Then the adversary can make arbitrary oracle queries to the confirmer via $CVerC$ and $CConv$. For doing this, the adversary is allowed at anytime (and repeatedly) to create additional signature-key pairs $(x_{S'}, y_{S'})$ (not necessarily by running the key generator) and to interact with the confirmer with respect to these keys. Then, the adversary has to present two messages $m_1, m_2 \in \{0,1\}^*$.*

After that we flip a fair coin. If the result is heads, the adversary is given $\sigma = CSig(m_1, x_S, y_S, y_C)$*, if it is tails, the adversary is given a string* $\sigma = CSig(m_2, x_S, y_S, y_C)$*. Now the adversary is again allowed to query the signer and the confirmer except that* σ *is not allowed in any of these queries. Finally, the adversary must output* 0 *or* 1*. We require that for all such adversaries, all polynomials* $p(\cdot)$*, and all sufficiently large* ℓ*, the probability that the adversary's output equals our coin flip is smaller than* $1/2 + 1/p(\ell)$*, where the probability is taken over the coin tosses of the signer, the confirmer, and the key generators.*

Non-transferability of verification/disavowal: *Consider the following two games involving the adversary, a signer, a confirmer, and a simulator:*

Game 1. The adversary is given the public keys y_S *and* y_C *of the signer and the confirmer. Then it can make arbitrary oracle queries to both of them via CSig, CVerC, and CConv. (Again the adversary is allowed at any time to create its own key pairs* $(x_{S'}, y_{S'})$ *and run, e.g., CSig with these keys, and then interact with the confirmer with respect to these keys as well.) Then the adversary must present two strings,* m *and* σ *for which it wishes to carry out the protocol* $(CVerC, CVerV)$ *with the confirmer. Next the confirmer and the adversary carry out this protocol with common input* (m, σ, y_S, y_C)*. The confirmer's secret input will be* x_C*. In parallel, the adversary is allowed to make arbitrary queries to the signer and confirmer. Eventually, the adversary stops producing an output.*

Game 2. This game is the same as Game 1 with the difference that when it comes to the interaction with the confirmer on m *and* σ *the simulator is plugged in the place of the confirmer. However, in all other interactions with the adversary the real confirmer or the real signer speak with the adversary. The simulator is not given the secret key of the confirmer, but it is allowed a single call to an oracle that tells it whether the strings* m *and* σ *produced by the adversary are a valid confirmer signature w.r.t.* y_S *and* y_C*.*

Now we require that for every adversary there exists a simulator such that for all sufficiently large ℓ*, all* $(x_S, y_S) \in \{CKGS(1^\ell)\}$*, and all* $(x_C, y_C) \in \{CKGC(1^\ell)\}$*, the outputs of the adversary when playing Game 1 and Game 2 are indistinguishable. In other words, there must exist a simulator such that the adversary cannot distinguish whether he is playing Game 1 or 2.*

We call a confirmer signature scheme *perfect convertible* with respect to some (ordinary) signature scheme if converted confirmer signatures are valid signatures with respect to this signature scheme.

Throughout the paper we assume that the policy stating the circumstances under which the confirmer is allowed to confirm/disavow a confirmer signature is part of the actual message and that he refuses cooperation whenever the policy requires so. This is sufficient to ensure that verifiers cannot circumvent a policy.

Schemes according to our definition are *separable*, i.e., all parties can run their key generation algorithms independent of each other (cf. [6]). This enables signers to choose a confirmer on a per signature basis at signing time.

Remark 1. One could easily add a protocol between a confirmer signature recipient and the signer in which the signer proves to the recipient that a confirmer signature *just generated* is valid. The only modification to our model would be that one would have to add a security requirement for this protocol that is similar to the one of *non-transferability of verification/disavowal* for $(CVerC, CVerV)$. Furthermore, the adversary has to be allowed to use this new protocol in the games defined in *security for the signer* and *non-transferability of verification/disavowal*.

2.2 Comparison with Previous Formal Models

Let us point out the differences between our model and the previous formal models [21, 23].

As mentioned in the introduction, Okamoto's model enables the signer to confirm and deny signatures, which makes the signer vulnerable to a coercer that forces him to confirm or deny a signature. The model does not include selective conversion. Moreover, his model defines a weaker notion of security of the confirmer: the adversary knowing the signer's secret key wins the game only if he is able to behave like the confirmer, i.e., to confirm and disavowal signatures, but does not win the game if he can distinguish between two confirmer signatures (or between a valid and an invalid confirmer signature). The crucial difference, however, lies in the definition of invisibility and untransferability, where the adversary has access only to the confirmation and disavowal protocol run with the *true signer*, but *not* with the confirmer. Thus it does not cover adaptive attacks against the confirmer. For instance, the signature transformation attack mentioned below is not captured by this model. In fact, one can construct a scheme that is secure in Okamoto's model but is vulnerable to this signature transformation attack. Such a scheme is obtained from one of the schemes in [21] by having the signer choose an encryption public key and then appending to the signature an encryption of all random choices made by the signer in the signing protocol under this public key (this encryption also must be signed together with the message). This will allow the signer to confirm/disavow signatures as required in Okamoto's model.

The model by Michels and Stadler does not explicitly enable the signer to confirm and deny signatures, but it does not exclude it either. In particular, the security for the confirmer (where the adversary gets the signer's secret key) as well as the selective conversion are not included. Their definition of invisibility allows the adversary only to query the confirmer with respect to a certain signer and is not given the signer's secret key, i.e., they allow only a very restricted kind of adaptive attack. This model is realistic only if there is a single signer that is furthermore assumed to be honest. However, if several signer are allowed and they are not all assumed to be honest, then their schemes are vulnerable to the signature transformation attack as described in the next paragraph.

2.3 Adaptive Signature-Transformation Attacks

This paragraph points out that the previously suggested schemes [9, 21, 23] are vulnerable to an adaptive attack and are indeed insecure in our model. Before describing this attack, we note that the scheme proposed in [22] is not secure in our model because is has the property that given a signature and two different messages it's publically verifiable w.r.t. which message the signature is potentially valid. Due to this property the invisibility requirement in our model cannot be satisfied. Furthermore, the scheme presented in [11] is insecure in *all* models, i.e., even against non-adaptive attackers (see Appendix A).

We first show that the proof-based scheme by Michels and Stadler [21, Section 5.2], which was proved secure in their model, is vulnerable to a so-called adaptive signature-transformation attack that exploits the malleability of the used building block. The practical scheme by Okamoto [23], with or without the heuristic fix of another vulnerability suggested in [21], as well as Chaum's scheme [9] are vulnerable to a similar attack. We omit the details regarding those schemes here.

Let us first recall the scheme by Michels and Stadler. It uses as building blocks so-called *proof-based signature schemes* (an example is Schnorr's signature scheme [26]) and *confirmer commitments*. For simplicity, let us use the confirmer commitment provided in [21, Section 4.2] and the Schnorr signature scheme [26] in the following. With these choices the public key of the signer is a group $G = \langle g \rangle$, a prime $q = |G|$, and $y \in G$. The signer's secret key is $x = \log_g y$. The confirmer's public key is $H = \langle h \rangle$, a prime $p = |H|$, and $z \in H$. The confirmer's secret key is $u = \log_h z$. Furthermore, a suitable hash function $\mathcal{H}$ is publicly known. The signer can issue a confirmer signature on m as follows.

1. $r_1 \in \mathbb{Z}_q$, $r_2 \in \mathbb{Z}_p$, $t := g^{r_1}$, $d := (d_1, d_2) := (z^{r_2}, h^{r_2 + \mathcal{H}(t,m)})$,
2. $c := \mathcal{H}(d)$, and $s := r_1 - cx \bmod q$.

The confirmer signature is $(t, (d_1, d_2), s)$. The confirmer can tell whether a given confirmer signature $(t, (d_1, d_2), s)$ is valid by checking if $d_2 / d_1^{1/u} \stackrel{?}{=} h^{\mathcal{H}(t,m)}$ and $y^{\mathcal{H}(d)} g^s \stackrel{?}{=} t$ hold. We refer to [21] for the confirmation/disavowal protocol.

Now we are ready to describe the signature transformation attack. We are given an alleged confirmer signature $(t, (d_1, d_2), s)$ on m w.r.t. a signer's public key (G, g, q, y). Furthermore, assume that the confirmer is not allowed to tell us whether this particular signature is a valid. The following attack will allow us to transform the signature into another signature that is independent from $(t, (d_1, d_2), s)$. To do so, we choose our own signing public and secret keys $\tilde{G} = \langle \tilde{g} \rangle$ with $|\tilde{G}| = \tilde{q}$, $\tilde{y} = \tilde{g}^{\tilde{x}}$. Then we choose a random message $\tilde{m}$ and

1. $\tilde{r}_1 \in \mathbb{Z}_{\tilde{q}}$, $\tilde{r}_2 \in \mathbb{Z}_p$, $\tilde{t} := \tilde{g}^{\tilde{r}_1}$, $\tilde{d} := (\tilde{d}_1, \tilde{d}_2) := (d_1 z^{\tilde{r}_2}, d_2 h^{\tilde{r}_2 + \mathcal{H}(\tilde{t},\tilde{m}) - \mathcal{H}(t,m)})$,
2. $\tilde{c} := \mathcal{H}(\tilde{d})$, and $\tilde{s} := \tilde{r}_1 - \tilde{c}\tilde{x} \bmod \tilde{q}$

and get the new confirmer signature $(\tilde{t}, (\tilde{d}_1, \tilde{d}_2), \tilde{s})$. This confirmer signature is valid if and only if the original confirmer signature $(t, (d_1, d_2), s)$ is valid. Furthermore, if the original confirmer signature is valid, then the new confirmer

signature is indistinguishable from a confirmer signature made using the real signing algorithm with our public key. Hence we can simply feed the signature $(\tilde{t}, (\tilde{d}_1, \tilde{d}_2), \tilde{s})$ to the confirmer and he will tell in fact whether $(t, (d_1, d_2), s)$ is valid. This attack breaks the invisibility property, and it is possible because the confirmer commitment is malleable. Note that the definition of security for confirmer commitments in [21] does not consider adaptive adversaries.

A variant of this attack works even if the used confirmer commitment is non-malleable: After the attacker has obtained the confirmer signature (t, d, s) on m w.r.t. a signer's public key (G, g, q, y), he computes a new public key $(G, \tilde{g}, q, \tilde{y})$ by picking $\tilde{r}_1, \tilde{x} \in \mathbb{Z}_q$ and computing $\tilde{g} := t^{1/\tilde{r}_1}$ and $\tilde{y} := \tilde{g}^{\tilde{x}}$. Now $(t, d, (\tilde{r}_1 - \mathcal{H}(d)\tilde{x} \bmod q))$ will be a valid confirmer signature on m w.r.t. the signer's public key $(G, \tilde{g}, q, \tilde{y})$ if and only if (t, d, s) is a valid w.r.t. (G, g, q, y). In a similar way as above this attack breaks the invisibility property. The second scheme proposed in [21] is also vulnerable to this attack. However, this kind of attack can be easily countermeasured by adding the signer's public key to the input of the confirmer commitment.

3 A Generic Realization of Confirmer Signature Schemes

This section presents a generic confirmer signature scheme, proves its security, and discusses its implications. As we will see in the next section, this generic scheme has concrete instances that are quite efficient.

Let $SIG = (SKG, Sig, Ver)$ denote a signature scheme, where SKG is the key-generation algorithm (which on input 1^ℓ outputs a key pair (x, y)), Sig is the signing algorithm (which on input of a secret key x, the corresponding public key y, and a message $m \in \{0,1\}^*$ outputs a signature s on m), and Ver is the verification algorithm (which on input of a message m, an alleged signature s, and public key y outputs 1 if and only if s is a signature on m with respect to y). Moreover, let $ENC = (EKG, Enc, Dec)$ denote a public key encryption scheme. On input of a security parameter, EKG outputs a key pair (x', y'). On input of a public key y' and a message m', Enc outputs a ciphertext c. On input of the ciphertext c of the message m', the secret key x', and the public key y', Dec outputs m' if c is valid and $\perp$ otherwise.

Given a suitable signature scheme $SIG = (SKG, Sig, Ver)$ and a suitable encryption scheme $ENC = (EKG, Enc, Dec)$, a confirmer signature scheme can be constructed as follows. We will later see what suitable means.

1. The respective key generators are chosen as $CKGS(1^\ell) \mathrel{\widehat{=}} SKG(1^\ell)$ and $CKGC(1^\ell) \mathrel{\widehat{=}} EKG(1^\ell)$.
2. The signer signs a message $m \in \{0,1\}^*$ by computing $s := Sig(x_S, y_S, m)$ and $e := Enc(y_C, s)$. The confirmer signature on m is given by e.
3. The confirmation and disavowal protocol $(CVerC, CVerV)$ between the confirmer and a verifier is done as follows: Given an alleged confirmer signature e and a message m, the confirmer decrypts e to get $\hat{s} := Dec(e, x_C, y_C)$. If $Ver(m, \hat{s}, y_S) = 1$, then the confirmer tells the verifier that the confirmer

signature is valid and shows this by proving in *concurrent* zero-knowledge that he knows values α and β such that "β is the secret key corresponding to y_C AND $\alpha = Dec(e, \beta, y_C)$ AND $Ver(m, \alpha, y_S) = 1$." Otherwise, the confirmer tells the verifier that the confirmer signature is invalid and proves in *concurrent* zero-knowledge that he knows values α and β such that "β is the secret key corresponding to y_C AND (($\alpha = Dec(e, \beta, y_C)$ AND $Ver(m, \alpha, y_S) = 0$) OR decryption fails)."

4. The selective conversion algorithm $CConv(m, e, y_S, x_C, y_C)$ outputs $Dec(e, x_C, y_C)$, provided $Ver(m, Dec(e, x_C, y_C), y_S) = 1$, and $\perp$ otherwise.
5. The public verification algorithm for converted signatures is defined as $COVer(m, s, y_S) \mathrel{\widehat{=}} Ver(m, s, y_S)$.

Theorem 1. *If SIG is existentially unforgeable under an adaptive chosen-message attack and ENC is secure against adaptive chosen-ciphertext attacks, then the above construction constitutes a secure confirmer signature scheme with perfect conversion.*

Proof: [Sketch] The properties *correctness of confirmation/disavowal*, *validity of confirmation/disavowal*, and *correctness of conversion* are obviously satisfied. Let us consider the remaining properties.

Security for the signer: We show that if there is an adversary A that can forge a confirmer signature, then A could be used to forge signatures of the signature scheme *SIG* in an adaptive chosen-message attack: The messages m_i that are queried by A are simply forwarded to the signing oracle of the underlying signature scheme *SIG* and then the result is encrypted using *ENC*. If A is able to produce a valid confirmer signature to any message that is not in the set of queried messages, we can convert this confirmer signature into a valid ordinary signature by the conversion algorithm. If A is able to compute a valid signature to any message that is not in the set of messages previously queried, we are already done. Both cases contradict the security of *SIG*.

Security for the confirmer/Invisibility of signatures: We show that if there exists an adversary A that can violate this property, then the encryption scheme *ENC* is not secure against adaptive chosen-ciphertext attacks: When getting A's request for confirmation/disavowal of a message m and an alleged confirmer signature e, we forward e to the decryption oracle of the underlying encryption scheme and obtain s. If s is an (ordinary) signature of m, then we tell A that the confirmer signature is valid and carry out the proof that this is indeed the case. Of course, we cannot carry out the real protocol, but as it is required to be concurrent zero-knowledge, there exists a simulator for it which we can use. The case where s is not a valid signature is similar. If A requests the conversion of m and e, we forward e to the decryption oracle, get s, and then we output s if s is a valid ordinary signature on m, or $\perp$ otherwise.

When it comes to the point where A presents the "test messages" m_1 and m_2, we produce signatures of them, i.e., s_1 and s_2, and present these as "test messages" to the game in the underlying encryption scheme. Then we forward the encryption we get as challenges from the underlying game to A as a

challenge. The following queries of A are handled as before. When A eventually halts and outputs 0 or 1, we forward this output as an answer in the game against the underlying encryption scheme. This concludes the reduction.

Non-transferability of verification/disavowal: This property follows in a straightforward manner from the concurrent zero-knowledge property of the proofs in the confirmation/disavowal protocol.

□

Corollary 1. *I. If trapdoor one-way permutations exist then there exists a secure confirmer signature scheme. II. A secure confirmer signature scheme exists if and only if a public key encryption scheme secure against adaptive chosen-ciphertext attacks exists (cf. [23, Theorem 3]).*

Proof: Part I. The existence of trapdoor one-way permutations implies a secure signature scheme and an encryption scheme secure against adaptive chosen-ciphertext attacks [3, 16, 20, 25]. Due to Theorem 1, this is sufficient for a secure confirmer signature scheme. Part II. On the one hand, an encryption scheme for encrypting a single bit follows from a secure confirmer signature scheme (cf. [23]). Let the public key of the encryption scheme be the public key of the confirmer. To encrypt, one chooses a signer's key pair and then a 0 is encrypted by issuing a valid confirmer signature on a randomly chosen message and a 1 is encrypted by issuing a simulated (invalid) confirmer signature on a randomly chosen message. On the other hand, if a secure public key encryption scheme exists then there exist one-way functions and hence a signature scheme secure against adaptive chosen-message attacks [20, 25]. Due to Theorem 1, this is sufficient for a secure confirmer signature scheme. □

Remark 2. The generic confirmer signature scheme exhibited in this section provides perfect convertibility with respect to the signature scheme *SIG*.

Remark 3. The described generic confirmer signature scheme has some similarities to the generic scheme due to Okamoto[23]. However, as Okamoto's model requires the signer to have the ability to deny invalid confirmer signature scheme this scheme cannot satisfy the invisibility property as stated above. Whereas Okamoto's generic scheme is a theoretical construction requiring general zero-knowledge proofs for confirmation and disavowal, our scheme has concrete instances with quite efficient protocols for confirmation and disavowal.

4 An Instance Providing Perfect Conversion of Signatures

This section provides an instance based on an arbitrary deterministic RSA signature scheme [24] and the Cramer–Shoup encryption scheme [13]. Instances for other signature schemes such as DSS or Schnorr can be realized similarly using the signature reduction techniques from [1].

4.1 Notation

We use notation from [6, 7] for the various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$PK\{(\alpha,\beta,\gamma) : y = g^{\alpha}h^{\beta} \ \wedge \ \tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\gamma} \ \wedge \ (u \leq \alpha \leq v)\} \ ,$$

denotes a "*zero-knowledge* P*roof of* K*nowledge of integers* α*,* β*, and* γ *such that* $y = g^{\alpha}h^{\beta}$ *and* $\tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\gamma}$ *holds, where* $v < \alpha < u$," where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The convention is that Greek letters denote the knowledge proved, whereas all other parameters are known to the verifier. The scheme presented in this section uses proofs of knowledge of double discrete logarithms and of roots of discrete logarithms [7, 27] and proofs that a discrete logarithm lies in an interval [5, 12], e.g., $(u \leq \log_g y \leq v)$. These protocols are 3-move zero-knowledge proofs of knowledge with binary challenges.

An important variant of such protocols are *concurrent* zero-knowledge proofs (of knowledge). They are characterized by remaining zero-knowledge even if several instances of the same protocol are run arbitrarily interleaved [14, 15, 17]. Damgård [15] shows that 3-move proofs (this includes all protocols considered in this paper) can easily be made concurrent zero-knowledge in many practical scenarios. We denote the resulting protocols by, e.g., *CZK-PK*$\{\alpha : y = g^{\alpha}\}$

4.2 Description of the Scheme

We review both schemes we use as building block briefly and then describe the resulting confirmer signature scheme.

Let (n, e) be an RSA public key of a signer and $Pad_S(\cdot) : \{0,1\}^* \to \{1, \ldots, n\}$ be some padding function. To sign a message $m \in \{0,1\}^*$, the signer computes $s := Pad_S(m)^{1/e} \bmod n$. To verify a signature s on m, one checks whether $s^e \equiv Pad_S(m) \pmod{n}$. If the padding function is assumed to be a truly random function, the system is secure against adaptively chosen-message attacks under the RSA assumption [4].

The Cramer–Shoup encryption scheme works over some group H of (large) prime order q of which two generators h_1 and h_2 are known. The secret key consists of five elements $x_1, \ldots, x_5 \in_R \mathbb{Z}_q$ and the public key (y_1, y_2, y_3) is computed as $y_1 := h_1^{x_1}h_2^{x_2}$, $y_2 := h_1^{x_3}h_2^{x_4}$, and $y_3 := h_1^{x_5}$. Encryption of a message $m \in H$ is done by choosing a random $r \in_R \mathbb{Z}_q$ and computing $c_1 := h_1^r$, $c_2 := h_2^r$, $c_3 := y_3^r m$, and $c_4 := y_1^r y_2^{r\mathcal{H}(c_1,c_2,c_3)}$. Decryption of a tuple $(c_1, c_2, c_3, c_4) \in H^4$ is done by computing $u := \mathcal{H}(c_1, c_2, c_3)$ and checking whether $c_1^{x_1+x_3u}c_2^{x_2+x_4u} = c_4$. If this condition does not hold, the decryption algorithm outputs $\perp$. Otherwise, it computes $m' := c_3/c_1^{x_5}$ and outputs m'. Provided the decision Diffie–Hellman assumption holds in H and the hash function $\mathcal{H}$ used is chosen collision resistant, the system is secure against adaptive chosen-ciphertext attacks [13].

We are now ready to present the different procedures of the confirmer signature scheme.

Key Generation: *CKGS*: The signer chooses an RSA public key (n, e). Furthermore, the signer also publishes a group $G = \langle g_1 \rangle = \langle g_2 \rangle$ of order n. *CKGC*: The confirmer chooses sufficiently large primes q and $p = 2q + 1$ and two elements h_1 and h_2 from $\mathbb{Z}_p^\star$ such that $\left(\frac{h_1}{p}\right) = \left(\frac{h_2}{p}\right) = 1$ and $\log_{h_1} h_2$ is unknown. Furthermore, the confirmer publishes a group $\tilde{H} = \langle \tilde{h}_1 \rangle$ of order p. This group is required for the proofs in the confirmation/disavowal protocol. A collision resistant hash function $\mathcal{H}$ is fixed.

Signing: We assume $n < p/2$. (The case $p/2 < n$ can be handled by splitting the signature into two or more parts before encryption. We refer to the forthcoming full version of the paper for details.) To sign a message $m \in \{0,1\}^*$, the signer computes $\hat{s} := Pad_S(m)^{1/e} \bmod n$, sets $s := \hat{s}$ if $\left(\frac{\hat{s}}{p}\right) = 1$ and $s := p - \hat{s}$ otherwise (hence $\left(\frac{s}{p}\right) = 1$). The signer encrypts s by choosing a random $r \in_R \mathbb{Z}_q$ and computing $c_1 := h_1^r$, $c_2 := h_2^r$, $c_3 := y_3^r s$, and $c_4 := y_1^r y_2^{r\mathcal{H}(c_1,c_2,c_3)}$. The confirmer signature on m is $\sigma := (c_1, c_2, c_3, c_4)$.

Confirmation and disavowal: The verifier chooses $\tilde{h}_2 \in_R \tilde{H}$ and $h_3 \in_R H$. Upon receipt of a request $m \in \{0,1\}^*$, $(n, e) \in \{0,1\}^* \times \mathbb{Z}_n$, $\sigma = (c_1, c_2, c_3, c_4) \in H^4$, and $(\tilde{h}_2, h_3) \in \tilde{H} \times H$ from a verifier the confirmer first decrypts (c_1, c_2, c_3, c_4) and gets a value $\hat{s}$ if decryption does not fail. If $0 < \hat{s} < n$ he sets $s := \hat{s}$ and $s := p - \hat{s}$ otherwise. If $0 < s < n$ and $s^e \equiv Pad_S(m) \pmod n$ holds, the confirmer tells the verifier that the confirmer signature σ is valid and otherwise that it is not valid.

If σ is valid (confirmation): The confirmer computes commitments $C_1 := g_1^s g_2^{v_1}$ and $C_2 := \tilde{h}_1^{\hat{s}} \tilde{h}_2^{v_2}$ with $v_1 \in_R \mathbb{Z}_n$ and $v_2 \in_R \mathbb{Z}_p$ and sends C_1 and C_2 to the verifier. Then, confirmer and verifier carry out the following protocol:

$$
\begin{aligned}
&CZK\text{–}PK\{(\alpha, \beta, \gamma, \varepsilon, \vartheta, \lambda, \rho, \xi, \alpha, \nu_1, \ldots, \nu_6) : \\
&\qquad y_1 = h_1^\gamma h_2^\beta \ \wedge\ y_2 = h_1^\lambda h_2^\rho \ \wedge\ y_3 = h_1^\xi \ \wedge \\
&\qquad \tilde{h}_1^{c_3} = C_2^{c_1^\xi} \tilde{h}_2^{\nu_1} \ \wedge\ c_4 = c_1^\gamma (c_1^{\mathcal{H}(c_1,c_2,c_3)})^\lambda c_2^\beta (c_2^{\mathcal{H}(c_1,c_2,c_3)})^\rho \ \wedge \\
&\qquad \Big(\big(C_2 = \tilde{h}_1^\varepsilon \tilde{h}_2^{\nu_2} \ \wedge\ C_1 = g_1^\varepsilon g_2^{\nu_3} \ \wedge\ (1 \le \varepsilon \le n-1) \big) \vee \\
&\qquad \big(C_2 = (1/\tilde{h}_1)^\vartheta \tilde{h}_2^{\nu_4} \ \wedge\ C_1 = g_1^\vartheta g_2^{\nu_5} \ \wedge\ (1 \le \vartheta \le n-1) \big) \Big) \wedge \\
&\qquad\qquad C_1 = g_1^\alpha g_2^{\nu_6} \ \wedge\ g_1^{Pad_S(m)} = g_1^{\alpha^e} \}.
\end{aligned}
$$

With this protocol the confirmer convinces the verifier that decryption was successful and that either the decrypted value or p minus the decrypted value are a valid RSA signature with respect to m, e, n, and Pad_S. We refer the reader to the full paper for the protocol in all its details.

If σ is not valid (disavowal): If decryption failed, the confirmer chooses $\hat{s} \in_R \mathbb{Z}_p$ and $s \in_R \mathbb{Z}_N$ such that $\left(\frac{\hat{s}}{p}\right) = 1$. Then he computes the following commitments $C_1 := g_1^s g_2^{v_1}$, $C_2 := \tilde{h}_1^{\hat{s}} \tilde{h}_2^{v_2}$, $C_3 := g_1^{s^e} g_2^{v_3}$, $C_4 := h_3^{v_4} c_1^{x_1 + x_2 \mathcal{H}(c_1,c_2,c_3)} c_2^{x_3 + x_4 \mathcal{H}(c_1,c_2,c_3)}$, and $C_5 := \tilde{h}_1^{h_3^{v_4}} \tilde{h}_2^{v_5}$ with $v_1, v_3 \in_R \mathbb{Z}_n$, $v_2, v_5 \in_R \mathbb{Z}_p$, and $v_4 \in_R \mathbb{Z}_q$. He sends $(C_1, C_2, C_3, C_4, C_5)$ to the verifier.

Confirmer and verifier carry out the following protocol:

$$
\begin{aligned}
&\mathit{CZK\text{–}PK}\Big\{(\gamma,\beta,\lambda,\rho,\xi,\delta,\kappa,\alpha_1,\alpha_2,\alpha_3,\nu_1,\ldots,\nu_{14}):\\
&\qquad y_1 = h_1^{\gamma}h_2^{\beta} \ \wedge\ y_2 = h_1^{\lambda}h_2^{\rho} \ \wedge\ y_3 = h_1^{\xi} \ \wedge\\
&\Big((C_4 = h_3^{\delta}c_1^{\gamma}(c_1^{\mathcal{H}(c_1,c_2,c_3)})^{\lambda}c_2^{\beta}(c_2^{\mathcal{H}(c_1,c_2,c_3)})^{\rho} \ \wedge\ C_5 = \tilde{h}_1^{h_3^{\delta}}\tilde{h}_2^{\nu_1} \ \wedge\\
&\qquad \tilde{h}_1 = (\tilde{h}_1^{C_4/c_4}/C_5)^{\kappa}(1/\tilde{h}_2)^{\nu_2}\Big) \ \vee\\
&\Big(\tilde{h}_1^{c_3} = C_2^{c_1^{\xi}}\tilde{h}_2^{\nu_3} \ \wedge\ c_4 = c_1^{\gamma}(c_1^{\mathcal{H}(c_1,c_2,c_3)})^{\lambda}c_2^{\beta}(c_2^{\mathcal{H}(c_1,c_2,c_3)})^{\rho} \ \wedge\\
&\Big((C_2 = \tilde{h}_1^{\alpha_1}\tilde{h}_2^{\nu_4} \wedge C_1 = g_1^{\alpha_1}g_2^{\nu_5} \ \wedge\ C_3 = g_1^{\alpha_1^{e}}g_2^{\nu_6} \wedge\\
&\qquad g_1 = (C_3/g_1^{Pad_S(m)})^{\nu_7}g_2^{\nu_8} \ \wedge\ (1 \le \alpha_1 \le n-1)\Big) \vee\\
&\big(C_2 = (1/\tilde{h}_1)^{\alpha_2}\tilde{h}_2^{\nu_9} \wedge C_1 = g_1^{\alpha_2}g_2^{\nu_{10}} \ \wedge\ C_3 = g_1^{\alpha_2^{e}}g_2^{\nu_{11}} \wedge\\
&\qquad g_1 = (C_3/g_1^{Pad_S(m)})^{\nu_{12}}g_2^{\nu_{13}} \ \wedge\ (1 \le \alpha_2 \le n-1)\big) \vee\\
&\qquad \big(C_2 = \tilde{h}_1^{\alpha_3}\tilde{h}_2^{\nu_{14}} \ \wedge\ (n \le \alpha_3 \le p-n)\big)\Big)\Big)\Big)\Big\}.
\end{aligned}
$$

This protocol proves that either decryption fails or that both the encrypted value and p minus the encrypted value are either not in $[1, n-1]$ or not a valid RSA signature with respect to m, e, n, and Pad_S.

Selective conversion: If (c_1, c_2, c_3, c_4) is a valid confirmer signature, then the confirmer just return the decryption of (c_1, c_2, c_3, c_4) and otherwise answers $\perp$.

Remark 4. As the confirmation and the disavowal protocol involve double discrete logarithms, they are not very efficient because they use binary challenges. If batch verification technology [2] is incorporated, the computational load of both the verifier and the confirmer is about 20 times that of a similar proof with non-binary challenges. Furthermore, the protocol could be made more efficient by allowing non-binary challenges for parts of the protocol. Moreover, if e is small (e.g., 3 or 5), then there is a much more efficient way of proving the knowledge of a root of a discrete log (cf. [7]).

5 Alternative Generic Solutions

Although the two generic schemes presented in [21] are demonstrably insecure, they can both be modified such that they are provably secure in our model. In contrast to the scheme exhibited in Section 3, these schemes cannot provide perfect convertibility with respect to signature schemes such as RSA or DSS. However, they have instances where the confirmation and disavowal protocol is on order of magnitude more efficient than for the scheme described in the previous section. We note that the bi-proof proposed in [21] for disavowal is *not* computational zero-knowledge, however, it can be replaced by a similar proof that is perfect zero-knowledge (we refer to the full version of this paper for details).

The first scheme in [21] is based on signature schemes that are derived from 3-move honest-verifier zero-knowledge proofs of knowledge. The Schnorr signature scheme [26] is a typical example thereof. If an encryption scheme secure against adaptive chosen-ciphertext attacks is used as confirmer commitment scheme and the public keys of the signer and the confirmer are appended to the message that is signed, then the resulting confirmer signature scheme can be proven secure in our model provided that the underlying 3-move proofs of knowledge have the property that the third message is uniquely defined by the first two messages.

The second scheme in [21] is based on signature schemes that are existentially forgeable in their basic variant but become secure if a hash of the message is signed instead of the plain message. The RSA signature scheme is a typical representative for this class of signature schemes. Again, if an encryption scheme secure against adaptive chosen-ciphertext attacks is used, the public keys of the signer and the confirmer are appended to the message, and the signature scheme is deterministic, then the resulting confirmer signature scheme can be shown to be secure in our model.

Details will be given in the forthcoming full version of this paper.

6 Applications to Other Scenarios

As mentioned in [21], confirmer signatures schemes with conversion can be used to realize fair contract signing schemes as follows. The trusted third party in the contract signing scheme plays the role of the confirmer. Furthermore, recall that a signer can always confirm a valid confirmer signature. Thus, a confirmer signature scheme together with a confirmation protocol for the signer can be used to replace the "verifiable signature encryption scheme" in [1]: the parties issue confirmer signatures and prove the correctness of their respective signatures. After this step, either the real signatures can be exchanged or, if this fails, they can ask the TTP/confirmer to convert the confirmer signatures (a suitable policy for the TTP/confirmer should be included in the signed messages). The resulting optimistic fair contract signing scheme can be shown secure in the standard model (i.e., not in the random oracle model) if the security of the underlying signature scheme is assumed.

It is also possible to employ the techniques used for our confirmer signature scheme for realizing verifiable signature sharing schemes [18]. In a nutshell, a promise to a signature is split into shares according to a given secret sharing scheme. Then each of the shares is encrypted (similarly as the ordinary signature in our confirmer signature scheme) and it is proved that the encrypted values are indeed correct shares. Such a proof is similar as the confirmation protocol exhibited in Section 4. This approach is possible for signature schemes such as RSA or DSS. The resulting scheme will enjoy separability and be secure against adaptive attackers while previous solutions were either insecure [18] or secure only in a non-adaptive model [8, 19].

7 Acknowledgements

The authors are grateful to Victor Shoup for various discussions and to the anonymous referees for their helpful and detailed comments.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 591–606, 1998.
2. M. Bellare, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 236–250. Springer Verlag, 1998.
3. M. Bellare and S. Micali. How to sign given any trapdoor function. In *CRYPTO '88*, vol. 403 of *LNCS*, pp. 200–215. Springer-Verlag, 1990.
4. M. Bellare and P. Rogaway. The exact security of digital signature – how to sign with RSA and Rabin. In *EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 399–416. Springer Verlag, 1996.
5. F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT 2000*, vol. 1807 of *LNCS*, pp. 437–450. Springer Verlag, 2000 (this volume).
6. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *CRYPTO '99*, vol. 1296 of *LNCS*, pp. 413–430, 1999.
7. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO '97*, vol. 1296 of *LNCS*, pp. 410–424. Springer Verlag, 1997.
8. D. Catalano and R. Gennaro. New efficient and secure protocols for verifiable signature sharing and other applications. In *CRYPTO '98*, vol. 1642 of *LNCS*, pp. 105–120. Springer Verlag, 1998.
9. D. Chaum. Designated confirmer signatures. In *EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 86–91. Springer Verlag, 1994.
10. D. Chaum and H. van Antwerpen. Undeniable signatures. In *CRYPTO '89*, vol. 435 of *LNCS*, pp. 212–216. Springer-Verlag, 1990.
11. L. Chen. Efficient fair exchange with verifiable confirmation of signatures. In *ASIACRYPT '98*, vol. 1514 of *LNCS*, pp. 286–299. Springer Verlag, 1998.
12. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, vol. 839 of *LNCS*, pp. 174–187. Springer Verlag, 1994.
13. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO '98*, vol. 1642 of *LNCS*, pp. 13–25. Springer Verlag, 1998.
14. G. Di Crescenzo and R. Ostrovsky. On concurrent zero-knowledge with preprocessing. In *CRYPTO '99*, vol. 1296 of *LNCS*, pp. 485–502, 1999.
15. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000*, *LNCS*. Springer Verlag, 2000.
16. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proc. 35th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1994.
17. C. Dwork and A. Sahai. Concurrrent zero-knowledge: Reducing the need for timing constraints. In *CRYPTO '98*, vol. 1642 of *LNCS*, pp. 105–120, 1998.
18. M. Franklin and M. Reiter. Verifiable signature sharing. In *EUROCRYPT '95*, vol. 921 of *LNCS*, pp. 50–63. Springer Verlag, 1995.

19. E. Fujisaki and T. Okamoto. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 32–46. Springer Verlag, 1998.
20. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308.
21. M. Michels and M. Stadler. Generic constructions for secure and efficient confirmer signature schemes. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 406–421, 1998.
22. K. Nguyen, Y. Mu, and V. Varadharajan. Undeniable confirmer signature. Proc. *Information Security Workshop'99*, *LNCS*, Springer-Verlag, 1999.
23. T. Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In *CRYPTO '94*, vol. 839 of *LNCS*, pp. 61–74. Springer Verlag, 1994.
24. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, 21(2):120–126, Feb. 1978.
25. J. Rompel. One-way functions are necessary and sufficient for secure signature. In *Proc. 22nd Annual ACM STOC*, 387–394, 1990.
26. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
27. M. Stadler. Publicly verifiable secret sharing. In *EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 191–199. Springer Verlag, 1996.

A An Insecure Confirmer Signature Scheme

This section show that the scheme due to Chen [11] is insecure because the confirmer can forge confirmer signatures of any message for an arbitrary signer.

Let us review this scheme briefly. Public system parameters are a group $G = \langle g \rangle$ and a prime $q = |G|$. The signer's public key is $y \in G$ and its secret key is $x = \log_g y$. The confirmer's public key is $z \in G$ and its secret key is $w = \log_g z$. Furthermore, a suitable hash function $\mathcal{H}$ is known. The signer generates a confirmer signature on m by picking $u, k_1, k_2 \in_R \mathbb{Z}_q$ and computing $\tilde{y} := y^u$, $\hat{y} := z^{xu}$, $r_1 := y^{k_1}$, $r_2 := z^{k_2}$, $c := \mathcal{H}(m, r_1, r_2)$, $s_1 := k_1 - uc \bmod q$, and $s_2 := k_2 - uxc \bmod q$. The resulting confirmer signature on m is given by $(c, s_1, s_2, \tilde{y}, \hat{y})$. It is valid if and only if $c = h(m, y^{s_1}\tilde{y}^c, z^{s_2}\hat{y}^c)$ and $\log_{\tilde{y}} \hat{y} = \log_g z$. We refer to [11] for a discussion of how the confirmer confirms/disavows.

This scheme is insecure because the confirmer can fake confirmer signatures for an arbitrary signer and message m: He picks random values $t, s_2, d \in_R \mathbb{Z}_q$ and computes $c := h(m, y^t, z^{s_2}y^d)$, $a := d/(wc) \mod q$, $\tilde{y} := y^a$, $\hat{y} := \tilde{y}^w$, and $s_1 := t - ac \mod q$. As $y^t = y^{s_1}\tilde{y}^c$ and $z^{s_2}y^d = z^{s_2}\hat{y}^c$ holds, $(c, s_1, s_2, \tilde{y}, \hat{y})$ is a confirmer signature on the message m. This attack is possible although the security of the scheme is proved in [11]. The problem is that it is erroneously assumed in the security proof that the knowledge extractor learns $log_z \hat{y}$, which is not necessarily the case.

Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements

Mihir Bellare[1], Alexandra Boldyreva[1], and Silvio Micali[2]

[1] Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA.
E-mails: {mihir, aboldyre}@cs.ucsd.edu.
URLs: www-cse.ucsd.edu/users/{mihir, aboldyre}.
[2] MIT Laboratory for Computer Science, 545 Technology Square, Cambridge MA 02139, USA.

Abstract. This paper addresses the security of public-key cryptosystems in a "multi-user" setting, namely in the presence of attacks involving the encryption of related messages under different public keys, as exemplified by Håstad's classical attacks on RSA. We prove that security in the single-user setting implies security in the multi-user setting as long as the former is interpreted in the strong sense of "indistinguishability," thereby pin-pointing many schemes guaranteed to be secure against Håstad-type attacks. We then highlight the importance, in practice, of considering and improving the concrete security of the general reduction, and present such improvements for two Diffie-Hellman based schemes, namely El Gamal and Cramer-Shoup.

1 Introduction

TWO SETTINGS. The setting of public-key cryptography is usually presented like this: there is a receiver R, possession of whose public key pk enables anyone to form ciphertexts which the receiver can decrypt using the secret key associated to pk. This *single-user setting* —so called because it considers a single recipient of encrypted data— is the one of formalizations such as indistinguishability and semantic security [9]. Yet it ignores an important dimension of the problem: in the real world there are many users, each with a public key, sending each other encrypted data. Attacks presented in the early days of public-key cryptography had highlighted the presence of security threats in this *multi-user setting* that were not present in the single-user setting, arising from the possibility that a sender might encrypt, under different public keys, plaintexts which although unknown to the attacker, satisfy some known relation to each other.

HÅSTAD'S ATTACKS. An example of the threats posed by encrypting related messages under different public keys is provided by Håstad's well-known attacks on the basic RSA cryptosystem [10].[1] Suppose we have many users where the

[1] As Håstad points out, the simple version of the attack discussed here was discovered by Blum and others before his work. His own paper considers extensions of the attack

public key of user U_i is an RSA modulus N_i and (for efficiency) all users use encryption exponent $e = 3$. Given a single ciphertext $y_i = m^3 \bmod N_i$, the commonly accepted one-wayness of the RSA function implies that it is computationally infeasible for an adversary to recover the plaintext m. However, suppose now that a sender wants to securely transmit the same plaintext m to three different users, and does so by encrypting m under their respective public keys, producing ciphertexts y_1, y_2, y_3 where $y_i = m^3 \bmod N_i$ for $i = 1, 2, 3$. Then an adversary given y_1, y_2, y_3 can recover m. (Using the fact that N_1, N_2, N_3 are relatively prime, y_1, y_2, y_3 can be combined by Chinese remaindering to yield $m^3 \bmod N_1 N_2 N_3$. But $m^3 < N_1 N_2 N_3$ so m can now be recovered.)

Several counter-measures have been proposed, e.g. padding the message with random bits. The benefit of such measures is, however, unclear in that although they appear to thwart the specific known attacks, we have no guarantee of security against other similar attacks.

A GENERAL REDUCTION. The first and most basic question to address is whether it is possible to prove security against the kinds of attacks discussed above, and if so how and for which schemes. This question turns out to have a simple answer: the schemes permitting security proofs in the multi-user setting are exactly those permitting security proofs in the single-user setting, as long as we use "strong-enough" notions of security in the two cases. What is "strong-enough"? Merely having the property that it is hard to recover the plaintext from a ciphertext is certainly not: basic RSA has this property, yet Håstad's attacks discussed above show it is not secure in the multi-user setting. Theorem 1 interprets "strong enough" for the single-user setting in the natural way: secure in the sense of indistinguishability of Goldwasser and Micali [9]. As to the multi-user setting, the notion used in the theorem is an appropriate extension of indistinguishability that takes into account the presence of multiple users and the possibility of an adversary seeing encryptions of related messages under different public keys. We prove the general reduction for security both under chosen-plaintext attack and chosen-ciphertext attack, in the sense that security under either type of attack in one setting implies security under the same type of attack in the other setting. (The analogous statement can be shown with regard to non-malleability [7] under chosen-plaintext attack, and a simple way to extend our proof to that setting is to exploit the characterization of [5].)

We view ourselves here as establishing what most theoreticians would have "expected" to be true. The proof is indeed simple, yet validating the prevailing intuition has several important elements and fruits beyond the obvious one of filling a gap in the literature, as we now discuss.

IMMEDIATE CONSEQUENCES. The above-mentioned results directly imply security guarantees in the multi-user setting for all schemes proven to meet the notion of indistinguishability, under the same assumptions that were used to establish indistinguishability. This includes practical schemes secure against chosen-

using lattice reduction [10]. For simplicity we will continue to use the term "Håstad's attack(s)" to refer to this body of cryptanalysis.

plaintext attack [8], against chosen-ciphertext attack [6], and against chosen-ciphertext attack in the random oracle model [4, 12].

These results confirm the value of using strong, well-defined notions of security and help to emphasize this issue in practice. As we have seen, designers attempt to thwart Håstad-type attacks by specific counter-measures. Now we can say that the more productive route is to stick to schemes meeting notions of security such as indistinguishability. Designers are saved the trouble of explicitly considering attacks in the multi-user setting.

THE MODEL. The result requires, as mentioned above, the introduction of a new model and notion. We want to capture the possibility of an adversary seeing encryptions of related messages under different keys when the choice of the relation can be made by the adversary. To do this effectively and elegantly turns out to need some new definitional ideas. Very briefly —see Section 3 for a full discussion and formalization— the formalization introduces the idea of an adversary given (all public keys and) a list of "challenge encryption oracles," one per user, each oracle capable of encrypting one of two given equal-length messages, the choice of which being made according to a bit that *although hidden from the adversary is the same for all oracles.*[2] This obviates the need to *explicitly* consider relations amongst messages. This model is important because its use extends beyond Theorem 1, as we will see below.

ISN'T SIMULATION ENOUGH? It may appear at first glance that the implication (security in the single-user setting implies security in the multi-user setting for strong-enough notions of security) is true for a trivial reason: an adversary attacking one user can just simulate the other users, itself picking their public keys so that it knows the corresponding secret keys. This doesn't work, and misses the key element of the multi-user setting. Our concern is an adversary that sees ciphertexts of related messages under different keys. Given a challenge ciphertext of an unknown message under a target public key, a simulator cannot produce a ciphertext of a related message under a different public key, even if it knows the secret key corresponding to the second public key, because it does not know the original message. Indeed, our proof does not proceed by this type of simulation.

THE NEED FOR CONCRETE SECURITY IMPROVEMENTS. Perhaps the most important impact of the general reduction of Theorem 1 is the manner in which it leads us to see the practical importance of concrete security issues and improvements for the multi-user setting.

Suppose we have a system of n users in which each user encrypts up to q_e messages. We fix a public-key cryptosystem $\mathcal{PE}$ used by all users. Theorem 1 says that the maximum probability that an adversary with running time t can

[2] An encryption oracle is used in definitions of security for private-key encryption [3] because there the encryption key is secret, meaning not given to the adversary. One might imagine that oracles performing encryption are unnecessary in the public-key case because the adversary knows the public keys: can't it just encrypt on its own? Not when the message in question is a challenge one which it doesn't know, as in our setting.

compromise security in the multi-user setting —this in the sense of our definition discussed above— is at most $q_e n$ times the maximum probability that an adversary with running time closely related to t can compromise security in the standard sense of indistinguishability. Notationally, $\mathsf{Adv}^{n\text{-cpa}}_{\mathcal{PE},I}(t, q_e) \leq q_e n \cdot \mathsf{Adv}^{1\text{-cpa}}_{\mathcal{PE},I}(t')$ where $t' \approx t$. (Here I represents any possible information common to all users and should be ignored at a first reading, and the technical term for the "maximum breaking probabilities" represented by the notation is "advantage".) It follows that if any poly-time adversary has negligible success probability in the single-user setting, the same is true in the multi-user setting. This corollary is what we have interpreted above as saying that "the schemes secure in the single-user setting are exactly those secure in the multi-user setting". However, what this theorem highlights is that the advantage in the multi-user setting may be more than that in the single-user setting by a factor of $q_e n$. Security can degrade linearly as we add more users to the system and also as the users encrypt more data. The practical impact of this is considerable, and in the full version of this work [2] we illustrate this with some numerical examples that are omitted here due to lack of space.

We prove in Proposition 1 that there is no general reduction better than ours: if there is any secure scheme, there is also one whose advantage in the two settings provably differs by a factor of $q_e n$. So we can't expect to reduce the security loss in general. But we can still hope that there are *specific* schemes for which the security degrades less quickly as we add more users to the system. These schemes become attractive in practice because for a fixed level of security they have lower computational cost than schemes not permitting such improved reductions. We next point to two popular schemes for which we can provide new security reductions illustrating such improvements.

El Gamal. The El Gamal scheme in a group of prime order can be proven to have the property of indistinguishability under chosen-plaintext attack (in the single-user setting) under the assumption that the decision Diffie-Hellman (DDH) problem is hard. (This simple observation is made for example in [11, 6]). The reduction is essentially tight, meaning that the maximum probability that an adversary of time-complexity t can compromise the security of the El Gamal scheme in the single-user setting is within a constant factor of the probability of solving the DDH problem in comparable time. Theorem 1 then implies that the maximum probability of breaking the El Gamal scheme under chosen-plaintext attack in the presence of n users each encrypting q_e messages is bounded by $2q_e n$ times the probability of solving the DDH problem in comparable time. We show in Theorem 2 that via an improved reduction the factor of $q_e n$ can be essentially eliminated. In other words, the maximum probability of breaking the El Gamal scheme under chosen-plaintext attack, even in the presence of n users each encrypting q_e messages, remains tightly related to the probability of solving the DDH problem in comparable time.

Our reduction exploits a self-reducibility property of the decisional Diffie-Hellman problem due to Stadler and Naor-Reingold [15, 11], and a variant thereof that was also independently noted by Shoup [14]. See Lemma 1.

CRAMER-SHOUP. The Cramer-Shoup scheme [6] is shown to achieve indistinguishability under chosen-ciphertext attack (in the single-user setting) assuming the DDH problem is hard. Their reduction of the security of their scheme to that of the DDH problem is essentially tight. Applying our general result to bound the advantage in the multi-user setting would indicate degradation of security by a factor of $q_e n$. We present in Theorem 3 an improved reduction which (roughly speaking) reduces the factor of $q_e n$ to a factor of q_e only. Thus the maximum probability of breaking the Cramer-Shoup scheme under chosen-ciphertext attack, in the presence of n users, each encrypting q_e messages, is about the same as is proved if there was only one user encrypting q_e messages. (The result is not as strong as for El Gamal because we have not eliminated the factor of q_e, but this is an open problem even when there is only one user.) This new result exploits Lemma 1 and features of the proof of security for the single-user case given in [6].

DISCUSSION AND RELATED WORK. A special case of interest in these results is when $n = 1$. Meaning we are back in the single-user setting, but are looking at an extension of the notion of indistinguishability in which one considers the encryption of up to q_e messages. Our results provide improved security for the El Gamal scheme in this setting.

The questions raised here can also be raised in the private-key setting: what happens there when there are many users? The ideas of the current work are easily transfered. The definitions of [3] for the single-user case can be adapted to the multi-user case using the ideas in Section 3. The analogue of Theorem 1 for the private-key setting is then easily proven.

Baudron, Pointcheval and Stern have independently considered the problem of public-key encryption in the multi-user setting [1]. Their notion of security for the multi-user setting —also proved to be polynomially-equivalent to the standard notion of single-user indistinguishability— is slightly different from ours. They do not consider concrete-security or any specific schemes. (The difference in the notions is that they do not use the idea of encryption oracles; rather, their adversary must output a pair of vectors of plaintexts and get back as challenge a corresponding vector of ciphertexts. This makes their model weaker since the adversary does not have adaptive power. If only polynomial-security is considered, their notion, ours and the single-user one are all equivalent, but when concrete security is considered, our notion is stronger.)

2 Definitions

We specify a concrete-security version of the standard notion of security of a public-key encryption scheme in the sense of indistinguishability. We consider both chosen-plaintext and chosen-ciphertext attacks.

First recall that a *public-key encryption scheme* $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms. The *key generation* algorithm $\mathcal{K}$ is a randomized algorithm that takes as input some global information I and returns a pair (pk, sk) of keys, the public key and matching secret key, respectively; we write $(pk, sk) \stackrel{R}{\leftarrow}$

$\mathcal{K}(I)$. (Here I includes a security parameter, and perhaps other information. For example in a Diffie-Hellman based scheme, I might include a global prime number and generator of a group which all parties use to create their keys.) The *encryption* algorithm $\mathcal{E}$ is a randomized algorithm that takes the public key pk and a *plaintext* M to return a *ciphertext* C; we write $C \xleftarrow{R} \mathcal{E}_{pk}(M)$. The *decryption* algorithm $\mathcal{D}$ is a deterministic algorithm that takes the secret key sk and a ciphertext C to return the corresponding plaintext M; we write $M \leftarrow \mathcal{D}_{sk}(C)$. Associated to each public key pk is a *message space* $\mathrm{MsgSp}(pk)$ from which M is allowed to be drawn. We require that $\mathcal{D}_{sk}(\mathcal{E}_{pk}(M)) = M$ for all $M \in \mathrm{MsgSp}(pk)$.

An adversary B runs in two stages. In the "find" stage it takes the public key and outputs two equal length messages m_0, m_1 together with some state information s. In the "guess" stage it gets a challenge ciphertext C formed by encrypting a random one of the two messages, and must say which message was chosen. Below the superscript of "1" indicates that we are in the single-user setting, meaning that although there may be many senders, only one person holds a public key and is the recipient of encrypted information. In the case of a chosen-ciphertext attack the adversary gets an oracle for $\mathcal{D}_{sk}(\cdot)$ and is allowed to invoke it on any point with the restriction of not querying the challenge ciphertext during the guess stage [13].

Definition 1. [Indistinguishability of encryptions] Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let $B_{\mathrm{cpa}}, B_{\mathrm{cca}}$ be adversaries where the latter has access to an oracle. Let I be some initial information string. For $b = 0, 1$ define the experiments

Experiment $\mathbf{Exp}^{\text{1-cpa}}_{\mathcal{PE},I}(B_{\mathrm{cpa}}, b)$	**Experiment** $\mathbf{Exp}^{\text{1-cca}}_{\mathcal{PE},I}(B_{\mathrm{cca}}, b)$
$(pk, sk) \leftarrow \mathcal{K}(I)$	$(pk, sk) \leftarrow \mathcal{K}(I)$
$(m_0, m_1, s) \leftarrow B_{\mathrm{cpa}}(\mathsf{find}, I, pk)$	$(m_0, m_1, s) \leftarrow B^{\mathcal{D}_{sk}(\cdot)}_{\mathrm{cca}}(\mathsf{find}, I, pk)$
$C \xleftarrow{R} \mathcal{E}_{pk}(m_b)$	$C \xleftarrow{R} \mathcal{E}_{pk}(m_b)$
$d \leftarrow B_{\mathrm{cpa}}(\mathsf{guess}, C, s)$	$d \leftarrow B^{\mathcal{D}_{sk}(\cdot)}_{\mathrm{cca}}(\mathsf{guess}, C, s)$
`Return` d	`Return` d

It is mandated that $|m_0| = |m_1|$ above. We require that B_{cca} not make oracle query C in the guess stage. We define the *advantage* of B_{cpa} and B_{cca}, respectively, as follows:

$$\mathsf{Adv}^{\text{1-cpa}}_{\mathcal{PE},I}(B_{\mathrm{cpa}}) = \Pr\left[\mathbf{Exp}^{\text{1-cpa}}_{\mathcal{PE},I}(B_{\mathrm{cpa}}, 0) = 0\right] - \Pr\left[\mathbf{Exp}^{\text{1-cpa}}_{\mathcal{PE},I}(B_{\mathrm{cpa}}, 1) = 0\right]$$

$$\mathsf{Adv}^{\text{1-cca}}_{\mathcal{PE},I}(B_{\mathrm{cca}}) = \Pr\left[\mathbf{Exp}^{\text{1-cca}}_{\mathcal{PE},I}(B_{\mathrm{cca}}, 0) = 0\right] - \Pr\left[\mathbf{Exp}^{\text{1-cca}}_{\mathcal{PE},I}(B_{\mathrm{cca}}, 1) = 0\right] .$$

We define the *advantage function of the scheme for privacy under chosen-plaintext (resp. chosen-ciphertext) attacks in the single-user setting* as follows. For any t, q_{d}, let

$$\mathsf{Adv}^{\text{1-cpa}}_{\mathcal{PE},I}(t) = \max_{B_{\mathrm{cpa}}} \{\, \mathsf{Adv}^{\text{1-cpa}}_{\mathcal{PE},I}(B_{\mathrm{cpa}}) \,\}$$

$$\mathsf{Adv}^{\text{1-cca}}_{\mathcal{PE},I}(t, q_{\mathrm{d}}) = \max_{B_{\mathrm{cca}}} \{\, \mathsf{Adv}^{\text{1-cca}}_{\mathcal{PE},I}(B_{\mathrm{cca}}) \,\}$$

where the maximum is over all $B_{\rm cpa}, B_{\rm cca}$ with "time-complexity" t, and, in the case of $B_{\rm cca}$, also making at most $q_{\rm d}$ queries to the $\mathcal{D}_{sk}(\cdot)$ oracle. ▮

The "time-complexity" is the worst case execution time of the associated experiment plus the size of the code of the adversary, in some fixed RAM model of computation. (Note that the execution time refers to the entire experiment, not just the adversary. In particular, it includes the time for key generation, challenge generation, and computation of responses to oracle queries if any.) The same convention is used for all other definitions in this paper and will not be explicitly mentioned again. The advantage function is the maximum likelihood of the security of the encryption scheme $\mathcal{PE}$ being compromised by an adversary, using the indicated resources, and with respect to the indicated measure of security.

Definition 2. We say that $\mathcal{PE}$ is *polynomially-secure against chosen-plaintext attack (resp. chosen-ciphertext attack) in the single-user setting* if $\mathsf{Adv}^{\text{1-cpa}}_{\mathcal{PE},I}(B)$ (resp. $\mathsf{Adv}^{\text{1-cca}}_{\mathcal{PE},I}(B)$) is negligible for any probabilistic, poly-time adversary B.

Here complexity is measured as a function of a security parameter that is contained in the global input I. If I consists of more than a security parameter (as in the El Gamal scheme), we fix a probabilistic generator for this information and the probability includes the choices of this generator.

3 Security in the multi-user setting

We envision a set of n users. All users use a common, fixed cryptosystem $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. User i has a public key pk_i and holds the matching secret key sk_i. It is assumed that each user has an authentic copy of the public keys of all other users.

As with any model for security we need to consider attacks (what the adversary is allowed to do) and success measures (when is the adversary considered successful). The adversary is given the global information I and also the public keys of all users. The main novel concern is that the attack model must capture the possibility of an adversary obtaining encryptions of related messages under different keys. To have a strong notion of security, we will allow the adversary to choose how the messages are related, and under which keys they are encrypted. For simplicity we first address chosen-plaintext attacks only.

SOME INTUITION. To get a start on the modeling, consider the following game. We imagine that a message m is chosen at random from some known distribution, and the adversary is provided with $\mathcal{E}_{pk_1}(m)$, a ciphertext of m under the public key of user 1. The adversary's job is to compute some partial information about m. To do this, it may, for example, like to see an encryption of m under pk_3. We allow it to ask for such an encryption. More generally, it may want to see an encryption of the bitwise complement of m under yet another key, or perhaps the encryption of an even more complex function of m. We could capture this

by allowing the adversary to specify a polynomial-time "message modification function" Δ and a user index j, and obtain in response $\mathcal{E}_{pk_j}(\Delta(m))$, a ciphertext of the result of applying the modification function to the challenge message. After many such queries, the adversary must output a guess of some partial information about m and wins if it can do this with non-trivial advantage. Appropriately generalized, these ideas can be used to produce a semantic-security type notion of security for the multi-user setting, but, as should be evident even from our brief discussion here, it would be relatively complex. We prefer an indistinguishability version because it is simpler and extends more easily to a concrete security setting. It is nonetheless useful to discuss the semantic security setting because here we model the attacks in which we are interested in a direct way that helps provide intuition.

INDISTINGUISHABILITY BASED APPROACH. The adversary is provided with all the public keys. But unlike in the single-user indistinguishability setting of Section 2, it will not run in two phases, and there will be no single challenge ciphertext. Rather the adversary is provided with n different oracles $\mathcal{O}_1, \ldots, \mathcal{O}_n$. Oracle i takes as input any pair m_0, m_1 of messages (of equal length) and computes and returns a ciphertext $\mathcal{E}_{pk_i}(m_b)$. The challenge bit b here (obviously not explicitly given to the adversary) is chosen only once at the beginning of the experiment and *is the same across all oracles and queries.* The adversary's success is measured by its advantage in predicting b.

We suggest that this simple model in fact captures encryption of related messages under different keys; the statement in the italicized text above is crucial in this regard. The possibility of the adversary's choosing the relations between encrypted messages is captured implicitly; we do not have to worry about explicitly specifying message modification functions.

THE FORMAL DEFINITION. Formally, the *left or right selector* is the map LR defined by $\mathrm{LR}(m_0, m_1, b) = m_b$ for all equal-length strings m_0, m_1, and for any $b \in \{0,1\}$. The adversary A is given n oracles, which we call *LR (left-or-right) encryption oracles*,

$$\mathcal{E}_{pk_1}(\mathrm{LR}(\cdot,\cdot,b)) \;,\; \ldots \;,\; \mathcal{E}_{pk_n}(\mathrm{LR}(\cdot,\cdot,b))$$

where pk_i is a public key of the encryption scheme and b is a bit whose value is unknown to the adversary. (LR oracles were first defined by [3] in the symmetric setting.) The oracle $\mathcal{E}_{pk_i}(\mathrm{LR}(\cdot,\cdot,b))$, given query (m_0, m_1) where $m_0, m_1 \in \mathrm{MsgSp}(pk_i)$ must have equal length, first sets $m_b \leftarrow \mathrm{LR}(m_0, m_1, b)$, meaning m_b is one of the two query messages, as dictated by bit b. Next the oracle encrypts m_b, setting $C \leftarrow \mathcal{E}_{pk_i}(m_b)$ and returns C as the answer to the oracle query. The adversary also gets as input the public keys and the global information I.

In the case of a chosen-ciphertext attack the adversary is also given a decryption oracle with respect to each of the n public keys. Note we must disallow a query C to $\mathcal{D}_{sk_i}(\cdot)$ if C is an output of oracle $\mathcal{E}_{pk_i}(\mathrm{LR}(\cdot,\cdot,b))$. This is necessary for meaningfulness since if such a query is allowed b is easily computed, and moreover disallowing such queries seems the least limitation we can impose,

meaning the adversary has the maximum meaningful power. Below we indicate the number n of users as a superscript.

Definition 3. Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let $A_{\rm cpa}$, $A_{\rm cca}$ be adversaries. Both have access to $n \geq 1$ oracles, each of which takes as input any two strings of equal length, and $A_{\rm cca}$ has access to an additional n oracles each of which take a single input. Let I be some initial information string. For $b = 0, 1$ define the experiments:

Experiment $\mathbf{Exp}^{n\text{-cpa}}_{\mathcal{PE},I}(A_{\rm cpa}, b)$
  For $i = 1, \ldots, n$ do $(pk_i, sk_i) \leftarrow \mathcal{K}(I)$ EndFor
  $d \leftarrow A_{\rm cpa}^{\mathcal{E}_{pk_1}(\mathrm{LR}(\cdot,\cdot,b)), \ldots, \mathcal{E}_{pk_n}(\mathrm{LR}(\cdot,\cdot,b))}(I, pk_1, \ldots, pk_n)$; Return d

Experiment $\mathbf{Exp}^{n\text{-cca}}_{\mathcal{PE},I}(A_{\rm cca}, b)$
  For $i = 1, \ldots, n$ do $(pk_i, sk_i) \leftarrow \mathcal{K}(I)$ EndFor
  $d \leftarrow A_{\rm cca}^{\mathcal{E}_{pk_1}(\mathrm{LR}(\cdot,\cdot,b)), \ldots, \mathcal{E}_{pk_n}(\mathrm{LR}(\cdot,\cdot,b)), \mathcal{D}_{sk_1}(\cdot), \ldots, \mathcal{D}_{sk_n}(\cdot)}(I, pk_1, \ldots, pk_n)$
  Return d

It is mandated that a query to any LR oracle consists of two messages of *equal* length and that for each $i = 1, \ldots, n$ adversary $A_{\rm cca}$ does not query $\mathcal{D}_{sk_i}(\cdot)$ on an output of $\mathcal{E}_{pk_i}(\mathrm{LR}(\cdot, \cdot, b))$ We define the *advantage* of $A_{\rm cpa}$, and the *advantage* of $A_{\rm cca}$, respectively, as follows:

$$\mathsf{Adv}^{n\text{-cpa}}_{\mathcal{PE},I}(A_{\rm cpa}) = \Pr\left[\mathbf{Exp}^{n\text{-cpa}}_{\mathcal{PE},I}(A_{\rm cpa}, 0) = 0\right] - \Pr\left[\mathbf{Exp}^{n\text{-cpa}}_{\mathcal{PE},I}(A_{\rm cpa}, 1) = 0\right]$$
$$\mathsf{Adv}^{n\text{-cca}}_{\mathcal{PE},I}(A_{\rm cca}) = \Pr\left[\mathbf{Exp}^{n\text{-cca}}_{\mathcal{PE},I}(A_{\rm cca}, 0) = 0\right] - \Pr\left[\mathbf{Exp}^{n\text{-cca}}_{\mathcal{PE},I}(A_{\rm cca}, 1) = 0\right] .$$

We define the *advantage function of the scheme for privacy under chosen- plaintext (resp. chosen-ciphertext) attacks, in the multi-user setting*, as follows. For any $t, q_{\rm e}, q_{\rm d}$ let

$$\mathsf{Adv}^{n\text{-cpa}}_{\mathcal{PE},I}(t, q_{\rm e}) = \max_{A_{\rm cpa}} \{\, \mathsf{Adv}^{n\text{-cpa}}_{\mathcal{PE},I}(A_{\rm cpa}) \,\}$$
$$\mathsf{Adv}^{n\text{-cca}}_{\mathcal{PE},I}(t, q_{\rm e}, q_{\rm d}) = \max_{A_{\rm cca}} \{\, \mathsf{Adv}^{n\text{-cca}}_{\mathcal{PE},I}(A_{\rm cca}) \,\}$$

where the maximum is over all $A_{\rm cpa}, A_{\rm cca}$ with "time-complexity" t, making at most $q_{\rm e}$ queries to each LR oracle, and, in the case of $A_{\rm cca}$, also making at most $q_{\rm d}$ queries to each decryption oracle. ∎

The advantage function is the maximum likelihood of the security of the symmetric encryption scheme $\mathcal{PE}$ being compromised by an adversary, using the indicated resources, and with respect to the indicated measure of security.

Remark 1. Notice that when $n = q_{\rm e} = 1$ in Definition 3, the adversary's capability is limited to seeing a ciphertext of one of two messages of its choice under a single target key. Thus Definition 3 with $n = q_{\rm e} = 1$ is equivalent to Definition 1. We can view Definition 3 as extending Definition 1 along two dimensions: the number of users and the number of messages encrypted by each user.

Definition 4. We say that $\mathcal{PE}$ is *polynomially-secure against chosen-plaintext (resp. chosen-ciphertext) attack in the multi-user setting* if $\mathsf{Adv}^{n\text{-cpa}}_{\mathcal{PE},I}(A)$ (resp. $\mathsf{Adv}^{n\text{-cca}}_{\mathcal{PE},I}(A)$) is negligible for any probabilistic, poly-time adversary A and polynomial n.

Again complexity is measured as a function of a security parameter that is contained in the global input I, and the latter is generated by a fixed probabilistic polynomial-time generation algorithm if necessary.

4 A general reduction and its tightness

Fix a public-key encryption scheme $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The following theorem says that the advantage of an adversary in breaking the scheme in a multi-user setting can be upper bounded by a function of the advantage of an adversary of comparable resources in breaking the scheme in the single-user setting. The factor in the bound is polynomial in the number n of users in the system and the number q_{e} of encryptions performed by each user, and the theorem is true for both chosen-plaintext attacks and chosen-ciphertext attacks. The proof of Theorem 1 is via a simple hybrid argument that is omitted here due to lack of space but can be found in the full version of this paper [2].

Theorem 1. *Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let $n, q_{\mathrm{e}}, q_{\mathrm{d}}$, t be integers and I some initial information string. Then*

$$\mathsf{Adv}^{n\text{-cpa}}_{\mathcal{PE},I}(t, q_{\mathrm{e}}) \le q_{\mathrm{e}} n \cdot \mathsf{Adv}^{1\text{-cpa}}_{\mathcal{PE},I}(t')$$
$$\mathsf{Adv}^{n\text{-cca}}_{\mathcal{PE},I}(t, q_{\mathrm{e}}, q_{\mathrm{d}}) \le q_{\mathrm{e}} n \cdot \mathsf{Adv}^{1\text{-cca}}_{\mathcal{PE},I}(t', q_{\mathrm{d}})$$

where $t' = t + O(\log(q_{\mathrm{e}} n))$. ∎

The relation between the advantages being polynomial, we obviously have the following:

Corollary 1. *Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme that is polynomially-secure against chosen-plaintext (resp. chosen-ciphertext) attack in the single-user setting. Then $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is also polynomially-secure against chosen-plaintext (resp. chosen-ciphertext) attack in the multi-user setting.* ∎

TIGHTNESS OF THE BOUND. We present an example that shows that in general the bound of Theorem 1 is essentially tight. Obviously such a statement is vacuous if no secure schemes exist, so first assume one does, and call it $\mathcal{PE}$. We want to modify this into another scheme $\mathcal{PE}'$ for which $\mathsf{Adv}^{n\text{-cpa}}_{\mathcal{PE}',I}(t, q_{\mathrm{e}})$ is $\Omega(q_{\mathrm{e}} n)$ times $\mathsf{Adv}^{1\text{-cpa}}_{\mathcal{PE}',I}(t)$. This will be our counter-example. The following proposition does this, modulo some technicalities. In reading it, think of $\mathcal{PE}$ as being very good, so that $\mathsf{Adv}^{1\text{-cpa}}_{\mathcal{PE},I}(t)$ is essentially zero. With that interpretation we indeed have the claimed relation.

Proposition 1. *Given any public-key encryption scheme $\mathcal{PE}$ and integers n, q_e we can design another public-key encryption $\mathcal{PE}'$ such that for any I and large enough t we have*

$$\mathsf{Adv}^{n\text{-cpa}}_{\mathcal{PE}',I}(t, q_e) \geq 0.6 \text{ and } \mathsf{Adv}^{1\text{-cpa}}_{\mathcal{PE}',I}(t) \leq \frac{1}{q_e n} + \mathsf{Adv}^{1\text{-cpa}}_{\mathcal{PE},I}(t) \ . \ \blacksquare$$

The proof of Proposition 1 is in [2]. An analogous result holds in the chosen-ciphertext attack case, and we omit it.

5 Improved security for DH based schemes

The security of the schemes we consider is based on the hardness of the Decisional Diffie-Hellman (DDH) problem. Accordingly we begin with definitions for latter.

Definition 5. Let G be a group of a large prime order q and let g be a generator of G. Let D be an adversary that on input q, g and three elements $X, Y, K \in G$ returns a bit. We consider the experiments

Experiment $\mathbf{Exp}^{\text{ddh-real}}_{q,g}(D)$	Experiment $\mathbf{Exp}^{\text{ddh-rand}}_{q,g}(D)$
$x \xleftarrow{R} Z_q \ ; \ X \leftarrow g^x$	$x \xleftarrow{R} Z_q \ ; \ X \leftarrow g^x$
$y \xleftarrow{R} Z_q \ ; \ Y \leftarrow g^y$	$y \xleftarrow{R} Z_q \ ; \ Y \leftarrow g^y$
$K \leftarrow g^{xy}$	$K \xleftarrow{R} G$
$d \leftarrow D(q, g, X, Y, K)$	$d \leftarrow D(q, g, X, Y, K)$
Return d	Return d

The advantage of D in solving the Decisional Diffie-Hellman (DDH) problem with respect to q, g, and the advantage of the DDH with respect to q, g, are defined, respectively, by

$$\mathsf{Adv}^{\text{ddh}}_{q,g}(D) = \Pr\left[\mathbf{Exp}^{\text{ddh-real}}_{q,g}(D) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ddh-rand}}_{q,g}(D) = 1\right]$$
$$\mathsf{Adv}^{\text{ddh}}_{q,g}(t) = \max_D \{\, \mathsf{Adv}^{\text{ddh}}_{q,g}(D) \,\}$$

where the maximum is over all D with "time-complexity" t. $\blacksquare$

The "time-complexity" of D is the maximum of the execution times of the two experiments $\mathbf{Exp}^{\text{ddh-real}}_{q,g}(D)$ and $\mathbf{Exp}^{\text{ddh-rand}}_{q,g}(D)$, plus the size of the code for D, all in our fixed RAM model of computation.

A common case is that G is a subgroup of order q of Z_p^* where p is a prime such that q divides $p-1$. But these days there is much interest in the use of Diffie-Hellman based encryption over elliptic curves, where G would be an appropriate elliptic curve group. Our setting is general enough to encompass both cases.

Our improvements exploit in part some self-reducibility properties of the DDH problem summarized in Lemma 1 below. The case $x \neq 0$ below is noted in [15, Proposition 1] and [11, Lemma 3.2]. The variant with $x = 0$ was noted independently in [14]. Below T^{exp}_q denotes the time needed to perform an exponentiation operation with respect to a base element in G and an exponent in Z_q, in our fixed RAM model of computation. A proof of Lemma 1 is in [2].

Lemma 1. *Let G be a group of a large prime order q and let g be a generator of G. There is a probabilistic algorithm R running in $O(T_q^{\exp})$ time such for any a, b, c, x in Z_q the algorithm takes input q, g, g^a, g^b, g^c, x and returns a triple $g^{a'}, g^{b'}, g^{c'}$ such that the properties represented by the following table are satisfied, where we read the row and column headings as conditions, and the table entries as the properties of the outputs under those conditions:*

	$x = 0$	$x \neq 0$
$c = ab \bmod q$	$a' = a$ b' *is random* $c' = a'b' \bmod q$	a' *is random* b' *is random* $c' = a'b' \bmod q$
$c \neq ab \bmod q$	$a' = a$ b' *is random* c' *is random*	a' *is random* b' *is random* c' *is random*

Here random means distributed uniformly over Z_q independently of anything else. ∎

El Gamal. As indicated above, our reduction of multi-user security to single-user security is tight in general. Here we will obtain a much better result for a specific scheme, namely the El Gamal encryption scheme over a group of prime order, by exploiting Lemma 1. We fix a group G for which the decision Diffie-Hellman problem is hard and let q (a prime) be its size. Let g be a generator of G. The prime q and the generator g comprise the global information I for the El Gamal scheme. The algorithms describing the scheme $\mathcal{EG} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ are depicted below. The message space associated to a public key (q, g, X) is the group G itself, with the understanding that all messages from G are properly encoded as strings of some common length whenever appropriate.

Algorithm $\mathcal{K}(q, g)$	Algorithm $\mathcal{E}_{q,g,X}(M)$	Algorithm $\mathcal{D}_{q,g,x}(Y, W)$
$x \xleftarrow{R} Z_q$	$y \xleftarrow{R} Z_q$	$K \leftarrow Y^x$
$X \leftarrow g^x$	$Y \leftarrow g^y$	$M \leftarrow WK^{-1}$
$pk \leftarrow (q, g, X)$	$K \leftarrow X^y$	`Return` M
$sk \leftarrow (q, g, x)$	$W \leftarrow KM$	
`Return` (pk, sk)	`Return` (Y, W)	

We noted in Section 1 that the hardness of the DDH problem implies that the El Gamal scheme meets the standard notion of indistinguishability of encryptions (cf.[11, 6]), and the reduction is essentially tight: $\mathsf{Adv}^{1\text{-cpa}}_{\mathcal{EG},(q,g)}(t)$ is at most $2\mathsf{Adv}^{\text{ddh}}_{q,g}(t)$. We want to look at the security of the El Gamal scheme in the multi-user setting. Directly applying Theorem 1 in conjunction with the above would tell us that

$$\mathsf{Adv}^{n\text{-cpa}}_{\mathcal{EG},(q,g)}(t, q_e) \leq 2q_e n \cdot \mathsf{Adv}^{\text{ddh}}_{q,g}(t') \tag{1}$$

where $t' = t + O(\log(q_e n))$. This is enough to see that polynomial security of the DDH problem implies polynomial security of El Gamal in the multi-user

setting, but we want to improve the concrete security of this relation and say that the security of the El Gamal scheme in the multi-user setting almost does not degrade with respect to the assumed hardness of the DDH problem. The following theorem states our improvement.

Theorem 2. *Let G be a group of a large prime order q and let g be a generator of the group G. Let $\mathcal{EG} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the El Gamal public-key encryption scheme associated to these parameters as described above. Let n, q_e, t be integers. Then*

$$\mathsf{Adv}^{n\text{-cpa}}_{\mathcal{EG},(q,g)}(t, q_e) \;\le\; 2 \cdot \mathsf{Adv}^{\text{ddh}}_{q,g}(t') + \frac{1}{q}$$

where $t' = t + O(q_e n \cdot T^{\exp}_q)$. ∎

The $1/q$ term is negligible in practice since q is large, so the theorem is saying that the security of the encryption scheme is within a constant factor of that of the DDH problem, even where there are many users and the time-complexities are comparable.

Proof of Theorem 2: Let A be an adversary attacking the El Gamal public-key encryption scheme $\mathcal{EG}$ in the multi-user setting (cf. Definition 3). Suppose it makes at most q_e queries to each of its n oracles and has time-complexity at most t. We will design an adversary D_A for the Decisional Diffie-Hellman problem (cf. Definition 5) so that D_A has running time at most t' and

$$\mathsf{Adv}^{\text{ddh}}_{q,g}(D_A) \;\ge\; \frac{1}{2} \cdot \mathsf{Adv}^{n\text{-cpa}}_{\mathcal{EG},(q,g)}(A) - \frac{1}{2q} \,. \tag{2}$$

The statement of theorem follows by taking maximums. So it remains to specify D_A. The code for D_A is presented in Figure 1. It has input q, g, and also three elements $X, Y, K \in G$. It will use adversary A as a subroutine. D_A will provide for A as input public keys $pk_1, \ldots, pk_n$ and global information q, g and will simulate for A the n LR oracles, $\mathcal{E}_{pk_i}(\mathrm{LR}(\cdot,\cdot,b))$ for $i = 1, \ldots, n$. We use the notation $A \to (i, m_0, m_1)$ to indicate that A is making query (m_0, m_1) to its i-th LR oracle, where $1 \le i \le n$ and $|m_0| = |m_1|$. We use the notation $A \leftarrow C$ to indicate that we are returning ciphertext C to A as the response to this LR oracle query. We are letting R denote the algorithm of Lemma 1.

An analysis of this algorithm —which is omitted here due to lack of space but can be found in [2]— shows that

$$\Pr\left[\, \mathbf{Exp}^{\text{ddh-real}}_{q,g}(D) = 1 \,\right] = \frac{1}{2} + \frac{1}{2} \cdot \mathsf{Adv}^{n\text{-cpa}}_{\mathcal{EG},(q,g)}(A) \,. \tag{3}$$

and

$$\Pr\left[\, \mathbf{Exp}^{\text{ddh-rand}}_{q,g}(D) = 1 \,\right] \;\le\; \frac{1}{2} \cdot \left(1 - \frac{1}{q}\right) + \frac{1}{q} \;=\; \frac{1}{2} + \frac{1}{2q} \,. \tag{4}$$

Subtracting Equations (3) and (4) we get Equation (2). □

Adversary $D_A(q, g, X, Y, K)$
  $b \xleftarrow{R} \{0,1\}$
  `For` $i = 1, \ldots, n$ `do`
    $(X_i'[1], Y_i'[1], K_i'[1]) \leftarrow R(q, g, X, Y, K, 1)$; $pk_i \leftarrow (q, g, X_i'[1])$; $ctr_i \leftarrow 0$
    `For` $j = 2, \ldots, q_e$ `do`
      $(X_i'[j], Y_i'[j], K_i'[j]) \leftarrow R(q, g, X_i'[1], Y_i'[1], K_i'[1], 0)$
    `EndFor`
  `EndFor`
  Run A replying to oracle queries as follows:
    $A \rightarrow (i, m_0, m_1)$ $[1 \le i \le n$ and $m_0, m_1 \in G]$
    $ctr_i \leftarrow ctr_i + 1$; $W_i \leftarrow K_i'[ctr_i] \cdot m_b$
    $A \leftarrow (Y_i'[ctr_i], W_i[ctr_i])$
  Eventually A halts and outputs a bit d
  `If` $b = d$ `then return 1 else return 0`

Fig. 1. *Distinguisher D_A in proof of Theorem 2, where R is the algorithm of Lemma 1.*

CRAMER-SHOUP. Now we consider another specific scheme, namely the practical public-key cryptosystem proposed by Cramer and Shoup, which is secure against chosen-ciphertext attack in the single-user setting as shown in [6]. We are interested in the security of this scheme (against chosen-ciphertext attack) in the multi-user setting. Let us define the basic scheme. Let G be a group of a large prime order q and let g be a generator of G. The prime q and the generator g comprise the global information I for the scheme. Let $\mathcal{H}$ be a family of collision-resistant hash functions, each member of which maps strings of arbitrary length to the elements of Z_q. The message space is the group G. The algorithms describing the scheme $\mathcal{CS} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ are defined as follows:

Algorithm $\mathcal{K}(q, g)$
  $g_1 \leftarrow g$; $g_2 \xleftarrow{R} G$
  $H \xleftarrow{R} \mathcal{H}$
  $x_1, x_2, y_1, y_2, z \xleftarrow{R} Z_q$
  $c \leftarrow g_1^{x_1} g_2^{x_2}$
  $d \leftarrow g_1^{y_1} g_2^{y_2}$
  $h \leftarrow g_1^{z}$
  $pk \leftarrow (g_1, g_2, c, d, h)$
  $sk \leftarrow (x_1, x_2, y_1, y_2, z)$
  `Return` (pk, sk)

Algorithm $\mathcal{E}_{pk}(M)$
  $r \xleftarrow{R} Z_q$
  $u_1 \leftarrow g_1^r$
  $u_2 \leftarrow g_2^r$
  $e \leftarrow h^r M$
  $\alpha \leftarrow H(u_1, u_2, e)$
  $v \leftarrow c^r d^{r\alpha}$
  $C \leftarrow (u_1, u_2, e, v)$
  `Return` C

Algorithm $\mathcal{D}_{sk}(C)$
  parse C as (u_1, u_2, e, v)
  $\alpha \leftarrow H(u_1, u_2, e)$
  `If` ${u_1}^{x_1+y_1\alpha} {u_2}^{x_2+y_2\alpha} = v$
  `then` $M \leftarrow e/{u_1}^{z}$
  `else reject`
  `Return` M

Although Cramer and Shoup do not explicitly state the concrete security of their reduction, it can be gleaned from the proof in [6, Section 4]. Their reduction is essentially tight. In our language:

$$\mathsf{Adv}^{\text{1-cca}}_{\mathcal{CS},(q,g)}(t, q_{\mathrm{d}}) \le 2 \cdot \mathsf{Adv}^{\mathrm{ddh}}_{q,g}(t) + 2 \cdot \mathsf{Adv}^{cr}_{\mathcal{H}}(t) + \frac{2(4q_{\mathrm{d}} + 1)}{q} \,. \tag{5}$$

as long as $q_{\rm d} \leq q/2$. The first term represents the advantage of the scheme in the single-user setting under chosen-ciphertext attack. Note that in this attack mode a new parameter is present, namely the number $q_{\rm d}$ of decryption queries made by the adversary, and hence the advantage is a function of this in addition to the time t. (Definition 1 has the details.) We are using $\mathsf{Adv}^{cr}_{\mathcal{H}}(t)$ to represent the maximum possible probability that an adversary with time t can find collisions in a random member H of the family $\mathcal{H}$. The last term in Equation (5) is negligible because q is much bigger than $q_{\rm d}$ in practice, which is why we view this reduction as tight. Moving to the multi-user setting, Theorem 1 in combination with the above tells us that

$$\mathsf{Adv}^{n\text{-cca}}_{\mathcal{CS},(q,g)}(t, q_{\rm e}, q_{\rm d}) \leq 2 \cdot q_{\rm e} n \cdot \mathsf{Adv}^{\rm ddh}_{q,g}(t') + 2 \cdot q_{\rm e} n \cdot \mathsf{Adv}^{cr}_{\mathcal{H}}(t') + \frac{2q_{\rm e} n \cdot (4q_{\rm d} + 1)}{q}$$

where $t' = t + (\log(q_{\rm e} n))$. The first term represents the advantage of the scheme in the multi-user setting under chosen-ciphertext attack, with n users, $q_{\rm e}$ encryption queries per user, and $q_{\rm d}$ decryption queries per user. Our improvement is the following.

Theorem 3. *Let G be a group of a large prime order q. Let $\mathcal{H}$ be a family of collision-resistant hash function, each member of which maps from $\{0,1\}^*$ into Z_q. Let g be a generator of G. Let $\mathcal{CS} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the Cramer-Shoup public-key encryption scheme associated to these parameters as defined above. Let $n, q_{\rm e}, q_{\rm d}, t$ be integers with $q_{\rm d} \leq q/2$. Then*

$$\mathsf{Adv}^{n\text{-cca}}_{\mathcal{CS},(q,g)}(t, q_{\rm e}, q_{\rm d}) \leq 2q_{\rm e} \cdot \mathsf{Adv}^{\rm ddh}_{q,g}(t') + 2q_{\rm e} \cdot \mathsf{Adv}^{cr}_{\mathcal{H}}(t') + \frac{2(4q_{\rm e} n q_{\rm d} + q_{\rm e} n)}{q}$$

where $t' = t + O(n \cdot T^{\rm exp}_q)$. ∎

Note that the last term is negligible for any reasonable values of $n, q_{\rm e}, q_{\rm d}$ due to the fact that q is large. So comparing with Equation (5) we see that we have essentially the same proven security for n users or one user when each encrypts $q_{\rm e}$ messages.

The reduction we got for Cramer-Shoup is not as tight as the one we got for El Gamal. We did not avoid the factor of $q_{\rm e}$ in a degradation of security of Cramer-Shoup for the multi-user setting. However it is still an open problem to avoid the factor of $q_{\rm e}$ even when there is only a single user encrypting $q_{\rm e}$ messages, so our result can be viewed as the optimal extension to the multi-user setting of the *known* results in the single-user setting.

To obtain this result we use Lemma 1 and modify the simulation algorithm from [6]. We provide a full proof and discuss the difficulties in improving the quality of the reduction in [2].

Acknowledgments

We thank Victor Shoup for information about the concrete security of the reduction in [6] and for pointing out to us the difficulties in attempting to improve the

quality of the Cramer-Shoup reduction (in the single-user setting) as a function of the number of encryption queries. We also thank the Eurocrypt 2000 referees for their comments.

Mihir Bellare and Alexandra Boldyreva are supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering.

References

1. O. Baudron, D. Pointcheval and J. Stern, "Extended notions of security for multicast public key cryptosystems," Manuscript.
2. M. Bellare, A. Boldyreva, and S. Micali, "Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements," Full version of this paper, available via `http://www-cse.ucsd.edu/users/mihir`.
3. M. Bellare, A. Desai, E. Jokipii and P. Rogaway, "A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation," Proceedings *of the* 38th *Symposium on Foundations of Computer Science*, IEEE, 1997.
4. M. Bellare, P. Rogaway, "Optimal asymmetric encryption – How to encrypt with RSA," *Advances in Cryptology – Eurocrypt* 94 *Proceedings*, Lecture Notes in Computer Science Vol. 950, A. De Santis ed., Springer-Verlag, 1994.
5. M. Bellare and A. Sahai, "Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization," *Advances in Cryptology – Crypto* 99 *Proceedings*, Lecture Notes in Computer Science Vol. 1666, M. Wiener ed., Springer-Verlag, 1999.
6. R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," *Advances in Cryptology – Crypto* 98 *Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.
7. D. Dolev, C. Dwork, and M. Naor, "Non-malleable cryptography," Proceedings *of the* 23rd *Annual Symposium on Theory of Computing*, ACM, 1991.
8. T. ElGamal, "A public key cryptosystem and signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol 31, 1985, pp. 469–472.
9. S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Science*, Vol. 28, 1984, pp. 270–299.
10. J. Håstad, "Solving simultaneous modular equations of low degree," *SIAM J. on Computing* Vol. 17, No. 2, April 1988.
11. M. Naor and O. Reingold, "Number-theoretic constructions of efficient pseudo-random functions," Proceedings *of the* 38th *Symposium on Foundations of Computer Science*, IEEE, 1997.
12. RSA Laboratories, "PKCS-1," `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/`.
13. C. Rackoff and D. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," *Advances in Cryptology – Crypto* 91 *Proceedings*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.
14. V. Shoup, "On formal models for secure key exchange, " Theory of Cryptography Library Record 99-12, `http://philby.ucsd.edu/cryptolib/`.

15. M. Stadler, "Publicly verifiable secret sharing," *Advances in Cryptology – Eurocrypt 96 Proceedings*, Lecture Notes in Computer Science Vol. 1070, U. Maurer ed., Springer-Verlag, 1996.
16. Y. Tsiounis and M. Yung, "On the security of El Gamal based encryption," *Proceedings of the First International workshop on practice and theory in Public Key Cryptography (PKC'98)*, Lecture Notes in Computer Science Vol. 1431, H. Imai and Y. Zheng eds., Springer-Verlag, 1998.

Using Hash Functions as a Hedge against Chosen Ciphertext Attack

Victor Shoup

IBM Zurich Research Lab, Säumerstr. 4, 8803 Rüschlikon, Switzerland
sho@zurich.ibm.com

Abstract. The cryptosystem recently proposed by Cramer and Shoup [CS98] is a practical public key cryptosystem that is secure against adaptive chosen ciphertext attack provided the Decisional Diffie-Hellman assumption is true. Although this is a reasonable intractability assumption, it would be preferable to base a security proof on a weaker assumption, such as the Computational Diffie-Hellman assumption. Indeed, this cryptosystem in its most basic form is in fact *insecure* if the Decisional Diffie-Hellman assumption is false. In this paper we present a practical hybrid scheme that is just as efficient as the scheme of of Cramer and Shoup; indeed, the scheme is slightly more efficient than the one originally presented by Cramer and Shoup; we *prove* that the scheme is secure if the Decisional Diffie-Hellman assumption is true; we *give strong evidence* that the scheme is secure if the weaker, Computational Diffie-Hellman assumption is true by providing a proof of security in the *random oracle* model.

1 Introduction

It is largely agreed upon in the cryptographic research community that the "right" definition of security for a public key cryptosystem is security against adaptive chosen ciphertext attack, as defined by Rackoff and Simon [RS91] and Dolev, Dwork, and Naor [DDN91]. At least, this is the definition of security that allows the cryptosystem to be deployed safely in the widest range of applications.

Dolev, Dwork, and Naor [DDN91] presented a cryptosystem that could be proven secure in this sense using a reasonable intractability assumption. However, their scheme was quite impractical. Subsequently, Bellare and Rogaway [BR93,BR94] presented very practical schemes, and analyzed their security under the standard RSA assumption; more precisely, they proved the security of these schemes in the *random oracle* model, wherein a cryptographic hash function is treated *as if* it were a "black box" containing a random function. However, the security of these schemes in the "real world" (i.e., the standard model of computation) has never been proved.

A proof of security in the random oracle model provides strong evidence that breaking the scheme without breaking the underlying intractability assumptions will be quite difficult to do, although it does not rule out this possibility altogether. The advantage of a proof of security in the "real world" is that it does not

just provide such strong evidence, it *proves* that the scheme cannot be broken without breaking the underlying intractability assumptions.

Recently, Cramer and Shoup [CS98] presented a practical cryptosystem and proved its security in the standard model, based on the Decisional Diffie-Hellman (DDH) assumption. It is hard to compare the security of this scheme with that of the schemes of Bellare and Rogaway—although the former scheme can be analyzed in the "real world," and the latter schemes only in the random oracle model, the underlying intractability assumptions are incomparable. Indeed, a proof of security is worthless if the underlying assumptions turn out to be false, and in fact, both the Cramer-Shoup scheme (in its basic form) and the Bellare-Rogaway schemes can be broken if their respective assumptions are false.

Perhaps the strongest criticism against the Cramer-Shoup scheme is that the assumption is too strong; in particular, it has not been studied as extensively as other assumptions, including the RSA assumption.

In this paper, we address this criticism by presenting a hybrid variation of the Cramer-Shoup scheme. This scheme is actually somewhat simpler and more efficient than the original, and a proof of security in the "real world" can also be made based on the DDH assumption. However, the same scheme can also be proved secure in the random oracle model based on the Computational Diffie-Hellman (CDH) assumption. This assumption was introduced by Diffie and Hellman [DH76] in their work that opened the field of public key cryptography, and has been studied at least as intensively as any other intractability assumption used in modern cryptography. Thus, in comparison to other available practical encryption schemes, the scheme discussed here is arguably no less secure, while still admitting a proof of security in the "real world" under a reasonable, if somewhat strong, intractability assumption.

We believe this "hedging with hash" approach may be an attractive design paradigm. The general form of this approach would be to design practical cryptographic schemes whose security can be proved in the "real world" based on a reasonable, if somewhat strong, intractability assumption, but whose security can also be proved in the random oracle model under a weaker intractability assumption. This same "hedging with hash" security approach has also been applied to digital signature schemes: Cramer and Shoup [CS99] presented and analyzed a practical signature scheme that is secure in the "real world" under the so-called Strong RSA assumption, but is also secure in the random oracle model under the ordinary RSA assumption. Although that paper and this paper both advocate this "hedging with hash" security approach, the technical details and proof techniques are quite unrelated. In the context of encryption or signatures, one can also "hedge" just by combining two schemes based on different intractability assumptions (via composition for encryption and via concatenation for signatures). However, this type of hedging is much more expensive computationally, and much less elegant than the type of hedging we are advocating here.

Other Diffie-Hellman based encryption schemes. [TY98] present a scheme, but it cannot be proved secure against adaptive chosen ciphertext attack under any

intractability assumption, even in the random oracle model. There is indeed a security analysis in [TY98], but rather than basing the proof of security on the hardness of a specific problem, it is based on the assumption that the adversary behaves in a specific way, similar to as was done in [ZS92]. [SG98] present two schemes; the first can be proved secure against adaptive chosen ciphertext attack in the random oracle model under the CDH, while the proof of security for the second relies on the DDH. Both schemes are amenable to distributed decryption. Moreover, the techniques in the current paper can be applied to the second scheme to weaken the intractability assumption, replacing the DDH with the CDH (but not the distributed version). [SG98] also discusses an encryption scheme that is essentially the same as that in [TY98], and argues why it would be quite difficult using known techniques to prove that such is scheme is secure against adaptive chosen ciphertext attack even in the random oracle model. [ABR98] present a scheme for which security against adaptive chosen ciphertext attack can only be proved under non-standard assumptions—these assumptions relate to the hardness of certain "interactive" problems, and as such they do not qualify as "intractability assumptions" in the usual sense of the term. Furthermore, using random oracles does not seem to help. [FO99] present a scheme that can be proven secure against adaptive chosen ciphertext attack under the CDH assumption in the random oracle model. Moreover, they present a fairly general method of converting any public-key encryption scheme that is semantically secure into one that can be proved secure against adaptive chosen ciphertext attack in the random oracle model. However, nothing at all can be said about the security of this scheme in the "real world."

2 Security against Adaptive Chosen Ciphertext Attack

We recall the definition of security against adaptive chosen ciphertext attack.

We begin by describing the attack scenario.

First, the key generation algorithm is run, generating the public key and private key for the cryptosystem. The adversary, of course, obtains the public key, but not the private key.

Second, the adversary makes a series of arbitrary queries to a *decryption oracle*. Each query is a ciphertext ψ that is decrypted by the decryption oracle, making use of the private key of the cryptosystem. The resulting decryption is given to the adversary. The adversary is free to construct the ciphertexts in an arbitrary way—it is certainly *not* required to compute them using the encryption algorithm.

Third, the adversary prepares two messages m_0, m_1, and gives these two an *encryption oracle*. The encryption oracle chooses $b \in \{0,1\}$ at random, encrypts m_b, and gives the resulting "target" ciphertext ψ' to the adversary. The adversary is free to choose m_0 and m_1 in an arbitrary way, except that if message lengths are not fixed by the cryptosystem, then these two messages must nevertheless be of the same length.

Fourth, the adversary continues to submit ciphertexts ψ to the decryption oracle, subject only to the restriction that $\psi \neq \psi'$.

Just before the adversary terminates, it outputs $b' \in \{0,1\}$, representing its "guess" of b.

That completes the description of the attack scenario.

The adversary's *advantage* in this attack scenario is defined to be the distance from $1/2$ of the probability that $b' = b$.

A cryptosystem is defined to be *secure against adaptive chosen ciphertext attack* if for any efficient adversary, its advantage is negligible.

Of course, this is a complexity-theoretic definition, and the above description suppresses many details, e.g., there is an implicit security parameter which tends to infinity, and the terms "efficient" and "negligible" are technical terms, defined in the usual way. One can work in a *uniform* (i.e., Turing machines) or a *non-uniform* model (i.e., circuits) of computation. This distinction will not affect any results in this paper.

3 Intractability Assumptions

In this section, we discuss the intractability assumptions used in this paper.

Let G be a group of large prime order q.

The Discrete Logarithm (DL) problem is this: given $g \in G$ with $g \neq 1$ and g^x, compute x (modulo q).

The Computational Diffie-Hellman (CDH) problem is this: given $g \in G$ with $g \neq 1$, along with g^x and g^y, compute g^{xy}. A "good" algorithm for this problem is an efficient, probabilistic algorithm such that for all inputs, its output is correct with all but negligible probability. The CDH assumption is the assumption that no such "good" algorithm exists. Using well-known random-self reductions, along with the results of [MW96] or [Sho97], the existence of such a "good" algorithm is equivalent to the existence of a probabilistic algorithm that outputs a correct answer with non-negligible probability, where the probability is taken over the coin flips of the algorithm, as well as a random choice of $g \in G$, and $x, y \in \mathbf{Z}_q$.

The Decisional Diffie-Hellman (DDH) problem is this: given $g \in G$ with $g \neq 1$, along with g^x, g^y, and g^z decide if $z \equiv xy \bmod q$. A "good" algorithm is an efficient, probabilistic algorithm such that for all inputs, its output is correct with all but negligible probability. The DDH assumption is the assumption that no such "good" algorithm exists. Using the random-self reduction presented by Stadler [Sta96], the existence of such a "good" algorithm is equivalent to the existence of a probabilistic statistical test distinguishing the distributions (g, g^x, g^y, g^z) and (g, g^x, g^y, g^{xy}), where $g \in G$, and $x, y, z \in \mathbf{Z}_q$ are randomly chosen.

All of these problems are equally hard in a "generic" model of computation, where an algorithm is not allowed to exploit the representation of the group G [Sho97]; in this model, $O(\sqrt{q})$ group operations are both necessary and sufficient. However, for specific groups, special methods, such as "index calculus" methods, may apply, allowing for more efficient algorithms.

In general, the only known way to solve either the CDH or DDH problems is to first solve the DL problem. However, there remains the possibility that the DL problem is hard and the CDH problem is easy, or that the CDH problem is hard, and the DDH problem is easy. Maurer [Mau94] has shown that under certain circumstances, an algorithm for solving the CDH problem can be used to solve the DL problem. This reduction is a "generic" reduction that does not depend on the representation of the group G. It can also be shown that there is no such generic reduction allowing one to efficiently solve the CDH or DL problems using an algorithm for the DDH problem. This fact could be considered as evidence supporting the claim that the DDH assumption is possibly stronger than the CDH assumption.

It is perhaps worth stressing that although the DDH may be a stronger assumption than either the DL or CDH assumption, these latter two "usual" assumptions have rarely, if ever, been used to prove the security of a practical cryptographic scheme of any kind—except in the random oracle model. Indeed, it appears to be a widely held misconception that the security of the Diffie-Hellman key exchange protocol [DH76] and variants thereof (e.g., [DvOW92]) is implied by the CDH assumption. This is simply not the case—under *any* reasonable definition of security, except in the random oracle model. One can use the DDH assumption, however, as the basis for proving the security of such schemes (see, e.g., [BCK98,Sho99]).

The DDH assumption appears to have first surfaced in the cryptographic literature in [Bra93]. For other applications and discussion of the DDH, see [Bon98,NR97].

As in the previous section, we have suppressed many details in the above discussion, e.g., there is an implicit security parameter that tends to infinity, and for each value of the security parameter, there is an implicit probability distribution of groups.

4 The Encryption Scheme

4.1 The basic Cramer-Shoup scheme

We recall the basic Cramer-Shoup cryptosystem, as presented in [CS98]. The cryptosystem works with a group G of large prime order q.

Key Generation. The key generation algorithm runs as follows. Random elements $g_1, g_2 \in G \backslash \{1\}$ are chosen, and random elements

$$x_1, x_2, y_1, y_2, z \in \mathbf{Z}_q$$

are also chosen. Next, the group elements

$$c = g_1^{x_1} g_2^{x_2}, \ d = g_1^{y_1} g_2^{y_2}, \ h = g_1^z$$

are computed. Finally, a random key k indexing a universal one-way hash function UOWH is chosen. We assume that the output of the hash function is an element of $\mathbf{Z}_q$. The public key is (g_1, g_2, c, d, h, k), and the private key is (x_1, x_2, y_1, y_2, z).

Encryption. To encrypt, we assume a message m can be encoded as an element of G. The encryption algorithm runs as follows. First, it chooses $r \in \mathbf{Z}_q$ at random. Then it computes

$$u_1 = g_1^r,\ u_2 = g_2^r,\ e = h^r m,\ \alpha = \mathsf{UOWH}(k; u_1, u_2, e),\ v = c^r d^{r\alpha}.$$

The ciphertext is

$$(u_1, u_2, e, v).$$

Decryption. Given a ciphertext (u_1, u_2, e, v), the decryption algorithm runs as follows. It first computes $\alpha = \mathsf{UOWH}(k; u_1, u_2, e)$, and tests if

$$u_1^{x_1 + y_1\alpha} u_2^{x_2 + y_2\alpha} = v.$$

If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs

$$m = e/u_1^z.$$

In [CS98], it was shown that this scheme is secure against adaptive chosen ciphertext attack, under the DDH assumption for G, and assuming UOWH is a secure universal one-way hash function. Although there are theoretical constructions for UOWH [NY89], a reasonable construction would be to use the compression function of SHA-1, in conjunction with the constructions in [BR97] or [Sho00]. With this approach, the security of UOWH can be based on the assumption that the SHA-1 compression function is second-preimage collision resistant, a potentially much weaker assumption than full collision resistance.

4.2 A general hybrid construction

We describe here a general method for constructing a hybrid encryption scheme. To this end, it is convenient to define the notion of a *key encapsulation scheme*. This is a scheme that allows a party to generate a random bit string and send it to another party, encrypted under the receiving party's public key.

A key encapsulation scheme works just like a public key encryption scheme, except that the encryption algorithm takes no input other than the recipient's public key. Instead, the encryption algorithm generates a pair (K, ψ), where K is a random bit string of some specified length, say l, and ψ is an encryption of K, that is, the decryption algorithm applied to ψ yields K.

One can always use a public key encryption scheme for this purpose, generating a random bit string, and then encrypting it under the recipient's public key. However, as we shall see, one can construct a key encapsulation scheme in other ways as well.

One can easily adapt the notion of security against adaptive chosen ciphertext attack to a key encapsulation scheme. The only difference in the attack scenario is the behavior of the encryption oracle. The adversary does not give two messages to the encryption oracle. Rather, the encryption oracle runs the key encapsulation algorithm to obtain a pair (K', ψ'). The encryption oracle

then gives the adversary either (K', ψ') or (K'', ψ'), where K'' is an independent random l-bit string; the choice of K' versus K'' depends on the value of the random bit b chosen by the encryption oracle.

Using a key encapsulation scheme that is secure against adaptive chosen ciphertext attack, we can construct a hybrid public key cryptosystem that is secure against adaptive chosen ciphertext attack as follows.

We need a pseudo-random bit generator PRBG. There are theoretical constructions for such a generator, but a perfectly reasonable approach is to construct the generator using a standard block cipher, such as DES, basing its security on a reasonable pseudo-randomness assumption on the underlying block cipher. We assume that PRBG stretches l-bit strings to strings of arbitrary length. We assume here that $1/2^l$ is a negligible quantity.

We need a hash function AXUH suitable for message authentication, i.e., an almost XOR-universal hash function [Kra94]. We assume that AXUH is keyed by an l'-bit string and hashes arbitrary bit strings to l-bit strings. Many efficient constructions for AXUH exist that do not require any intractability assumptions.

To encrypt a message m, we run the key encapsulation scheme to obtain a random string K along with its encryption ψ. Next, we apply PRBG to K to obtain an l'-bit string K_1, an l-bit string K_2, and an $|m|$-bit string f. Finally, we compute

$$e = f \oplus m, \ a = \mathsf{AXUH}(K_1; e) \oplus K_2.$$

The ciphertext is

$$(\psi, e, a).$$

To decrypt (ψ, e, a), we first decrypt ψ to obtain K. Note that decrypting ψ may result in a "reject," in which case we "reject" as well. Otherwise, we apply PRBG to K to obtain an l'-bit string K_1, an l-bit string K_2, and an $|e|$-bit string f. We then test if $a = \mathsf{AXUH}(K_1; e) \oplus K_2$. If this condition does not hold, we "reject." Otherwise, we output $m = e \oplus f$.

Theorem 1. *If the underlying key encapsulation scheme is secure against adaptive chosen ciphertext attack, and* PRBG *is a secure pseudo-random bit generator, then the above hybrid scheme is also secure against adaptive chosen ciphertext attack.*

This appears to be somewhat of a "folk theorem." The proof is straightforward, and is left as an easy exercise for the reader.

4.3 A hybrid Cramer-Shoup scheme

We now describe a key encapsulation scheme based on the Cramer-Shoup encryption scheme. Combined with the general hybrid construction in §4.2, this yields a hybrid encryption scheme. As a hybrid scheme, it is much more flexible than the "basic" version of the scheme described in §4.1, as messages may be arbitrary bit strings and do not need to be encoded as group elements. This flexibility allows one greater freedom in choosing the group G, which can be exploited to obtain a much more efficient implementation as well. Also, the scheme

we describe incorporates some modifications that lead to a simpler and more efficient decryption algorithm.

We need a pair-wise independent hash function PIH. We assume that PIH takes a key μ and maps elements $\lambda \in G$ to l-bit strings. Many efficient constructions for PIH exist that do not require any intractability assumptions. We will want to apply the Entropy Smoothing Theorem (see [Lub96, Ch. 8] or [IZ89]) to PIH, assuming that the input λ is a random group element. To do this effectively, the relative sizes of q and l must be chosen appropriately, so that $\sqrt{2^l/q}$ is a negligible quantity.

We also need a "magic" hash function MH mapping elements of $G \times G$ to l-bit strings. This function is not required to satisfy any particular security requirements. A construction using a cryptographic hash like MD5 or SHA-1 is recommended (see [BR93]). This function will only play a role when we analyze the scheme in the random oracle model, where MH will be modeled as a random oracle.

Now we are ready to describe the key encapsulation scheme.

Key Generation. A random element $g_1 \in G \backslash \{1\}$ is chosen, together with $w \in \mathbf{Z}_q \backslash \{0\}$ and $x, y, z \in \mathbf{Z}_q$. Next, the following group elements are computed:

$$g_2 = g_1^w,\ c = g_1^x,\ d = g_1^y,\ h = g_1^z.$$

Finally, a random key k indexing a universal one-way hash function UOWH is chosen, as well as a random key μ for PIH; the public key is $(g_1, g_2, c, d, h, k, \mu)$.

Key Encapsulation. The key encapsulation scheme runs as follows. First, it chooses $r \in \mathbf{Z}_q$ at random. Then it computes

$$u_1 = g_1^r,\ u_2 = g_2^r, \alpha = \mathsf{UOWH}(k; u_1, u_2),\ v = c^r d^{r\alpha},\ \lambda = h^r.$$

Finally, it computes

$$K = \mathsf{PIH}(\mu; \lambda) \oplus \mathsf{MH}(u_1, \lambda).$$

The ciphertext is

$$(u_1, u_2, v),$$

which is an encryption of the key K.

Decryption. Given a ciphertext (u_1, u_2, v), the decryption algorithm runs as follows. It first computes $\alpha = \mathsf{UOWH}(k; u_1, u_2)$, and tests if

$$u_2 = u_1^w \quad \text{and} \quad v = u_1^{x+\alpha y}.$$

If this condition does not hold, the decryption algorithm outputs "reject" and halts. Otherwise, it computes $\lambda = u_1^z$, outputs the key $K = \mathsf{PIH}(\mu; \lambda) \oplus \mathsf{MH}(u_1, \lambda)$.

Theorem 2. *The above key encapsulation scheme is secure against adaptive chosen ciphertext attack, under the DDH assumption for G, and also assuming that* UOWH *is a secure universal one-way hash function.*

We only briefly sketch the proof, as it differs only slightly from the proof of the main theorem in [CS98]. The structure of the proof is as follows. We make a sequence of transformations to the attack game. In each of these transformations, we argue that we affect the adversary's advantage by a negligible amount, and in the final transformed game, the adversary's advantage is zero. The original game is denoted $\mathbf{G}_0$, and the transformed games are denoted $\mathbf{G}_i$, for $i = 1, 2, \ldots$.

First, some notation. Let $\psi' = (u_1', u_2', v')$ be the "target" ciphertext. For notational convenience and clarity, the internal variables used by the encryption algorithm in generating the target ciphertext will also be referred to in "primed" form, e.g., the value of α for the target ciphertext is denoted α'. Also, we will call a ciphertext (u_1, u_2, v) *valid* if $\log_{g_1} u_1 = \log_{g_2} u_2$; otherwise, it is called *invalid*.

In game $\mathbf{G}_1$, we change the key generation algorithm as follows. It chooses random $g_1, g_2 \in G \backslash \{1\}$ at random, along with

$$x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbf{Z}_q.$$

Next, it computes the following group elements:

$$c = g_1^{x_1} g_2^{x_2}, \ d = g_1^{y_1} g_2^{y_2}, \ h = g_1^{z_1} g_2^{z_2}.$$

It chooses the keys k and μ as before, and the public key is $(g_1, g_2, c, d, h, k, \mu)$. We also modify the decryption oracle as follows. Given a ciphertext (u_1, u_2, v), it computes $\alpha = \mathsf{UOWH}(k; u_1, u_2)$, and tests if

$$v = u_1^{x_1 + y_1 \alpha} u_2^{x_2 + y_2 \alpha}.$$

If this condition does not hold, the decryption oracle outputs "reject." Otherwise, it computes $\lambda = u_1^{z_1} u_2^{z_2}$, and outputs the key $K = \mathsf{PIH}(\mu; \lambda) \oplus \mathsf{MH}(u_1, \lambda)$.

We now claim that the adversary's advantage in game $\mathbf{G}_1$ differs from its advantage in game $\mathbf{G}_0$ by a negligible amount. The argument is runs along the same lines as that of the proof of Lemma 1 in [CS98]. That is, these two games are equivalent up to the point where an invalid ciphertext is not rejected; however, the probability that this happens is negligible.

In game $\mathbf{G}_2$, we modify the encryption oracle, simply choosing $u_1', u_2' \in G$ at random, setting

$$v' = (u_1')^{x_1 + y_1 \alpha'} (u_2')^{x_2 + y_2 \alpha'},$$

and computing the rest of the target ciphertext as usual.

It is clear that under the DDH assumption, the adversary's advantage in game $\mathbf{G}_2$ differs from its advantage in game $\mathbf{G}_1$ by a negligible amount.

In game $\mathbf{G}_3$, we move the computation of the target ciphertext $\psi' = (u_1', u_2', v')$ to the very beginning of the game, and if the adversary ever submits a ciphertext $\psi = (u_1, u_2, v)$ to the decryption oracle with $(u_1, u_2) \neq (u_1', u_2')$, but with $\alpha = \alpha'$, we simply halt the game.

It is clear that under the security assumption for UOWH, adversary's advantage in game $\mathbf{G}_3$ differs from its advantage in game $\mathbf{G}_2$ by a negligible amount.

In game $\mathbf{G}_4$, we modify the encryption oracle yet again, choosing λ' as a random group element.

That this only has a negligible impact on the adversary's advantage follows from the line of reasoning in the proof of Lemma 2 in [CS98]. That is, these two games are equivalent up to the point where an invalid ciphertext is not rejected (provided $\log_{g_1} u_1' \neq \log_{g_2} u_2'$); however, the probability that this happens is negligible (this makes use of the fact that no collisions in UOWH are found).

In game $\mathbf{G}_5$, we modify the encryption oracle again, this time choosing K' as a random l-bit string.

This modification only has a negligible impact on the adversary's advantage. Indeed, since PIH is a pair-wise independent hash function, so is the function

$$\lambda \mapsto \mathsf{PIH}(\mu; \lambda) \oplus \mathsf{MH}(u_1', \lambda),$$

where we view (μ, u_1') as the key to this hash function. By the Entropy Smoothing Theorem, the value $K' = \mathsf{PIH}(\mu; \lambda') \oplus \mathsf{MH}(u_1', \lambda')$ is statistically indistinguishable from a random l-bit string.

It is clear that in game $\mathbf{G}_5$, the adversary's advantage is zero. That completes the proof of the theorem.

Theorem 3. *Modeling* MH *as a random oracle, the above key encapsulation scheme is secure against adaptive chosen ciphertext attack, under the CDH assumption for* G*, and also assuming that* UOWH *is a secure universal one-way hash function.*

To prove this theorem, suppose there is an adversary that has a non-negligible advantage in the attack game. Now, Theorem 2 remains valid, even if we replace MH by a random oracle. So assuming the security properties of UOWH, the existence of an efficient adversary with non-negligible advantage implies the existence of an efficient algorithm for solving the DDH problem in G. In fact, the proof of Theorem 2 shows how to construct such an algorithm using the adversary as a subroutine; though technically "efficient," this may not be the most practical algorithm for solving the DDH problem in G; a more practical algorithm would certainly make the simulator we describe below more efficient.

In any case, we assume we have an efficient algorithm solving the DDH problem. To be precise, define the function $\mathsf{DHP}(g, g^x, g^y, g^z)$ to be 1 if $g \neq 1$ and $z \equiv xy \bmod q$, and 0 otherwise. Then our assumption is that there is an efficient probabilistic algorithm that on all inputs computes DHP correctly, with negligible error probability.

Now we show how to use such an algorithm for DHP, together with an adversary that has non-negligible advantage in the attack game, to construct an efficient algorithm for solving the CDH problem. We assume that the instance of the CDH problem consists of randomly chosen group elements $g_1, u_1', h \in G$ (with $g_1 \neq 1$), and our goal is to compute $\lambda' \in G$ such that $\mathsf{DHP}(g_1, u_1', h, \lambda') = 1$.

We describe a simulator that simulates the adversary's view in the attack game. The input to the simulator is $g_1, u_1', h \in G$ as above.

The simulator constructs a public key for the cryptosystem as follows. It chooses $w \in \mathbf{Z}_q \backslash \{1\}$ at random and sets $g_2 = g_1^w$. It chooses $x, y \in \mathbf{Z}_q$ at random, and computes $c = g_1^x, d = g_1^y \in G$, It also generates a random key k for UOWH and a random key μ for PIH. The public key is $(g_1, g_2, c, d, h, k, \mu)$.

The simulator is in complete control of the random oracle representing MH. We maintain a set S (initially empty) of tuples

$$(u_1, \lambda, \nu) \in G \times G \times \{0,1\}^l,$$

representing the portion of MH that has been defined so far. That is, $\mathsf{MH}(u_1, \lambda)$ is defined to be ν if and only if $(u_1, \lambda, \nu) \in S$. We also maintain the subset $S_{DDH} \subset S$ of tuples (u_1, λ, ν) satisfying the additional constraint that $\mathsf{DHP}(g_1, u_1, h, \lambda) = 1$. We also maintain a set S' (initially empty) of pairs

$$(u_1, K) \in G \times \{0,1\}^l.$$

To process a request to evaluate the random oracle at a point $(u_1, \lambda) \in G \times G$, the simulator executes the algorithm shown in Figure 1.

```
if (u_1, λ, ν) ∈ S for some ν ∈ {0,1}^l
      return ν
else if DHP(g_1, u_1, h, λ) = 0 {
      ν ←_R {0,1}^l
      S ← S ∪ {(u_1, λ, ν)}
      return ν
}
else if u_1 = u_1'
      output the solution λ = λ' to the CDH problem and halt
else {
      if (u_1, K) ∈ S' for some K ∈ {0,1}^l
            ν ← K ⊕ PIH(μ; λ)
      else
            ν ←_R {0,1}^l
      S ← S ∪ {(u_1, λ, ν)}
      S_DDH ← S_DDH ∪ {(u_1, λ, ν)}
      return ν
}
```

Fig. 1. Simulator's algorithm to evaluate random oracle at (u_1, λ).

The manner in which pairs are added to S' is described below, in the description of the simulation of the decryption oracle.

We next describe how the simulator deals with the encryption oracle. It computes $u_2' = (u_1')^w$, and computes $v' = (u_1')^{x + \alpha' y}$. It outputs a random l-bit string K' and the "target" ciphertext $\psi' = (u_1', u_2', v')$. Note that the output of the encryption oracle is independent of the random bit b.

Now we describe how the simulator deals with the decryption oracle. The algorithm used to process a request to decrypt a ciphertext $\psi = (u_1, u_2, v) \neq \psi'$ is shown in Figure 2.

$\alpha \leftarrow \mathsf{UOWH}(k; u_1, u_2)$
if $u_2 \neq u_1^w$ or $v \neq u_1^{x+\alpha y}$
    return "reject"
else if $(u_1, K) \in S'$ for some $K \in \{0,1\}^l$
    return K
else if $(u_1, \lambda, \nu) \in S_{DDH}$ for some $\lambda \in G$ and $\nu \in \{0,1\}^l$
    return $\nu \oplus \mathsf{PIH}(\mu; \lambda)$
else {
    $K \leftarrow_R \{0,1\}^l$
    $S' \leftarrow S' \cup \{(u_1, K)\}$
    return K
}

Fig. 2. Simulator's algorithm to decrypt $\psi = (u_1, u_2, v)$.

That completes the description of the simulator. It is easy to verify that the actual attack and the attack played against this simulator are equivalent, at least up to the point where the adversary queries the random oracle at the point (u_1', λ'). But up to that point, the hidden bit b is independent of the adversary's view. Therefore, since we are assuming the adversary does have a non-negligible advantage, the adversary must query the random oracle at the point (u_1', λ') with non-negligible probability.

That completes the proof of Theorem 3.

Remarks

Remark 1. The decryption algorithm tests if $u_2 = u_1^w$ and $v = u_1^{x+\alpha y}$. In the proof of Theorem 2, we show that we can replace this test with a different test that is equivalent from the point of view of the data the adversary sees; however, these tests may not be equivalent from the point of view of *timing information*. In particular, if the decryption algorithm returns "reject" immediately after finding that $u_2 \neq u_1^w$, this could perhaps leak timing information to the adversary that is not available in game $\mathbf{G}_1$ in the proof. We therefore recommend that *both* the tests $u_2 = u_1^w$ and $v = u_1^{x+\alpha y}$ are performed, even if the one of them fails.

Remark 2. In a typical implementation, the group G may be a subgroup of $\mathbf{Z}_p^*$ for a prime p, perhaps where p is much larger than q. In this case, after testing if the encodings of u_1, u_2, v properly represent elements of $\mathbf{Z}_p^*$, the decryption algorithm must check that $u_1^q = 1$, so as to ensure that $u_1 \in G$. We need not make any further tests to check that $u_2, v \in G$, since this is already implied by the tests $u_2 = u_1^w$ and $v = u_1^{x+\alpha y}$.

Remark 3. The decryption algorithm must compute either three or four exponentiations, all with respect to the same base u_1. An implementation can and should exploit this to get a significantly more efficient decryption algorithm by using precomputation techniques (see, e.g. [LL94]).

Remark 4. The reduction given in the proof of Theorem 3 is perhaps not as efficient as one would like. If T is the time required to solve the DDH problem, and Q queries are made to the random oracle, then the running time of the simulator will be essentially that of the adversary plus $O(T \cdot Q)$. Also, note that the inclusion of u_1 as an argument to MH is not essential to get a polynomial-time security reduction; however, if we dropped u_1 as an argument to MH, the only simulator we know how to construct has a term of $O(Q \cdot Q' \cdot T)$ in its running time, where Q' is the number of decryption oracle queries.

Remark 5. In the proof of Theorem 3, we argued that if there is an adversary with a non-negligible advantage in the attack game, then there is an efficient algorithm for solving the DDH. This perhaps deserves some elaboration. For such an adversary A, there exists a polynomial $P(\gamma)$ in the security parameter γ, and an infinite set Γ of choices of the security parameter, such that for all $\gamma \in \Gamma$, the advantage of A is at least $1/P(\gamma)$. We are assuming that the group G is generated by a probabilistic function $\mathcal{G}(\gamma)$ that takes the security parameter γ as input. For an algorithm A', a security parameter γ, and $0 \leq \epsilon \leq 1$, define $V(A', \gamma, \epsilon)$ be the set of outputs G of $\mathcal{G}(\gamma)$ such that A' computes DHP on G with error probability at most ϵ. As in the previous remark, let Q be (an upper bound on) the number or random oracle queries made by A. Then the existence of A, together with Theorem 2, implies that there exists an efficient algorithm A' and a polynomial $P'(\gamma)$, such that for all $\gamma \in \Gamma$, $\Pr[\mathcal{G}(\gamma) \in V(A', \gamma, 1/(2Q))] \geq 1/P'(\gamma)$. The reduction described in the proof of Theorem 3 only works when $\gamma \in \Gamma$ and $\mathcal{G}(\gamma) \in V(A', \gamma, 1/(2Q))$, but this is enough to contradict the CDH assumption.

References

[ABR98] M. Abdalla, M. Bellare, and P. Rogaway. DHAES: an encryption scheme based on the Diffie-Hellma problem. Submission to IEEE P1363, 1998.

[BCK98] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *30th Annual ACM Symposium on Theory of Computing*, 1998.

[Bon98] D. Boneh. The Decision Diffie-Hellman Problem. In *Ants-III*, pages 48–63, 1998. Springer LNCS 1423.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology—Crypto '94*, pages 92–111, 1994.

[BR97] M. Bellare and P. Rogaway. Collision-resistant hashing: towards making UOWHFs practical. In *Advances in Cryptology–Crypto '97*, 1997.

[Bra93] S. Brands. An efficient off-line electronic cash system based on the representation problem, 1993. CWI Technical Report, CS-R9323.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology–Crypto '98*, pages 13–25, 1998.

[CS99] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *6th ACM Conf. on Computer and Communications Security*, 1999.

[DDN91] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, 1991.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, 22:644–654, 1976.

[DvOW92] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and authenticated key exchange. *Designs, Code, and Cryptography*, 2:107–125, 1992.

[FO99] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology–Crypto '99*, pages 537–554, 1999.

[IZ89] R. Impagliazzo and D. Zuckermann. How to recycle random bits. In *30th Annual Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[Kra94] H. Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology—Crypto '94*, pages 129–139, 1994.

[LL94] C. H. Lim and P. J. Lee. More flexible exponentiation with precomputation. In *Advances in Cryptology–Crypto '94*, pages 95–107, 1994.

[Lub96] M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton University Press, 1996.

[Mau94] U. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In *Advances in Cryptology–Crypto '94*, pages 271–281, 1994.

[MW96] U. Maurer and S. Wolf. Diffie-Hellman oracles. In *Advances in Cryptology–Crypto '96*, pages 268–282, 1996.

[NR97] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, 1997.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, 1989.

[RS91] C. Rackoff and D. Simon. Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology–Crypto '91*, pages 433–444, 1991.

[SG98] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology–Eurocrypt '98*, 1998.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology–Eurocrypt '97*, 1997.

[Sho99] V. Shoup. On formal models for secure key exchange. IBM Research Report RZ 3120, April 1999.

[Sho00] V. Shoup. A composition theorem for universal one-way hash functions. In *Advances in Cryptology–Eurocrypt 2000,* pages 451–458, 2000.

[Sta96] M. Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology–Eurocrypt '96*, pages 190–199, 1996.

[TY98] Y. Tsiounis and M. Yung. On the security of ElGamal based encryption. In *PKC '98*, 1998.

[ZS92] Y. Zheng and J. Seberry. Practical approaches to attaining security against adaptively chosen ciphertext attacks. In *Advances in Cryptology–Crypto '92*, pages 292–304, 1992.

Security Aspects of Practical Quantum Cryptography

Gilles Brassard[1,*], Norbert Lütkenhaus[2,**], Tal Mor[3,4,***], and Barry C. Sanders[5]

[1] Département IRO, Université de Montréal, C.P. 6128, succ. centre–ville, Montréal (Québec), Canada H3C 3J7
brassard@iro.umontreal.ca

[2] Helsinki Institute of Physics, P.O.Box 9, 00014 Helsingin yliopisto, Finland
lutkenha@rock.helsinki.fi

[3] Electrical Engineering, College of Judea and Samaria, Ariel, Israel
talmo@cs.technion.ac.il

[4] Electrical Engineering, University of California at Los Angeles, Los Angeles, CA 90095–1594, USA
talmo@ee.ucla.edu

[5] Department of Physics, Macquarie University, Sydney, New South Wales 2109, Australia
barry.sanders@mq.edu.au

Abstract. The use of quantum bits (qubits) in cryptography holds the promise of secure cryptographic quantum key distribution schemes. Unfortunately, the implemented schemes are often operated in a regime which excludes unconditional security. We provide a thorough investigation of security issues for practical quantum key distribution, taking into account channel losses, a realistic detection process, and modifications of the "qubits" sent from the sender to the receiver. We first show that even quantum key distribution with perfect qubits might not be achievable over long distances when fixed channel losses and fixed dark count errors are taken into account. Then we show that existing experimental schemes (based on weak pulses) currently do not offer unconditional security for the reported distances and signal strength. Finally we show that parametric downconversion offers enhanced performance compared to its weak coherent pulse counterpart.

1 Introduction

Quantum information theory suggests the possibility of accomplishing tasks that are beyond the capability of classical computer science, such as information-theoretically secure cryptographic key distribution [3, 5]. The lack of security

* Supported in part by Canada's NSERC and Québec's FCAR.
** Supported by grant 43336 by the Academy of Finland.
*** Supported in part by grant #961360 from the Jet Propulsion Laboratory, and grant #530–1415–01 from the DARPA Ultra program.

proofs for standard (secret- and public-) key distribution schemes, and the insecurity of the most widely used classical schemes (such as RSA [27]) against potential attacks by quantum computers [29], emphasizes the need for provably information-theoretically secure key distribution.

Whereas the security of idealized quantum key distribution (QKD) schemes has been reported against very sophisticated collective [8, 7] and joint [22, 23, 6] attacks, we show here that already very simple attacks severely disturb the security of existing experimental schemes, for the chosen transmission length and signal strength. For a different parameter region a positive security proof against individual attacks has been given recently [19, 20] making use of ideas presented here.

In the four-state scheme introduced in 1984 by Bennett and Brassard [3], usually referred to as BB84, the sender (Alice) and the receiver (Bob) use two conjugate bases (say, the rectilinear basis, +, and the diagonal basis, ×) for the polarization of single photons. In basis + they use the two orthogonal basis states $|0_+\rangle$ and $|1_+\rangle$ to represent "0" and "1" respectively. In basis × they use the two orthogonal basis states $|0_\times\rangle = (|0_+\rangle + |1_+\rangle)/\sqrt{2}$ and $|1_\times\rangle = (|0_+\rangle - |1_+\rangle)/\sqrt{2}$ to represent "0" and "1". The basis is revealed later on via an authenticated classical channel that offers no protection against eavesdropping. The signals where Bob used the same basis as Alice form the *sifted key* on which Bob can decode the bit value. In absence of disturbance by an eavesdropper (Eve) and errors of various kinds, the sifted key should be identical between Alice and Bob. The remaining signals are ignored in the protocol and in this security analysis. Finally, Alice and Bob test a few bits to estimate the error rate, and if it is less than some threshold, they use error correction and privacy amplification [2, 4] to obtain a secure final key [6, 23].

In order to be practical and secure, a quantum key distribution scheme must be based on existing—or nearly existing—technology, but its security must be guaranteed against an eavesdropper with unlimited computing power whose technology is limited only by the rules of quantum mechanics. The experiments that have been performed so far are usually based on weak coherent pulses (WCP) as signal states with a low probability of containing more than one photon [2, 13, 30, 11, 25]. Initial security analysis of such weak-pulse schemes were done [2, 15], and evidence of some potentially severe security problems (which do not exist for the idealized schemes) have been shown [15, 32].

Using a conservative definition of security, we investigate such limitations much further to show insecurity of various existing setups, and to provide several explicit limits on experimental QKD. First, we show that secure QKD to arbitrary distance can be totally impossible for given losses and detector dark counts, even with the assumption of a perfect source. Second we show that QKD can be totally insecure even with perfect detection, if considering losses and multi-photon states. In a combination we compute a maximal distance beyond which (for any given source and detection units) secure QKD schemes cannot be implemented. Finally we prove the advantage of a better source which makes use of parametric downconversion (PDC).

2 Losses and Dark Counts

The effect of losses is that single-photon signals will arrive only with a probability F at Bob's site where they will lead to a detection in Bob's detectors with a probability η_{B} (detection efficiency). This leads to an expected probability of detected signals given by $p_{\mathrm{exp}}^{\mathrm{signal}} = F\eta_{\mathrm{B}}$. For optical fibres, as used for most current experiments, the transmission efficiency F is connected to the absorption coefficient β of the fibre, the length ℓ of the fibre and a distance-independent constant loss in optical components c, via the relation

$$F = 10^{-\frac{\beta\ell + c}{10}} \tag{1}$$

which, for given β and c, gives a one-to-one relation between distance and transmission efficiency. Quantum key distribution can also be achieved through free space [2, 11], in which case the problem of lossy fibres is replaced by the problem of beam broadening. Each of Bob's detectors is also characterized by a dark count probability d_{B} per time slot in the absence of the real signal, so that for a typical detection apparatus with two detectors the total dark count probability is given by $p_{\mathrm{exp}}^{\mathrm{dark}} \approx 2d_{\mathrm{B}}$. The dark counts are due to thermal fluctuations in the detector, stray counts, etc. Throughout the paper we assume conservatively that Eve has control on channel losses and on η_{B}, that all errors are controlled by Eve (including dark counts), and that Bob's detection apparatus cannot resolve the photon number of arriving signals. Without these assumptions, one gets a relaxed security condition, which, however, is difficult to analyse and to justify.

The total expected probability of detection events is given by

$$\begin{aligned} p_{\mathrm{exp}} &= p_{\mathrm{exp}}^{\mathrm{signal}} + p_{\mathrm{exp}}^{\mathrm{dark}} - p_{\mathrm{exp}}^{\mathrm{signal}} p_{\mathrm{exp}}^{\mathrm{dark}} \\ &\leq p_{\mathrm{exp}}^{\mathrm{signal}} + p_{\mathrm{exp}}^{\mathrm{dark}} \quad . \end{aligned} \tag{2}$$

There are two differently contributing error mechanisms. The signal contributes an error with some probability due to misalignment or polarization diffusion. On the other hand, a dark count contributes with probability approximately $1/2$ to the error rate. As the transmission efficiency F becomes smaller and smaller when the distance ℓ is increased, the errors due to dark counts become dominant. Therefore, considering the relevant limit where we can neglect the coincidence probability between a signal photon and a dark count, or between dark counts in both detectors, we have for the error rate e (per sent signal) the approximate lower bound

$$e \succeq \tfrac{1}{2}\, p_{\mathrm{exp}}^{\mathrm{dark}} \quad , \tag{3}$$

where "$x \succeq y$" means that x is approximately greater than or equal to y, when second-order terms are neglected. The contribution to the error rate per sifted key bit is then given by $p_e = e/p_{\mathrm{exp}}$.

If the error rate per sifted key bit p_e exceeds $1/4$, there is no way to create a secure key. With such an allowed error rate, a simple intercept/resend attack (in which Eve measures in one of the two bases and resends according to her

identification of the state) causes Bob and Eve to share (approximately) half of Alice's bits and to know nothing about the other half; hence, Bob does not possess information which is unavailable to Eve, and no secret key can be distilled. Using $p_e = e/p_{\rm exp}$ and $p_e < \frac{1}{4}$, we obtain a necessary condition for secure QKD

$$e < \tfrac{1}{4}\, p_{\rm exp} \ , \tag{4}$$

and using (2, 3), we finally obtain $p_{\rm exp}^{\rm signal} \succeq p_{\rm exp}^{\rm dark}$.

For ideal single-photon states we therefore obtain (with $p_{\rm exp}^{\rm signal} = F\eta_{\rm B}$ and $p_{\rm exp}^{\rm dark} \approx 2d_{\rm B}$) the bound $F\eta_{\rm B} \succeq 2d_{\rm B}$. We see that even for ideal single-photon sources (SP), the existence of a dark count rate leads to a minimum transmission efficiency

$$F > F_{\rm SP} \approx 2d_{\rm B}/\eta_{\rm B} \tag{5}$$

below which QKD cannot be securely implemented. Even for perfect detection efficiency ($\eta_{\rm B} = 1$) we get a bound $F > F_{\rm SP} \approx 2d_{\rm B}$. These bounds correspond, according to (1), to a maximal covered distance, which mainly depends on β.

3 Losses and Imperfect Sources

In a quantum optical implementation, single-photon states would be ideally suited for quantum key distribution. However, such states have not yet been practically implemented for QKD, although proposals exist and experiments have been performed to generate them for other purposes. The signals produced in the experiments usually contain zero, one, two, etc., photons in the same polarization (with probabilities p_0, p_1, p_2, etc., respectively). The multi-photon part of the signals, $p_{\rm multi} = \sum_{i\geq 2} p_i$, leads to a severe security gap, as has been anticipated earlier [2, 15, 32]. Let us present the *photon number splitting* (PNS) attack, which is a modification of an attack suggested in [15] (the attack of [15] was disputed in [32] so the modification is necessary): Eve deterministically splits one photon off each multi-photon signal. To do so, she projects the state onto subspaces characterized by n, which is the total photon number, which can be measured via a quantum nondemolition (QND) measurement. The projection into these subspaces does not modify the polarization of the photons. Then she performs a polarization-preserving splitting operation, for example by an interaction described by a Jaynes-Cummings Hamiltonian [26, 16] or an active arrangement of beamsplitters combined with further QND measurements. She keeps one photon and sends the other $(n-1)$ photons to Bob. When receiving the data regarding the basis, Eve measures her photon and obtains full information. Each signal containing more than one photon in this way will yield its complete information to an eavesdropper.

The situation becomes worse in the presence of loss, in which case the eavesdropper can replace the lossy channel by a perfect quantum channel and forward to Bob only chosen signals. This suppression is controlled such that Bob will find precisely the number of non empty signals as expected given the characterization

of the lossy channel. If there is a strong contribution by multi-photon signals, then Eve can use only those signals and suppress the single-photon signals completely, to obtain full information on the transmitted bits. For an error-free setup, this argument leads to the necessary condition for security,

$$p_{\mathrm{exp}} > p_{\mathrm{multi}} \ , \tag{6}$$

where now the signal contribution is given by

$$p_{\mathrm{exp}}^{\mathrm{signal}} = \sum_i p_i [1 - (1-F)^i] \ . \tag{7}$$

If this condition is violated, Eve gets full information without inducing any errors nor causing a change in the expected detection rate. For given probabilities p_i and transmission rate F, a bound on the distance is obtained, even for perfect detection. The limitation on practical QKD as shown in (6) was reported independently in [10, 18] after having been anticipated in [15].

Whereas this work concentrates mainly on insecurity results, we make here also an important observation, which is useful for positive security proofs. For a general source (emitting into the four BB84 polarization modes) analysing all possible attacks in a large Hilbert space (the Fock space) is a very difficult task. However, if Alice can dephase the states to create a mixture of "number states" (in the chosen BB84 polarization state) the transmitted signals are replaced by mixed states. Then, these states do not change at all when Eve performs a QND measurement on the total photon number as part of a PNS attack! Therefore Eve can be assumed to perform the QND part of the PNS attack without loss of generality. In that case, it is much easier to check that the PNS attack is actually optimal since we start with an eavesdropper who knows the total photon number of each signal. Fortunately, in realistic scenarios the dephasing happens automatically since the eavesdropper has no reference phase to the signal. Therefore, the signal states appear to be phase-averaged ("dephased") signals from her perspective. In some experiments, a phase reference exists initially [25], but could be destroyed by Alice adding random optical phase shifts to her weak signals. Following this observation, a complete positive security proof against all individual particle attacks has been subsequently given [19, 20]. More sophisticated collective and joint attacks can also potentially be restricted to the PNS attacks.

4 Putting it All Together

Let us return to the necessary condition for security. We can combine the idea of the two criteria (4, 6) above to a single, stronger one, given by

$$e < \tfrac{1}{4} \left(p_{\mathrm{exp}} - p_{\mathrm{multi}} \right) \ . \tag{8}$$

This criterion stems from the scenario that Eve splits all multi-photon signals while she eavesdrops on some of the single-photon signals—precisely on a proportion $(p_{\mathrm{exp}} - p_{\mathrm{multi}})/p_1$ of them—via the intercept/resend attack presented

before, and suppresses all other single photon signals. We can think of the key as consisting of two parts: an error-free part stemming from multi-photon signals, and a part with errors coming from single-photon signals. The error rate in the second part has therefore to obey the same inequality as used in criterion (4).

We now explore the consequences of the necessary condition for security for two practical signal sources. These are the weak coherent pulses and the signals generated by parametric downconversion.

5 Weak Coherent Pulse Implementations

In QKD experiments, the signal states are, typically, weak coherent pulses (WCP) containing, on average, much less than one photon. The information is contained in polarization mode of the WCP.

Coherent states

$$|\alpha\rangle = e^{-\alpha^2/2} \sum_n \alpha^n/\sqrt{n!}\,|n\rangle \tag{9}$$

with amplitude α (chosen to be real) give a photon number distribution (per pulse [9])

$$p_n = e^{-\alpha^2} \left(\alpha^2\right)^n / n! \ . \tag{10}$$

Since we analyse PNS attacks only, it doesn't matter if the realistic "coherent state" is a mixture of number states. Thus,

$$p_{\rm exp}^{\rm signal} = \sum_{n=1}^{\infty} e^{-F\eta_{\rm B}\alpha^2} \left(F\eta_{\rm B}\alpha^2\right)^n / n! \tag{11}$$

and

$$p_{\rm multi} = \sum_{n=2}^{\infty} e^{-\alpha^2} \left(\alpha^2\right)^n / n! \ . \tag{12}$$

With $p_{\rm exp} \leq p_{\rm exp}^{\rm signal} + 2d_{\rm B}$ and the error rate $e \succeq d_{\rm B}$ in (8) we find for $\alpha^2 \ll 1$ (by expanding to 4th order in α and neglecting the term proportional to $F^2\eta_{\rm B}^2\alpha^4$) the result

$$F \succeq \frac{2d_{\rm B}}{\eta_{\rm B}\,\alpha^2} + \frac{\alpha^2}{2\,\eta_{\rm B}} \ . \tag{13}$$

The optimal choice $\alpha^2 = 2\sqrt{d_{\rm B}}$ leads to the bound

$$F > F_{\rm WCP} \approx 2\sqrt{d_{\rm B}}/\eta_{\rm B} \ . \tag{14}$$

To illustrate this example we insert numbers $\eta_{\rm B} = 0.11$ and $d_{\rm B} = 5 \times 10^{-6}$ taken from the experiment performed at $1.3\mu m$ by Marand and Townsend [21]. Then

the criterion gives $F \succeq 0.041$. With a constant loss of 5 dB and a fibre loss at 0.38 dB/km, this is equivalent, according to (1), to a maximum distance of 24 km at an average (much lower than standard) photon number of 4.5×10^{-3}. As we used approximations to reach (14), the achievable distance could differ slightly from this value either way.

With $\alpha^2 = 0.1$, as in the literature, secure transmission to any distance is impossible, according to our conditions. In that case, even if we assume η_{B} to be out of control of the eavesdropper, we find that secure transmission to a distance of more than 21 km is impossible. Frequently we find even higher average photon numbers in the literature, although Townsend has demonstrated the feasibility of QKD with intensities as low as $\alpha^2 = 3 \times 10^{-5}$ at a wavelength of $0.8\mu m$ [30].

6 Parametric Downconversion Implementations

The WCP scheme seems to be prone to difficulties due to the high probability of signals carrying no photons (the vacuum contribution). This can be overcome in part by the use of a parametric downconversion (PDC) scheme, which serves to approximate single-photon states. Parametric downconversion has been used before for QKD [12, 28]. We use a different formulation, which enables us to analyse the advantages and limits of the PDC method relative to the WCP approach.

To a good approximation, PDC produces pairs of photons. Although each pair creation occurs at a random time, the two photons in the pair are created simultaneously, and they are correlated in energy, direction of propagation and polarization. Thus, detection of one photon provides information about the existence and properties of the partner photon without any destructive probing of the partner photon itself [14]. More technically, we create the state in an output mode described by photon creation operator $a^\dagger$ conditioned on the detection of a photon in another mode described by $b^\dagger$. If we neglect dispersion, then the output of the PDC process is described [31] on the two modes with creation operators $a^\dagger$ and $b^\dagger$ using the operator

$$T_{a\,b}(\chi) = e^{\mathrm{i}\chi(a^\dagger b^\dagger - a\,b)} \quad , \tag{15}$$

with $\chi \ll 1$, as

$$\begin{aligned} |\Psi_{a\,b}\rangle &= T_{a\,b}(\chi)|0,0\rangle \\ &\approx \left(1 - \tfrac{1}{2}\,\chi^2 + \tfrac{5}{24}\,\chi^4\right)|0,0\rangle + \left(\chi - \tfrac{5}{6}\,\chi^3\right)|1,1\rangle \\ &\quad + \left(\chi^2 - \tfrac{7}{6}\,\chi^4\right)|2,2\rangle + \chi^3|3,3\rangle + \chi^4|4,4\rangle \quad . \end{aligned} \tag{16}$$

This state is a superposition of two–mode number states where $|m,m\rangle$ corresponds to a *flux* of m photons in each mode. Whereas the earlier discussion on the WCP concerns distinct pulses, and the number state corresponds to a specific number of photons in the pulse (i.e. localized in time), the continuous output of the PDC is better represented in terms of photon flux states [9]. On the other

hand, we can interpret these number states for PDC as localized number states, to compare with the WCP case, by assuming the presence of choppers in each of the modes. A chopper periodically blocks the mode, thus converting a continuous output into a periodic sequence of pulses. By placing synchronized choppers in each mode, the continuous output becomes a sequence of pulses and the photon flux state can be regarded as a photon number state (per pulse).

If we had an ideal detector resolving photon numbers (that is, a perfect counter) then we could create a perfect single-photon state by using the state in mode a conditioned on the detection of precisely one photon in the pulse in mode b. However, realistic detectors useful for this task have a single-photon detection efficiency far from unity and can resolve the photon number only at high cost, if at all. Therefore, we assume a detection model which is described by a finite detection efficiency η_{A} and gives only two possible outcomes: either it is not triggered or it is triggered, thereby showing that at least one photon was present. The detector may experience a dark count rate at d_{A} per time slot. The two POVM elements describing this kind of detector can be approximated for our purpose by

$$E_0 = (1 - d_{\mathrm{A}})|0\rangle\langle 0| + \sum_{n=1}^{\infty}(1 - \eta_{\mathrm{A}})^n |n\rangle\langle n| \tag{17}$$

and

$$E_{\mathrm{click}} = d_{\mathrm{A}}|0\rangle\langle 0| + \sum_{n=1}^{\infty}(1 - (1 - \eta_{\mathrm{A}})^n)|n\rangle\langle n| \ . \tag{18}$$

The reduced density matrix for the output signal in mode b conditioned on a click of the detector monitoring mode a is then given by

$$\begin{aligned}\rho &= \frac{1}{N}\mathrm{Tr}_b\left[|\Psi_{a\,b}\rangle\langle\Psi_{a\,b}|E_{\mathrm{click}}\right] \\ &\approx \frac{1}{N}\Big[d_{\mathrm{A}}\left(1 - \chi^2 + \tfrac{2}{3}\chi^4\right)|0\rangle\langle 0| \\ &\quad + \eta_{\mathrm{A}}\chi^2\left(1 - \tfrac{5}{3}\chi^2\right)|1\rangle\langle 1| + \eta_{\mathrm{A}}(2 - \eta_{\mathrm{A}})\chi^4|2\rangle\langle 2|\Big]\end{aligned} \tag{19}$$

with the normalization constant N. To create the four signal states we rotate the polarization of the signal, for example using a beam-splitter and a phase shifter. Note that a mixture of Fock states is created by the detection process, so that the PNS attack is optimal for Eve.

After some calculation following the corresponding calculation in the WCP case, the necessary condition for security (8) takes for the signal state (19) the form

$$F \geq \frac{2d_{\mathrm{A}}\ d_{\mathrm{B}}}{\eta_{\mathrm{A}}\ \eta_{\mathrm{B}}\ \chi^2} + \frac{2d_{\mathrm{B}}}{\eta_{\mathrm{B}}} + \frac{2 - \eta_{\mathrm{A}}}{\eta_{\mathrm{B}}}\chi^2 \tag{20}$$

since we assume $d_{\mathrm{B}} \ll 1$ and $\chi^2 \ll 1$ and neglect terms going as χ^4, $d_{\mathrm{B}}d_{\mathrm{A}}$, and $\chi^2 d_{\mathrm{B}}$. The first error term is due to coincidence of dark counts, the second error

term is due to coincidence of a photon loss and a dark count at Bob's site; the third term is the effect of multi photon signal (signals that leak full information to the eavesdropper). As in the WCP case, the optimal choice of

$$\chi^2 = \sqrt{\frac{2d_{\rm A}d_{\rm B}}{\eta_{\rm A}(2-\eta_{\rm A})}} \tag{21}$$

leads to the necessary condition for security

$$F > F_{\rm PDC} \approx 2\sqrt{\frac{2d_{\rm A}\ d_{\rm B}\ (2-\eta_{\rm A})}{\eta_{\rm A}\ \eta_{\rm B}^2}} + \frac{2d_{\rm B}}{\eta_{\rm B}}\ . \tag{22}$$

If we now assume that Alice and Bob use the same detectors as in the WCP case with the numbers provided by [21], we obtain $F_{\rm PDC} \succeq 8.4 \times 10^{-4}$ corresponding via (1) to a distance of 68 km.

Since we can use downconversion setups which give photon pairs with different wavelength, we can use sources so that one photon has the right wavelength for transmission over long distances, e.g. 1.3 μm, while the other photon has a frequency which makes it easier to use efficient detectors [12]. In the limit of Alice using perfect detectors (but not perfect counters), $\eta_{\rm A} = 1$ and $d_{\rm A} = 0$, we obtain

$$F_{\rm PDC} \approx 2d_{\rm B}/\eta_{\rm B}\ , \tag{23}$$

as for single-photon sources, yielding a maximal distance of 93 km. This optimal distance might also be achievable using new single-photon sources of the type suggested in [17].

7 Conclusions

We have shown a necessary condition for secure QKD which uses current experimental implementations. We find that secure QKD might be achieved with the present experiments using WCP if one would use appropriate parameters for the expected photon number, which are considerably lower than those used today. With current parameters, it seems that all current WCP experiments cannot be proven secure. The distance that can be covered by QKD is mainly limited by the fibre loss, but, with $\alpha^2 > 0.1$, WCP schemes might be totally insecure even to zero distance (in several of the existing experiments), due to imperfect detection. The distance can be increased by the use of parametric downconversion as a signal source, but even in this case the fundamental limitation of the range persists, and a radical reduction of β or of the dark counts is required in order to increase the distance to thousands of kilometers.

The proposed "4+2" scheme [15], in which a strong reference pulse (as in [1]) from Alice is used in a modified detection process by Bob, might not suffer from the sensitivities discussed here, but the security analysis would have to follow

different lines. The use of quantum repeaters [24] (based on quantum error-correction or entanglement purification) or of a string of teleportation stations in the far future can yield secure transmission to any distance, and the security is not altered even if the repeaters or stations are controlled by Eve.

Acknowledgements

We thank C. H. Bennett, N. Gisin, J. H. Kimble, E. Polzik and H. Yuen, for important comments. We are also grateful to one anonymous referee for her generously detailed report.

References

1. C. H. Bennett, "Quantum cryptography using any two nonorthogonal states", *Physical Review Letters*, Vol. 68, no. 21, 25 May 1992, pp. 3121 – 3124.
2. C. H. Bennett, F. Bessette, G. Brassard, L. Salvail and J. Smolin, "Experimental quantum cryptography", *Journal of Cryptology*, Vol. 5, no. 1, 1992, pp. 3 – 28.
3. C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing", in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, Bangalore, India, December 1994, pp. 175 – 179.
4. C. H. Bennett, G. Brassard, C. Crépeau and U. M. Maurer, "Generalized privacy amplification", *IEEE Transactions on Information Theory*, Vol. IT–41, no. 6, November 1995, pp. 1915 – 1923.
5. C. H. Bennett, G. Brassard and A. K. Ekert, "Quantum cryptography", *Scientific American*, Vol. 267, no. 4, October 1992, pp. 50 – 57.
6. E. Biham, M. Boyer, P. O. Boykin, T. Mor and V. Roychowdhury, "A proof of security of quantum key distribution", in *Proceedings of 32nd ACM Symposium on Theory of Computing*, May 2000, to appear.
7. E. Biham, M. Boyer, G. Brassard, J. van-de-Graaf and T. Mor, "Security of quantum key distribution against all collective attacks", Los Alamos archives `quant-ph/9801022`, January 1998.
8. E. Biham and T. Mor, "Bounds on information and the security of quantum cryptography", *Physical Review Letters*, Vol. 79, no. 20, 17 November 1997, pp. 4034 – 4037.
9. K. J. Blow, R. Loudon, S. J. D. Phoenix and T. J. Sheperd, "Continuum fields in quantum optics", *Physical Review A*, Vol. 42, no. 7, October 1990, pp. 4102 – 4114.
10. G. Brassard, T. Mor and B. C. Sanders, "Quantum cryptography via parametric downconversion", in *Quantum Communication, Computing, and Measurement 2*, P. Kumar, G. Mauro D'Ariano and O. Hirota (editors), Kluwer Academic/Plenum Publishers, New York, 2000, pp. 381 – 386.
11. W. T. Buttler, R. J. Hughes, P. G. Kwiat, G. G. Luther, G. L. Morgan, J. E. Nordholt, C. G. Peterson and C. M. Simmons, "Free-space quantum-key distribution", *Physical Review A*, Vol. 57, no. 4, April 1998, pp. 2379 – 2382.
12. A. K. Ekert, J. G. Rarity, P. R. Tapster and G. M. Palma, "Practical quantum cryptography based on two-photon interferometry", *Physical Review Letters*, Vol. 69, no. 9, 31 August 1992, pp. 1293 – 1295.
13. J. D. Franson and H. Ilves, "Quantum cryptography using polarization feedback", *Journal of Modern Optics*, Vol. 41, no. 12, December 1994, pp. 2391 – 2396.

14. P. Hariharan and B. C. Sanders, "Quantum phenomena in optical interferometry", *Progress in Optics*, Vol. XXXVI, 1996, pp. 49 – 128.
15. B. Huttner, N. Imoto, N. Gisin and T. Mor, "Quantum cryptography with coherent states", *Physical Review A*, Vol. 51, no. 3, March 1995, pp. 1863 – 1869.
16. E. T. Jaynes and F. W. Cummings, "Comparison of quantum and semiclassical radiation theories with application to the beam maser", *Proceedings of the IEEE*, Vol. 51, 1963, pp. 89 – 109.
17. J. Kim, O. Benson, H. Kan and Y. Yamamoto, "A single-photon turnstile device", *Nature*, Vol. 397, no. 6719, 1999, pp. 500 – 503.
18. N. Lütkenhaus, "Dim coherent states as signal states in the BB84 protocol: Is it secure?", in *Quantum Communication, Computing, and Measurement 2*, P. Kumar, G. Mauro D'Ariano and O. Hirota (editors), Kluwer Academic/Plenum Publishers, New York, 2000, pp. 387 – 392.
19. N. Lütkenhaus, "Security of quantum cryptography with realistic sources", *Acta Physica Slovaca*, Vol. 49, 1999, pp. 549 – 556.
20. N. Lütkenhaus, "Security against individual attacks for realistic quantum key distribution", Los Alamos archives `quant-ph/9910093`, October 1999.
21. C. Marand and P. D. Townsend, "Quantum key distribution over distances as long as 30 km", *Optics Letters*, Vol. 20, 15 August 1995, pp. 1695 – 1697.
22. D. Mayers, "Quantum key distribution and string oblivious transfer in noisy channels", in *Advances in Cryptology: Proceedings of Crypto'96*, Lecture Notes in Computer Science, Vol. 1109, Springer-Verlag, August 1996, pp. 343 – 357.
23. D. Mayers, "Unconditional security in quantum cryptography", Los Alamos archives `quant-ph/9802025`, February 1998.
24. T. Mor, "Quantum memory in quantum cryptography", Ph.D. Thesis, Technion, Haifa, 1997; Los Alamos archives `quant-ph/9906073`.
25. A. Muller, T. Herzog, B. Huttner, W. Tittel, H. Zbinden and N. Gisin, "Plug and Play systems for quantum cryptography", *Applied Physics Letters*, Vol. 70, 1997, pp. 793 – 795.
26. K. Mølmer, private communication.
27. R. L. Rivest, A. Shamir and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, Vol. 21, no. 2, 1978, pp. 120 – 126.
28. A. V. Sergienko, M. Atatüre, Z. Walton, G. Jaeger, B. E. A. Saleh and M. C. Teich, "Quantum cryptography using femtosecond-pulsed parametric down-conversion", *Physical Review A*, Vol. 60, no. 4, October 1999, pp. R2622 – R2625.
29. P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM Journal on Computing*, Vol. 26, no. 5, 1997, pp. 1484 – 1509.
30. P. D. Townsend, "Experimental investigation of the performance limits for first telecommunications-window quantum cryptography systems", *IEEE Photonics Technology Letters*, Vol. 10, 1998, pp. 1048 – 1050.
31. D. F. Walls and G. J. Milburn, *Quantum Optics*, Springer-Verlag, Heidelberg, 1994.
32. H. P. Yuen, "Quantum amplifiers, quantum duplicators and quantum cryptography", *Quantum and Semiclassical Optics*, Vol. 8, 1996, pp. 939 – 949.

Perfectly Concealing Quantum Bit Commitment from any Quantum One-Way Permutation

Paul Dumais[1] *, Dominic Mayers[2] **, and Louis Salvail[3] ***

[1] Université de Montréal, Dept. of Computer Science, dumais@iro.umontreal.ca
[2] NEC Research Institute, Princeton, N-J, USA, mayers@research.nj.nec.com
[3] BRICS†, Dept. of Computer Science,University of Århus, Århus, Denmark. salvail@brics.dk

Abstract. We show that although unconditionally secure quantum bit commitment is impossible, it can be based upon any family of quantum one-way permutations. The resulting scheme is unconditionally concealing and computationally binding. Unlike the classical reduction of Naor, Ostrovski, Ventkatesen and Young, our protocol is non-interactive and has communication complexity $O(n)$ qubits for n a security parameter.

1 Introduction

The non-classical behaviour of quantum information provides the ability to expand an initially short and secret random secret-key shared between a pair of trusted parties into a much longer one without compromising its security. The BB84 scheme was the first proposed quantum secret-key expansion protocol[3] and was shown secure by Mayers [12, 14]. Secret-key expansion being incompatible with classical information theory indicates that quantum cryptography is more powerful than its classical counterpart. However, quantum information has also fundamental limits when cryptography between two potentially collaborative but untrusted parties is considered. Mayers [13] has proven that any quantum bit commitment scheme can either be defeated by the committer or the receiver as long as both sides have unrestricted quantum computational power. Mayers' general result was built upon previous works of Mayers [11] and Lo and Chau [9].

However, the no-go theorem does not imply that quantum cryptography in the two-party case is equivalent to complexity-based classical cryptography. For example, quantum bit commitment schemes can be built from physical assumptions that are independent of the existence of one-way functions [16]. Moreover, bit commitment is sufficient for quantum oblivious transfer [4, 19] which would

* Supported by a NSERC grant, part of this work was done while visiting BRICS and McGill SOCS.
** Part of this work was done while visiting BRICS and NEC Tsukuba Laboratory, Japan.
*** Supported by the Thomas B. Thriges Center for KvanteInformatik (CKI).
† Basic Research in Computer Science of the Danish National Research Foundation.

be true in the classical world only if one-way functions imply trapdoor one-way functions [8]. The physical assumption addressed in [16] restricts the size of the entanglement the adversary's quantum computer can deal with. Implementing any successful attack was shown, for a particular protocol with security parameter n, to require a $\Omega(n)$-qubits quantum computer. However, such a physical assumption says nothing about the complexity of the attack. In this paper, we construct an unconditionally concealing quantum bit commitment scheme which can be attacked successfully only if the adversary can break a general quantum computational assumption.

We show that similarly to the classical case [15], unconditionally concealing quantum bit commitment scheme can be based upon any family of quantum one-way permutations. This result is not the direct consequence of the classical construction proposed by Noar, Ostrovsky, Ventkatesen and Young (NOVY) [15]. One reason is that NOVY's analysis uses classical derandomization techniques (rewinding) in order to reduce the existence of an inverter to a successful adversary against the binding condition. In [18], it is shown that such a proof fails completely in a quantum setting: if rewinding was possible then no quantum one-way permutation would exist. Therefore, in order to show that NOVY's protocol is conditionally binding against the quantum computer, one has to provide a different proof.

We present a different construction using quantum communication in order to enforce the binding property. In addition, whereas one NOVY's commitment requires $\Omega(n)$ rounds (in fact $n-1$ rounds) of communication for some security parameter n, our scheme is non-interactive. Whether or not this is possible to achieve classically is still an open question. In addition, the total amount of communication of our scheme is $O(n)$ qubits which also improves the $\Omega(n^2)$ bits needed in NOVY's protocol, as far as qubits and bits may be compared. Since unconditionally concealing bit commitment is necessary and sufficient for Zero-Knowledge arguments [5], using our scheme gives implementations requiring few rounds of interaction with provable security based upon general computational assumptions. Perfectly concealing commitment schemes are required for the security of several applications (as in [5]). Using them typically forces the adversary to break the computational assumption before the end of the opening phase, whereas if the scheme was computationally concealing the dishonest receiver could carry out the attack as long as the secret bit remains relevant. Any secure application using NOVY as a sub-protocol can be replaced by one using our scheme instead thus improving communication complexity while preserving the security.

This work provides motivations for the study of one-way functions in a quantum setting. Quantum one-way functions and classical one-way functions are not easily comparable [6]. On the one hand, Shor's algorithm [17] for factoring and extracting discrete logs rules out any attempt to base quantum one-wayness upon those computational assumptions. This means that several flexible yet useful classical one-way functions cannot be used for computationally based quantum cryptography.

On the other hand, because the quantum computer evaluates some functions more efficiently than the classical one, some quantum one-way functions might not be classical one-way since classical computers could even not be able to compute them in the forward direction. This suggests that quantum cryptography can provide new foundations for computationally based security in cryptography.

Organization. First, we give some preliminaries and definitions in Sect.2. Therein, we define the model of computation, quantum one-way functions, and the security criteria for the binding condition. In Sect. 3, we describe our perfectly concealing but computationally binding bit commitment scheme. In Sect. 4, we show that our scheme is indeed unconditionally concealing. Then we model the attacks against the binding condition in Sec. 5. Section 6 reduces the existence of a perfect inverter for a family of one-way permutations to any perfect adversary against the binding condition of our scheme. In Sect. 7, we extend the reduction by showing that any efficient adversary to the binding condition implies an inverter for the family of one-way permutations working efficiently and having good probability of success.

2 Preliminaries

After having introduced the basic quantum ingredients, we define quantum one-way functions and the attacks against the binding condition of computationally binding quantum commitment schemes. We assume the reader familiar with the basics of quantum cryptography and computation.

2.1 Quantum Encoding

In the following, we denote the m-dimensional Hilbert space by $\mathcal{H}_m$. The basis $\{|0\rangle, |1\rangle\}$ denotes the computational or rectilinear or "+" basis for $\mathcal{H}_2$. When the context requires, we write $|b\rangle_+$ to denote the bit b in the rectilinear basis. The diagonal basis, denoted "×", is defined as $\{|0\rangle_\times, |1\rangle_\times\}$ where $|0\rangle_\times = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle_\times = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The states $|0\rangle, |1\rangle, |0\rangle_\times$ and $|1\rangle_\times$ are the four BB84 states. For any $x \in \{0,1\}^n$ and $\theta \in \{+,\times\}^n$, the state $|x\rangle_\theta$ is defined as $\otimes_{i=1}^n |x_i\rangle_{\theta_i}$. An orthogonal (or Von Neumann) measurement of a quantum state in $\mathcal{H}_m$ is described by a set of m orthogonal projections $\mathbb{M} = \{\mathbb{P}_i\}_{i=1}^m$ acting in $\mathcal{H}_m$ thus satisfying $\sum_i \mathbb{P}_i = \mathbb{1}_m$ for $\mathbb{1}_m$ denoting the identity operator in $\mathcal{H}_m$. Each projection or equivalently each index $i \in \{1, \ldots, m\}$ is a possible classical outcome for $\mathbb{M}$. In the following, we write $\mathbb{P}_+^0 = |0\rangle\langle 0|$, $\mathbb{P}_+^1 = |1\rangle\langle 1|$, $\mathbb{P}_\times^0 = |0\rangle_\times\langle 0|$ and $\mathbb{P}_\times^1 = |1\rangle_\times\langle 1|$ for the projections along the four BB84 states. We also define for any $y \in \{0,1\}^n$ the projection operators $\mathbb{P}_{+^n}^y = \otimes_{i=1}^n \mathbb{P}_+^{y_i}$ and $\mathbb{P}_{\times^n}^y = \otimes_{i=1}^n \mathbb{P}_\times^{y_i}$. Since the basis $+^n$ in $\mathcal{H}_{2^n}$ is the computational basis, we also write $\mathbb{P}^y = \mathbb{P}_{+^n}^y$. In order to simplify the notation, in the following we write $\theta(0) = +$ and $\theta(1) = \times$. For any $w \in \{0,1\}$, we denote by $\mathbb{M}_{\theta(w)^n}$ the Von Neumann measurement $\{\mathbb{P}_{\theta(w)^n}^y\}_{y\in\{0,1\}^n}$. We denote by $\mathbb{M}_n$ for $n \in \mathbb{N}$, the

Von Neumann measurement in the computational basis applied on an n-qubit register.

Finally, in order to indicate that $|\phi\rangle \in \mathcal{H}_{2^r}$ is the state of a quantum register $H_R \simeq \mathcal{H}_{2^r}$ we write $|\phi\rangle^R$. If $H_R \simeq \mathcal{H}_{2^r}$ and $H_S \simeq \mathcal{H}_{2^s}$ are two quantum registers and $|\phi\rangle = \sum_{x\in\{0,1\}^r} \sum_{y\in\{0,1\}^s} \gamma^{x,y}|x\rangle \otimes |y\rangle \in \mathcal{H}_{2^r} \otimes \mathcal{H}_{2^s}$ then we write $|\phi\rangle^{RS} = \sum_{x\in\{0,1\}^r} \sum_{y\in\{0,1\}^s} \gamma^{x,y}|x\rangle^R \otimes |y\rangle^S$ to denote the state of both registers H_R and H_S. Given any transformation U_R that acts on a register H_R and any state $|\phi\rangle \in H_R \otimes H_{Others}$, where H_{Others} corresponds to other registers, we define $U_R\,|\phi\rangle \stackrel{def}{=} (U_R \otimes \mathbb{1}_{Others})\,|\phi\rangle$. We use the same notation when U_R denotes a projection operator.

2.2 Model of Computation and Quantum One-Wayness

Quantum one-way functions are defined as the natural generalization of classical one-way functions. Informally, a quantum one-way function is a *classical* function that can be evaluated *efficiently* by a quantum algorithm but cannot be inverted *efficiently* and with *good* probability of success by any quantum algorithm. An algorithm for inverting a one-way function is called an inverter. In this paper, we model inverters (and adversaries against the binding condition) by quantum circuits built out of the universal set of quantum gates $\mathcal{UG} = \{\mathtt{CNot}, \mathtt{H}, \mathtt{R}_{\mathbb{Q}}\}$, where $\mathtt{CNot}$ denotes the controlled-NOT, $\mathtt{H}$ the one qubit Hadamard gate, and $\mathtt{R}_{\mathbb{Q}}$ is an arbitrary one qubit non-trivial rotation specified by a matrix containing only rational numbers [1]. A circuit $\mathcal{C}$ executed in the reverse direction is denoted $\mathcal{C}^\dagger$. The composition of two circuits $\mathcal{C}_1, \mathcal{C}_2$ is denoted $\mathcal{C}_1 \cdot \mathcal{C}_2$. If the initial state before the execution of a circuit $\mathcal{C}$ is $|\Phi\rangle$, the final state after the execution is $\mathcal{C}|\Phi\rangle$. To compute a deterministic function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$, we need a circuit $\mathcal{C}_n$ on $l(n)$ qubits and we must specify $n \leq l(n)$ input qubits and $m(n) \leq l(n)$ output qubits. The classical input x is encoded in the state $|x\rangle$ of the n input qubits. The other qubits, i.e. the non input qubits, are always initialized in the fixed state $|\mathbf{0}\rangle$. The random classical output of the circuit $\mathcal{C}_n$ on input $x \in \{0,1\}^n$ is defined as the classical outcome of $\mathbb{M}_{m(n)}$ on the $m(n)$ output qubits at the end of the circuit. A family $\mathbf{C} = \{\mathcal{C}_n\}_{n=1}^\infty$ is an *exact family of quantum circuits for the family of deterministic functions* $F = \{f_n : \{0,1\}^n \to \{0,1\}^{m(n)}\}_{n=1}^\infty$ if the the classical output of the circuit $\mathcal{C}_n$ on input $|x\rangle \otimes |\mathbf{0}\rangle \in \mathcal{H}_{2^{l(n)}}$ produces with certainty $f_n(x)$ as output. This definition can be generalized the obvious way in order to cover the non exact case and families of random functions.

The complexity of the circuit $\mathcal{C}_n$ is simply the number $\|\mathcal{C}_n\|_{\mathcal{UG}}$ of elementary gates in $\mathcal{UG}$ contained in $\mathcal{C}_n$. Finally, the family $\mathbf{C}$ is *uniform* if, given 1^n as input, there exists a (quantum) Turing machine that produces $\mathcal{C}_n \in \mathbf{C}$ in (quantum) polynomial time in n. The family $\mathbf{C}$ is *non-uniform* otherwise. Our results hold for both the uniform and the non-uniform cases. The following definition is largely inspired by Luby's definitions for classical one-way functions [10]. Let $\mathbf{x}_n$ be a uniformly distributed random variable over $\{0,1\}^n$.

Definition 1 *A family of deterministic functions* $F = \{f_n : \{0,1\}^n \to \{0,1\}^{m(n)} | n > 0\}$ *is* $R(n)$–secure *quantum one-way if*

- *there exists an exact family of quantum circuits* $\mathbf{C} = \{\mathcal{C}_n\}_{n>0}$ *for* F *such that for all* $n > 0$, $\|\mathcal{C}_n\| \leq poly(n)$ *and*
- *for all family of quantum circuits* $\mathbf{C}^{-1} = \{\mathcal{C}_n^{-1}\}_{n>0}$ *and for all* n *sufficiently large, it is always the case that* $\|\mathcal{C}_n^{-1}\|_{\mathcal{UG}}/S(n) \geq R(n)$ *where* $S(n) = \Pr\left(f_n(\mathcal{C}_n^{-1}(f_n(\mathbf{x}_n))) = f_n(\mathbf{x}_n)\right)$.

Each family of quantum circuits $\mathbf{C}^{-1}$ *is called an inverter and the mapping* $S(n)$ *is called its probability of success.*

Note that whenever f_n is a permutation, $S(n)$ can be written as $S(n) = \Pr\left(f_n(\mathcal{C}_n^{-1}(\mathbf{y}_n)) = \mathbf{y}_n\right)$ where $\mathbf{y}_n$ is a uniformly distributed random variable in $\{0,1\}^n$.

2.3 The Binding Condition

In a non interactive bit commitment scheme, an honest committer A for bit w starts with a system $H_{All} = H_{Keep} \otimes H_{Open} \otimes H_{Commit}$ in the initial state $|\mathbf{0}\rangle$, executes a quantum circuit $\mathcal{C}_{n,w}$ on $|\mathbf{0}\rangle$ returning the final state $|\Psi_w\rangle \in H_{All}$ and finally sends the subsystem H_{Commit} to B in the reduced state $\rho_B(w) = \mathrm{Tr}_A(|\Psi_w\rangle\langle\Psi_w|)$, where A's Hilbert space is $H_A = H_{Keep} \otimes H_{Open}$. Once the system H_{Commit} is sent away to B, A has only access to $\rho_A(w) = \mathrm{Tr}_B(|\Psi_w\rangle\langle\Psi_w|)$, where B's Hilbert space is $H_B = H_{Commit}$. To open the commitment, A needs only to send the system H_{Open} together with w. The receiver B then tests the value of w by measuring the system $H_{Open} \otimes H_{Commit}$ with some measurement that is fixed by the protocol in view of w. He obtains the outcome $w = 0$, $w = 1$, or $w = \perp$ when the value of w is rejected.

An attack of the committer $\tilde{A}$ must start with the state $|\mathbf{0}\rangle$ of some system $H_{All} = H_{Extra} \otimes H_A \otimes H_{Commit}$. A quantum circuit $\mathbf{C}^n$ that acts on H_{All} is executed to obtain a state $|\tilde{\Psi}\rangle$ and the subsystem H_{Commit} is sent to the receiver. Later, any quantum circuit $\mathbf{O}^n$ which acts on $H_{Extra} \otimes H_{Keep} \otimes H_{Open}$ can be executed before sending the subsystem H_{Open} to the verifier. The important quantum circuits which act on $H_{Extra} \otimes H_{Keep} \otimes H_{Open}$ are the quantum circuits $\mathbf{O}_w^n$, $w = 0, 1$, which respectively maximize the probability that the bit $w = 0$ and $w = 1$ is unveiled with success. Therefore, any attack can be modeled by triplets of quantum circuits $\{(\mathbf{C}^n, \mathbf{O}_0^n, \mathbf{O}_1^n)\}_{n>0}$.

The efficiency of an adversary is determined by 1) the total number of elementary gates $T(n) = \|\mathbf{C}^n\|_{\mathcal{UG}} + \|\mathbf{O}_0^n\|_{\mathcal{UG}} + \|\mathbf{O}_1^n\|_{\mathcal{UG}}$ in the three circuits $\mathbf{C}^n$, $\mathbf{O}_0^n$ and $\mathbf{O}_1^n$ and 2) the probabilities $S_w(n)$, $w = 0, 1$, that he succeeds to unveil w using the associated optimal circuit $\mathbf{O}_w^n$. The definition of $S_w(n)$ explicitly requires that the value of w, which the adversary tries to open, is chosen not only before the execution of the measurement on $H_{Open} \otimes H_{Commit}$ by the receiver but also before the execution of the circuit $\mathbf{O}_w^n$ by the adversary.

In the classical world, one can always fix the adversary's committed bit by fixing the content of his random tape, that is, we can require that either the probability to unveil 0 or the probability to unveil 1 vanishes, for every fixed value of the random tape. This way of defining the security of a bit commitment scheme

does not apply in the quantum world because, even if we fix the random tape, the adversary could still introduce randomness in the quantum computation. In particular, a quantum committer can always commit to a superposition of $w = 0$ and $w = 1$ by preparing the following state

$$|\Psi(c_0)\rangle = \sqrt{c_0}\,|0_A\rangle \otimes |\Psi_0\rangle + \sqrt{1-c_0}\,|1_A\rangle \otimes |\Psi_1\rangle, \tag{1}$$

where $|\Psi_0\rangle$ and $|\Psi_1\rangle$ are the honest states generated for committing to 0 and 1 respectively and $|0_A\rangle$ and $|1_A\rangle$ are two orthogonal states of H_{Extra}, an extra ancilla kept by A. In this case, for both value of $w \in \{0,1\}$, the opening circuit $\mathbf{O}_w^n$ can put H_{Open} into a mixture that will unveil w successfully with some non zero probability. So we have $S_0(n), S_1(n) > 0$. The fact that the binding condition $S_0(n) = 0 \vee S_1(n) = 0$ is too strong was previously noticed in [13]. We propose the weaker condition $S_0(n) + S_1(n) - 1 \leq \epsilon(n)$ where $\epsilon(n)$ is negligible (i.e. smaller than $1/poly(n)$ for any polynomial $p(n)$). For classical applications, this binding condition (with $\epsilon(n) = 0$) is as good as if the commiter was forced to honestly commit a random bit (with the bias of his choice) and only had the power to abort in view of the bit. The power of this binding condition for quantum applications is unclear, but we think it is a useful condition even in that context.

We now extend this binding condition to a computational setting. It is convenient to restrict ourselves to the cases where $\mathbf{O}_0^n$ is the identity circuit. We can adopt this restriction without lost of generality because any triplet $(\mathbf{C}^n, \mathbf{O}_0^n, \mathbf{O}_1^n)$ can easily be replaced by the three quantum circuits $(\mathbf{C}_0^n, \mathbb{1}, \mathbf{U}_{0,1}^n)$, where $\mathbf{C}_0^n = (\mathbf{O}_0^n \otimes \mathbb{1}_{Commit}) \cdot \mathbf{C}^n$ and $\mathbf{U}_{0,1}^n = \mathbf{O}_1^n \cdot (\mathbf{O}_0^n)^\dagger$, without changing the adversaries strategy. The difference in complexity between applying $(\mathbf{C}^n, \mathbf{O}_0^n, \mathbf{O}_1^n)$ and $(\mathbf{C}_0^n, \mathbb{1}, \mathbf{U}_{0,1}^n)$ is only $\Delta T(n) = \|\mathbf{O}_0^n\|_{\mathcal{UG}}$. Therefore, the adversary is completely determined by the pair $(\mathbf{C}_0^n, \mathbf{U}_{0,1}^n)$ where $\mathbf{C}_0^n$ acts on all registers in H_{All}, and $\mathbf{U}_{0,1}^n$ is restricted to act only in $H_{Extra} \otimes H_{Keep} \otimes H_{Open}$.

Definition 2 *An adversary $\tilde{A} = \{(\mathbf{C}_0^n, \mathbf{U}_{0,1}^n)\}_n$ for the binding condition of a quantum bit commitment scheme is* $(S(n), T(n))$–successful *if for all $n \in \mathbb{N}$, $\|\mathbf{U}_{0,1}^n\|_{\mathcal{UG}} + \|\mathbf{C}_0^n\|_{\mathcal{UG}} \leq T(n)$ and $S_0(n) + S_1(n) - 1 = S(n)$. An adversary with $S(n) = 1$ is called a* perfect adversary.

Any $(0, T(n))$-successful adversary does not achieve more than what an honest committer is able to do. In order to cheat, an adversary must be $(S(n), T(n))$-successful for some non-negligible $S(n) > 0$. The security of a quantum bit commitment scheme is defined as follow:

Definition 3 *A quantum bit commitment scheme is* $R(n)$–binding *if there exists no $(S(n), T(n)$–successful quantum adversary against the binding condition that satisfies $T(n)/S(n) \leq R(n)$. A quantum bit commitment scheme is* perfectly concealing (statistically concealing) *if the systems received for the commitments of* 0 *and* 1 *are identical (resp. statistically indistinguishable).*

It is easy to verify that if a $R(n)$-binding classical bit commitment scheme (satisfying the classical definition) allows to implement a cryptographic task securely,

then using a $R(n)$-binding quantum bit commitment scheme instead would also provide a secure implementation.

The scheme we describe next will be shown to be perfectly concealing and $\Omega(R(n))$-binding whenever used with a $R(n)^2$-secure family of one-way permutations.

3 The Scheme

Let $\Sigma = \{\sigma_n : \{0,1\}^n \to \{0,1\}^n | n > 0\}$ be a family of one-way permutations. The commitment scheme takes, as common input, a security parameter $n \in \mathbb{N}$ and the description of family Σ. The quantum part of the protocol below is similar to the protocol for quantum coin tossing described in [3]. Given Σ and n, the players determine the instance $\sigma_n : \{0,1\}^n \to \{0,1\}^n \in \Sigma$. A sends through the quantum channel $\sigma_n(x)$ for $x \in_R \{0,1\}^n$ polarized in basis $\theta(w)^n$ where $w \in \{0,1\}$ is the committed bit. B then stores the received quantum state until the opening phase. It is implicit here that B must protect the received system

---

$\texttt{commit}_{\Sigma,n}(w)$

1. A picks $x \in_R \{0,1\}^n$, computes $y = \sigma_n(x)$ for $\sigma_n \in \Sigma$,
2. A sends the quantum state $|\sigma_n(x)\rangle_{\theta(w)^n} \in \mathcal{H}_{Commit}$ to B.

$H_{Commit} \simeq \mathcal{H}_{2^n}$ against decoherence until the opening phase. The opening phase consists only for A to unveil all her previous random choices allowing B to verify the consistency of the announcement by measuring the received state. So, $H_{Open} \simeq \mathcal{H}_{2^n}$ is only used to store classical information.

$\texttt{open}_{\Sigma,n}(w, x)$

1. A announces w and x to B,
2. B measures ρ_B with measurement $\mathbb{M}_{\theta(w)^n}$ thus providing the classical outcome $\tilde{y} \in \{0,1\}^n$,
3. B accepts if and only if $\tilde{y} = \sigma_n(x)$.

---

4 The Concealing Condition

In this section, we show that every execution of $\texttt{commit}_{\Sigma,n}$ conceals w perfectly.

Let ρ_w for $w \in \{0,1\}$ be the density matrix corresponding to the mixture sent by A when classical bit w is committed. Since σ_n is a permutation of the elements in the set $\{0,1\}^n$, we get

$$\rho_0 = \sum_{x \in \{0,1\}^n} 2^{-n} |x\rangle_{+^n} \langle x| = 2^{-n} \mathbb{1}_{2^n} = \sum_{x \in \{0,1\}^n} 2^{-n} |x\rangle_{\times^n} \langle x| = \rho_1 \qquad (2)$$

where $\mathbb{1}_{2^n}$ is the identity operator in $\mathcal{H}_{2^n}$. The following lemma follows directly from (2).

Lemma 1. *Protocol* $\texttt{commit}_{\Sigma,n}(w)$ *is perfectly concealing.*

Proof: The quantum states ρ_0 and ρ_1 are the same. It follows that no quantum measurement can distinguish between the commitments of 0 and 1. □

5 The Most General Attack

Here we describe the most general adversary $\tilde{A} = \{(\mathbf{C}_0^n, \mathbf{U}_{0,1}^n)\}_{n \geq n_0}$ against the binding condition of our scheme. We shall prove that any such attack can be used to invert the one-way permutation in subsequent sections.

The adversary doesn't necessarily know which value will take y on the receiver's side after the measurement $\mathbb{M}_{\theta(w)^n}$ on H_{Commit} associated with the opening of w. He computes $x \in \{0,1\}^n$ using $\mathbf{O}_w^n$, announces (x, w) and hopes that $\sigma_n(x) = y$. So we have that $H_{Open} \simeq \mathcal{H}_{2^n}$ is used to encode $x \in \{0,1\}^n$. We separate the entire system in three parts: the system H_{Commit} that encodes y, the system H_{Open} that encodes x, and the remainder of the system that we conveniently denote all together by H_{Keep} (thus including for simplicity register H_{Extra}). We easily obtain that the states $|\tilde{\Psi}_w^n\rangle = \mathbf{C}_w^n|\mathbf{0}\rangle$, $w = 0, 1$, can be written in the form

$$|\tilde{\Psi}_0^n\rangle = \sum_{x,y \in \{0,1\}^n} |\gamma_0^{x,y}\rangle^{Keep} \otimes |x\rangle^{Open} \otimes |y\rangle_{+^n}^{Commit} = \mathbf{C}_0^n|\mathbf{0}\rangle \tag{3}$$

with $\sum_{x,y} \||\gamma_0^{x,y}\rangle\|^2 = 1$, and

$$|\tilde{\Psi}_1^n\rangle = \sum_{x,y \in \{0,1\}^n} |\gamma_1^{x,y}\rangle^{Keep} \otimes |x\rangle^{Open} \otimes |y\rangle_{\times^n}^{Commit} = \mathbf{U}_{0,1}^n|\tilde{\Psi}_0^n\rangle \tag{4}$$

with $\sum_{x,y} \||\gamma_1^{x,y}\rangle\|^2 = 1$. In the following, we shall refer to states $|\tilde{\Psi}_0^n\rangle$ and $|\tilde{\Psi}_1^n\rangle$ as the 0-state and the 1-state of the attack respectively. The transformation $\mathbf{U}_{0,1}^n$ is applied on the system $H_{Keep} \otimes H_{Open}$.

Next section restricts the analysis to the case where an adversary A can open both $w = 0$ and $w = 1$ with probability of success $p_w = 1$. Such an adversary is called a *perfect adversary*. We show that any perfect adversary can invert efficiently $\sigma_n(x)$ for any $x \in \{0,1\}^n$. In Sect. 7 we generalize the result to all imperfect but otherwise good adversaries. We show that any polynomial time adversary for which $p_0 + p_1 \geq 1 + \frac{1}{poly(n)}$ can invert $\sigma_n(x)$ for $x \in_R \{0,1\}^n$ efficiently and with non-negligible probability of success.

6 Perfect attacks

In this section, we prove that any efficient perfect adversary $A = \{(\mathbf{C}_0^n, \mathbf{U}_{0,1}^n)\}_n$ against the binding condition can be used to invert efficiently the one-way permutation with probability of success 1. In the next section, we shall use a similar technique for the case where the attack is not perfect.

By definition, a perfect adversary A is $(1, T(n))$-successful, that is: $S_0(n) = S_1(n) = 1$. We obtain that $\||\gamma_w^{x,y}\rangle\| = 0$ if $\sigma_n(x) \neq y$:

$$|\Psi_0^n\rangle = \sum_{x\in\{0,1\}^n} |\gamma_0^x\rangle^{Keep} \otimes |x\rangle^{Open} \otimes |\sigma_n(x)\rangle_{+^n}^{Commit} = \mathbf{C}_0^n|\mathbf{0}\rangle \tag{5}$$

where $|\gamma_0^x\rangle$ corresponds to $|\gamma_0^{x,\sigma_n(x)}\rangle$ and $\sum_x \||\gamma_0^x\rangle\|^2 = 1$, and

$$|\Psi_1^n\rangle = \sum_{x\in\{0,1\}^n} |\gamma_1^x\rangle^{Keep} \otimes |x\rangle^{Open} \otimes |\sigma_n(x)\rangle_{\times^n}^{Commit} = \mathbf{U}_{0,1}^n|\Psi_0^n\rangle \tag{6}$$

where $|\gamma_1^x\rangle$ corresponds to $|\gamma_1^{x,\sigma_n(x)}\rangle$ and $\sum_x \||\gamma_1^x\rangle\|^2 = 1$. Any pair of 0-state and 1-state satisfying (5) and (6) is called a *perfect pair*. Any perfect adversary $A = \{(\mathbf{C}_0^n, \mathbf{U}_{0,1}^n)\}_n$ generates a perfect pair for all $n > 0$.

Let $\mathbb{P}_{Commit}^{u,+}$ and $\mathbb{P}_{Commit}^{u,\times}$ be the projection operators $\mathbb{P}_{+^n}^u$ and $\mathbb{P}_{\times^n}^u$ respectively, acting upon register H_{Commit}. We assume that we have an input register $H_Y \simeq \mathcal{H}_{2^n}$ initialized in the basis state $|y\rangle$ on input y. The states $|\Phi_0^n(u)\rangle = \mathbb{P}_{Commit}^{u,\times}|\Psi_0^n\rangle$, $u \in \{0,1\}^n$, play an essential role in the mechanisms used by the inverter. These states have three key properties for every $u \in \{0,1\}^n$:

1. $\||\Phi_0^n(u)\rangle\| = 2^{-n/2}$,
2. there exists a simple circuit $\mathbf{W}_n$ on $H_Y \otimes H_{Open} \otimes H_{Commit}$ which, if u is encoded in register H_Y, unitarily maps $|\Psi_0^n\rangle$ into $2^{n/2}|\Phi_0^n(u)\rangle$, and
3. $\mathbf{U}_{0,1}^n|\Phi_0^n(u)\rangle = |\gamma^{\sigma^{-1}(u)}\rangle^{Keep} \otimes |\sigma_n^{-1}(u)\rangle^{Open} \otimes |u\rangle_{\times^n}^{Commit}$.

On input $y \in \{0,1\}^n$, the inverter creates the state $|\Psi_0^n\rangle$, then applies the circuit $\mathbf{W}_n$, then the circuit $\mathbf{U}_{0,1}^n$, and finally measures the register H_{Open} to obtain $\sigma_n^{-1}(y)$. We now prove these three properties.

6.1 Proof of Properties 1 and 3

First we write the state $|\Psi_0^n\rangle$ using the basis $\times^n$ for the register $H_{Commit} \simeq \mathcal{H}_{2^n}$. We get

$$|\Psi_0^n\rangle = 2^{-n/2} \sum_{u,v\in\{0,1\}^n} (-1)^{u\odot v} |\gamma_0^{\sigma_n^{-1}(v)}\rangle^{Keep} \otimes |\sigma_n^{-1}(v)\rangle^{Open} \otimes |u\rangle_{\times^n}^{Commit}$$

from which we easily obtain, after the change of variable $\sigma_n^{-1}(v) \to x$,

$$|\Phi_0^n(u)\rangle = 2^{-n/2} \sum_{x\in\{0,1\}^n} (-1)^{u\odot\sigma_n(x)} |\gamma_0^x\rangle^{Keep} \otimes |x\rangle^{Open} \otimes |u\rangle_{\times^n}^{Commit}. \tag{7}$$

Property 1 follows from (7). Note that the states $|\Phi_0^n(u)\rangle$ can be mapped one into the other by a unitary mapping, a conditional phase shift which depends on u and x. Because (6) can be rewritten as

$$|\Psi_1^n\rangle = \sum_{u\in\{0,1\}^n} |\gamma_1^{\sigma^{-1}(u)}\rangle^{Keep} \otimes |\sigma^{-1}(u)\rangle^{Open} \otimes |u\rangle_{\times^n}^{Commit},$$

it follows that, for all $u \in \{0,1\}^n$, we have

$$\begin{aligned}\mathbf{U}_{0,1}^n|\Phi_0^n(u)\rangle &= \mathbf{U}_{0,1}^n\,\mathbb{P}_{Commit}^{u,\times}|\Psi_0^n\rangle = \mathbb{P}_{Commit}^{u,\times}\,\mathbf{U}_{0,1}^n\,|\Psi_0^n\rangle \\ &= \mathbb{P}_{Commit}^{u,\times}|\Psi_1^n\rangle = |\gamma^{\sigma^{-1}(u)}\rangle^{Keep}|\sigma_n^{-1}(u)\rangle^{Open}|u\rangle_{\times^n}^{Commit}.\end{aligned}$$

which concludes the proof of property 3.

6.2 Proof of Property 2

A simple comparison of (5) and (7) suggests what needs to be done to obtain $2^{n/2}|\Phi_0^n(y)\rangle$ efficiently starting from $|\Psi_0^n\rangle$. Assume the input register $H_Y = H_Y^1 \otimes \ldots \otimes H_Y^n \simeq \mathcal{H}_{2^n}$ is in the basis state $|y\rangle$. The first step is to add the phase $(-1)^{y\odot\sigma_n(x)}$ in front of each term in the sum of (5). Note that, for every $y \in \{0,1\}^n$, this is a unitary mapping on $H_{Keep} \otimes H_{Open} \otimes H_{Commit}$. It is sufficient to execute a circuit $\hat{\oplus}_1$ which, for each $i \in \{1,\ldots,n\}$, acts on the corresponding pair of qubits in $H_Y^i \otimes H_{Commit}^i$. The circuit $\hat{\oplus}_1$ maps each state $|y_i\rangle \otimes |\sigma_n(x)_i\rangle$, $i = 1,\ldots,n$, into $(-1)^{(y_i \odot \sigma_n(x)_i)}(|y_i\rangle \otimes |\sigma_n(x)_i\rangle)$. It can easily be implemented as $\hat{\oplus}_1 = (\mathtt{H}\otimes\mathbb{1}_{Commit})\cdot\mathtt{CNot}\cdot(\mathtt{H}\otimes\mathbb{1}_{Commit})$ where each $\mathtt{H}$ is applied to register H_Y^i and where register H_{Commit}^i encodes the control bit of the $\mathtt{CNot}$ gate. We denote by $\hat{\oplus}_n$ the complete quantum circuit acting in $H_Y \otimes H_{Commit}$ and applying $\hat{\oplus}_1$ to each pair $i \in \{1,\ldots,n\}$ of qubits $|y_i\rangle \otimes |\sigma_n(x)_i\rangle \in H_Y^i \otimes H_{Commit}^i$.

The second step is to set the register H_{Commit} which contains the state $|\sigma_n(x)\rangle_{+^n}$ into the new state $|y\rangle_{\times^n}$. For this we use the composition of three circuits. The first circuit $\mathbf{U}_{\sigma_n} : |x\rangle^{Open}\otimes|u\rangle^{Commit} \mapsto |x\rangle^{Open}\otimes|u\oplus\sigma_n(x)\rangle^{Commit}$ sets the quantum register H_{Commit} into the new state $|\mathbf{0}\rangle_{+^n}$. Note that $\mathbf{U}_{\sigma_n}$ is the quantum circuit that is guaranteed to compute $\sigma_n(x)$ efficiently. The second circuit is $\oplus_n : |y\rangle^Y \otimes |u\rangle^{Commit} \mapsto |y\rangle^Y \otimes |y\oplus u\rangle^{Commit}$ which sets H_{Commit} into the state $|y\rangle_{+^n}$ by simply applying a $\mathtt{CNot}$ between registers $H_{Commit}^i, H_Y^i \simeq \mathcal{H}_2$ for $i \in \{1,\ldots,n\}$. Finally the third circuit executes the Hadamard transform $\mathtt{H}_n$ on H_{Commit} which maps the $+^n$ basis into the $\times^n$ basis (it is simply n Hadamard gates $\mathtt{H} \in \mathcal{UG}$) . The composition of $\hat{\oplus}_n$ with these three circuits is the circuit $\mathbf{W}_n$ shown in Fig. 1. This circuit allows to generate any $2^{n/2}|\Phi_0^n(y)\rangle$ for $y \in \{0,1\}^n$. Moreover, it is easy to verify that $\|\mathbf{W}_n\|_{\mathcal{UG}} = \|\mathbf{U}_{\sigma_n}\|_{\mathcal{UG}} + 5n$. The following is a

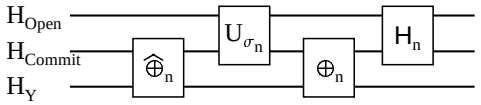


Fig. 1. Transformation $\mathbf{W}_n$.

straightforward consequence of these three properties, the definition of $\mathbf{W}_n$ and the above discussion:

Lemma 2. *If there exists a $(1, T(n))$-successful adversary against $\mathtt{commit}_{\Sigma,n}$ then there exists an adversary against Σ with time-success ratio*

$$R(n) \leq T(n) + \|\mathbf{U}_{\sigma_n}\|_{\mathcal{UG}} + 5n.$$

It follows that the adversary against Σ has about the same complexity than the one against the binding condition of $\texttt{commit}_{\Sigma,n}$. In the next section, we show that the same technique can be applied to the case where the adversary does not implement a perfect attack against $\texttt{commit}_{\Sigma,n}$.

7 The General Case

In this section, we are considering any attack that yields a non-negligible success probability to a cheating committer. In terms of Definition 2, such an adversary $\tilde{A} = \{(\mathbf{C}_0^n, \mathbf{U}_{0,1}^n)\}_n$ must be $(\epsilon(n), T(n))$-successful for some $\epsilon(n) \geq 1/poly(n) \geq 0$. In order for the attack to be efficient, $T(n)$ must also be upper bounded by some polynomial.

In general, the 0-state $|\tilde{\Psi}_0^n\rangle$ and 1-state $|\tilde{\Psi}_1^n\rangle$ of adversary $\tilde{A}$ can always be written as in (3) and (4) respectively. In this general case, the probability of success of unveiling the bit w, i.e. the probability of not being caught cheating, is the probability of the event $\tilde{A}$ *announces a value* x *and the outcome of* B*'s measurement happens to be* $\sigma_n(x)$. One can see easily that this probability is given by :

$$S_w^{\tilde{A}} = S_w^{\tilde{A}}(n) = \sum_v \| |\gamma_w^{v,\sigma_n(v)}\rangle \|^2. \tag{8}$$

If the adversary $\tilde{A}$ is $(\epsilon(n), T(n))$-successful then

$$S_0^{\tilde{A}} + S_1^{\tilde{A}} \geq 1 + \epsilon(n). \tag{9}$$

In that setting, our goal is to show that from such an adversary $\tilde{A}$, $\sigma_n^{-1}(y)$ can be computed similarly to the perfect case and with probability of success at least $1/poly(n)$ whenever $y \in_R \{0,1\}^n$ and $\epsilon(n)^{-1}$ is smaller than some positive polynomial.

7.1 The Inverter

Compared to the perfect case, the inverter for the general case will involve an extra step devised to produce a *perfect* $|\Psi_0^n\rangle$ from the initial and *imperfect* 0-state $|\tilde{\Psi}_0^n\rangle$. Although this preprocessing will succeed only with some probability, any $(\frac{1}{p(n)}, T(n))$-successful adversary can *distill* $|\Psi_0^n\rangle$ from $|\tilde{\Psi}_0^n\rangle$ efficiently and with good probability of success. From $|\Psi_0^n\rangle$, the inverter then proceeds the same way as in the perfect case.

The distillation process involves a transformation $\mathbf{T}_n$ acting in $H_{Open} \otimes H_{Commit} \otimes H_T$ where $H_T \simeq \mathcal{H}_{2^n}$ is an extra register. We define $\mathbf{T}_n$ as:

$$\mathbf{T}_n : |x\rangle^{Open}|y\rangle^{Commit}|a\rangle^T \mapsto |x\rangle^{Open}|y\rangle^{Commit}|\sigma_n(x) \oplus y \oplus a\rangle^T. \tag{10}$$

Clearly, one can always write

$$\mathbf{T}_n(|\tilde{\Psi}_0^n\rangle^{All} \otimes |\mathbf{0}\rangle^T) = \sum_{\sigma_n(x)\neq z} |\gamma_0^{x,z}\rangle^{Keep}|x\rangle^{Open}|z\rangle^{Commit}|\sigma_n(x)\oplus z\rangle^T + \sum_x |\gamma_0^{x,\sigma_n(x)}\rangle^{Keep}|x\rangle^{Open}|\sigma_n(x)\rangle^{Commit}|\mathbf{0}\rangle^T. \quad (11)$$

Upon standard measurement of register H_T in state $|\mathbf{0}\rangle$, the adversary obtains the quantum residue (by tracing out the ancilla):

$$|\Psi_0^n\rangle = \sum_x |\gamma_0^x\rangle^{Keep} \otimes |x\rangle^{Open} \otimes |\sigma_n(x)\rangle^{Commit} \quad (12)$$

where $|\gamma_0^x\rangle^{Keep} = \frac{1}{\sqrt{S_0^{\tilde{A}}}}|\gamma^{x,\sigma_n(x)}\rangle^{Keep}$, with probability

$$S_0^{\tilde{A}} = \sum_v \||\gamma_0^{v,\sigma_n(v)}\rangle\|^2 = |\langle\Psi_0^n|\tilde{\Psi}_0^n\rangle|^2.$$

It is easy to verify that $\mathbf{T}_n$ can be implemented by a quantum circuit of $O(\|\mathbf{U}_{\sigma_n}\|_{\mathcal{UG}})$ elementary gates. On input $y \in_R \{0,1\}^n$, the inverter then works exactly as in the perfect case. In Fig. 2, the quantum circuit for the general inverter $\mathbf{I}_n^{\tilde{A}}(y)$ is shown. The input quantum register is H_Y and the output register is H_{Open}. The output is the outcome of the standard measurement $\mathbb{M}_n$ applied to the output register H_{Open} which hopefully contains $x = \sigma_n^{-1}(y)$. The

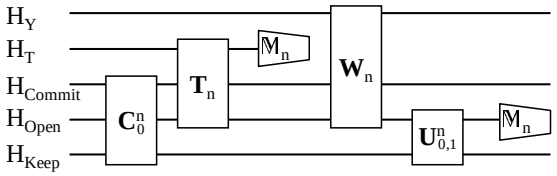


Fig. 2. The inverter $\mathbf{I}_n^{\tilde{A}}(y), y \in \{0,1\}^n$ obtained from adversary $\tilde{A} = (\mathbf{C}_0^n, \mathbf{U}_{0,1}^n)$.

following lemma is straightforward and establishes the efficiency of the inverter in terms of the efficiency of $\tilde{A}$'s against $\texttt{commit}_{\Sigma,n}$:

Lemma 3. *If $\tilde{A}$ is $(\cdot, T(n))$-successful then*

$$\|\mathbf{I}_n^{\tilde{A}}(y)\|_{\mathcal{UG}} \in O(T(n) + \|\mathbf{U}_{\sigma_n}\|_{\mathcal{UG}}).$$

It should be noted that gates $\oplus_n$ and $\mathtt{H}_n$ appearing in circuit $\mathbf{W}_n$ are not taken into account in the statement of Lemma 3. The reason is that none of them influence the final outcome since they commute with the final measurement in H_{Open}. They have been included in $\mathbf{W}_n$ to help the reader with the analysis of the success probability described in the next section.

7.2 Analysis of the Success Probability

Let $\tilde{A} = \{(\mathbf{C}_0^n, \mathbf{U}_{0,1}^n)\}_{n>0}$ be any $(\epsilon(n), \cdot)$-successful adversary for some $\epsilon(n) > 0$ thus satisfying $S_0^{\tilde{A}} + S_1^{\tilde{A}} \geq 1 + \epsilon(n)$. Let $\mathbb{P}_{Open}^x$ be the projection operator $\mathbb{P}^x$ applied to register H_{Open}. We recall that $\mathbb{P}_{Commit}^{y,+}$ and $\mathbb{P}_{Commit}^{y,\times}$ are the projection operators $\mathbb{P}_{+^n}^y$ and $\mathbb{P}_{\times^n}^y$ respectively, acting upon register H_{Commit}. We now define the two projection operators:

$$\boldsymbol{P}_0 = \sum_{x\in\{0,1\}^n} \mathbb{P}_{Open}^x \otimes \mathbb{P}_{Commit}^{\sigma_n(x),+} \text{ and } \boldsymbol{P}_1 = \sum_{x\in\{0,1\}^n} \mathbb{P}_{Open}^x \otimes \mathbb{P}_{Commit}^{\sigma_n(x),\times} \qquad (13)$$

which have the property, using (8), that $S_0^{\tilde{A}} = \|\boldsymbol{P}_0|\tilde{\Psi}_0^n\rangle\|^2$ and $S_1^{\tilde{A}} = \|\boldsymbol{P}_1|\tilde{\Psi}_1^n\rangle\|^2$. Next lemma relates the success probability to projections $\boldsymbol{P}_0$ and $\boldsymbol{P}_1$.

Lemma 4. *The probability of success p_s of inverter $\mathbf{I}_n^{\tilde{A}}(y)$ satisfies*

$$p_s = \|\boldsymbol{P}_1\mathbf{U}_{0,1}^n\boldsymbol{P}_0|\tilde{\Psi}_0^n\rangle\|^2.$$

Proof: We recall that the probability of success is defined in terms of a uniformly distributed input y. We will first compute the probability $p_s(y)$ that the inverter succeeds on input $y \in \{0,1\}^n$. Assume that right after gate $\mathbf{T}_n$, the register H_T is observed in state $|\mathbf{0}\rangle$. The registers $H_{All} \otimes H_Y$ have now collapsed to the state $|y\rangle^Y \otimes |\Psi_0^n\rangle$ where $|\Psi_0^n\rangle$ is the state $\boldsymbol{P}_0|\tilde{\Psi}_0^n\rangle$ after renormalization. Note that $|\Psi_0^n\rangle$ is a perfect 0-state. This event has probability $\|\boldsymbol{P}_0|\tilde{\Psi}_0\rangle\|^2 = S_0^{\tilde{A}}$ to happen according to (12). Next the circuit $\mathbf{W}_n$, with y encoded in H_Y, unitarily maps the state $|\Psi_0^n\rangle$ into the state $2^{n/2}|\Phi_0^n(y)\rangle = 2^{n/2}\,\mathbb{P}_{Commit}^{y,\times}|\Psi_0^n\rangle$ (see Sect. 6). Then the circuit $\mathbf{U}_{0,1}^n$ returns the state $2^{n/2}\,\mathbb{P}_{Commit}^{y,\times}\mathbf{U}_{0,1}^n\,|\Psi_0^n\rangle$. Finally, the register H_{Open} is measured and the probability of success given the initial state $|\Psi_0^n\rangle$ is $\|2^{n/2}\,\mathbb{P}_{Open}^{\sigma_n^{-1}(y)}\,\mathbb{P}_{Commit}^{y,\times}\mathbf{U}_{0,1}^n\,|\Psi_0^n\rangle\|^2$. Using (12), we get that $p_s(y) = S_0^{\tilde{A}}2^n\,\|\mathbb{P}_{Open}^{\sigma_n^{-1}(y)}\,\mathbb{P}_{Commit}^{y,\times}\mathbf{U}_{0,1}^n\,|\Psi_0^n\rangle\|^2 = 2^n\,\|\mathbb{P}_{Open}^{\sigma_n^{-1}(y)}\,\mathbb{P}_{Commit}^{y,\times}\mathbf{U}_{0,1}^n\,\boldsymbol{P}_0\,|\tilde{\Psi}_0^n\rangle\|^2$. Averaging over all values of the uniformly distributed variable y we obtain:

$$\begin{aligned} p_s &= \sum_{y\in\{0,1\}^n} 2^{-n}p_s(y) = \sum_{y\in\{0,1\}^n} \|\left(\mathbb{P}_{Open}^{\sigma_n^{-1}(y)} \otimes \mathbb{P}_{Commit}^{y,\times}\right)\mathbf{U}_{0,1}^n\boldsymbol{P}_0|\tilde{\Psi}_0^n\rangle\|^2 \\ &= \|\left(\sum_{y\in\{0,1\}^n}\mathbb{P}_{Open}^{\sigma_n^{-1}(y)} \otimes \mathbb{P}_{Commit}^{y,\times}\right)\mathbf{U}_{0,1}^n\boldsymbol{P}_0|\tilde{\Psi}_0^n\rangle\|^2 = \|\boldsymbol{P}_1\mathbf{U}_{0,1}^n\boldsymbol{P}_0\,|\tilde{\Psi}_0^n\rangle\|^2 \quad (14) \end{aligned}$$

where (14) is obtained from the fact that $\{\mathbb{P}_{Open}^x \otimes \mathbb{P}_{Commit}^{\sigma_n(x),\times}\}_{x\in\{0,1\}^n}$ is a set of orthogonal projections and from Pythagoras theorem. □

We are now ready to relate the probability of success for the inverter given a *good* adversary against the binding condition of $\texttt{commit}_{\Sigma,n}$.

Lemma 5. *Let $\boldsymbol{I}_n^{\tilde{A}}$ be the inverter obtained from a $(S_0^{\tilde{A}} + S_1^{\tilde{A}} - 1, \cdot)$-successful adversary $\tilde{A}$ with $S_0^{\tilde{A}} + S_1^{\tilde{A}} \geq 1 + \epsilon(n)$ for $\epsilon(n) > 0$ for all $n > 0$. Then the success probability p_s to invert with success a random image element satisfies*

$$p_s \geq (\sqrt{S_1^{\tilde{A}}} - \sqrt{1 - S_0^{\tilde{A}}})^2.$$

Proof: Using lemma 4, we can write

$$\begin{aligned} p_s &= \|\boldsymbol{P}_1 \mathbf{U}_{0,1}^n \boldsymbol{P}_0 |\tilde{\Psi}_0^n\rangle\|^2 = \|\boldsymbol{P}_1 \mathbf{U}_{0,1}^n (\mathbb{1}_{\bar{A}} - \boldsymbol{P}_0^{\perp}) |\tilde{\Psi}_0^n\rangle\|^2 \\ &= \|\boldsymbol{P}_1 \mathbf{U}_{0,1}^n |\tilde{\Psi}_0^n\rangle - \boldsymbol{P}_1 \mathbf{U}_{0,1}^n \boldsymbol{P}_0^{\perp} |\tilde{\Psi}_0^n\rangle\|^2 = \|\boldsymbol{P}_1 |\tilde{\Psi}_1^n\rangle - \boldsymbol{P}_1 \mathbf{U}_{0,1}^n \boldsymbol{P}_0^{\perp} |\tilde{\Psi}_0^n\rangle\|^2. \end{aligned}$$

Using the triangle inequality and $S_1^{\bar{A}} > 1 - S_0^{\bar{A}}$, we are led to

$$\begin{aligned} p_s &\geq \left(\|\boldsymbol{P}_1 |\tilde{\Psi}_1^n\rangle\| - \|\boldsymbol{P}_1 \mathbf{U}_{0,1}^n \boldsymbol{P}_0^{\perp} |\tilde{\Psi}_0^n\rangle\| \right)^2 \\ &\geq \left(\|\boldsymbol{P}_1 |\tilde{\Psi}_1^n\rangle\| - \|\boldsymbol{P}_0^{\perp} |\tilde{\Psi}_0^n\rangle\| \right)^2 = \left(\sqrt{S_1^{\bar{A}}} - \sqrt{1 - S_0^{\bar{A}}} \right)^2. \end{aligned}$$

□

From Lemma 5 and a few manipulations, we conclude that $S_0^{\bar{A}} + S_1^{\bar{A}} > 1 + \epsilon(n)$ implies that $p_s > \epsilon(n)^2/4$. In addition, if $\epsilon(n) \in \Omega(\frac{1}{poly(n)})$ and $T(n) \in O(poly(n))$ then the inverter works in polynomial time with probability of success in $\Omega(1/poly(n)^2)$.

8 Conclusion

The concealing condition is established unconditionally by Lemma 1. Lemmas 3 and 5 imply that any $(S(n), T(n))$-successful adversary against $\texttt{commit}_{\Sigma,n}$ can invert the family of one-way permutations Σ with time-success ratio roughly $T(n)/S(n)^2$. We finally obtain:

Theorem 1. *Let Σ be a $R(n)$-secure family of one-way permutations. Protocol $\texttt{commit}_{\Sigma,n}$ is unconditionally concealing and $R'(n)$–binding where $R'(n) \in \Omega(\sqrt{R(n)})$.*

Our reduction produces only a quadratic blow-up in the worst case between the time-success ratio of the inverter and the time-success ratio of the attack. Compared to NOVY's construction, the reduction is tighter by several degrees of magnitude. If Σ is $T(n)/S(n)$–secure with $\frac{1}{S(n)} \in O(\sqrt{T(n)})$ then the reduction is optimal.

In order for the scheme to be practical, the receiver should not be required to store the received qubits until the opening phase. It is an open question whether or not our scheme is still secure if the receiver measures each qubit π_i upon reception in a random basis $\theta_i \in_R \{+, \times\}$. The opening of $w \in \{0,1\}$ being accepted if each time $\theta_i = \theta(w)$, the announced $x \in \{0,1\}^n$ is such that $[\sigma_n(x)]_i = \tilde{y}_i$. That way, the protocol would require similar technology than the one needed for implementing the BB84 quantum-key distribution protocol [2].

It is also not clear how to modify the scheme in order to deal with noisy quantum transmissions. Another problem linked to practical implementation is the lack of tolerance to multi-photon pulses. If for $x, w \in \{0,1\}$, the quantum state $|\phi_x\rangle_{\theta(w)} \otimes |\phi_x\rangle_{\theta(w)}$ is sent instead of $|\phi_x\rangle_{\theta(w)}$ then $\texttt{commit}_{\Sigma,n}$ is no more

concealing. Moreover, it is impossible in practice to make sure that only one qubit per pulse is sent.

Our main open problem is the finding of candidates for families of quantum one-way permutations or functions. If a candidate family of quantum one-way functions was also computable efficiently on a classical computer then classical cryptography could provide computational security even against quantum adversaries. It would also be interesting to find candidates one-way functions that are not classical one-way. Quantum cryptography could then provide a different basis for computational security in cryptography.

Acknowledgements. Thanks to Ivan Damgård for several enlightening discussions and to Peter Høyer for helping with the circuitry. Thanks also to Alain Tapp for helpful comments on earlier drafts.

References

1. Barenco, A., C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin and H. Weinfurter, "Elementary Gates for Quantum Computation", *Physical Review A*, vol. 52, no 5, November 1995, pp. 3457 – 3467.
2. Bennett, C.H., F. Bessette, G. Brassard, L. Salvail and J. Smolin, "Experimental Quantum Cryptography", *Journal of Cryptology*, vol. 5, no. 1, 1992, pp. 3 – 28.
3. Bennett, C. H. and G. Brassard, "Quantum Cryptography: Public Key Distribution and Coin Tossing", *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing*, Bangalore, India, December 1984, pp. 175 – 179.
4. Bennett, C. H., G. Brassard, C. Crépeau and M.-H. Skubiszewska, "Practical Quantum Oblivious Transfer", *Advances in Cryptology : CRYPTO '91 : Proceedings*, Lecture Notes in Computer Science, no 576, Springer–Verlag, August 1992, pp. 362 – 371.
5. Brassard, G., D. Chaum and C. Crépeau, "Minimum Disclosure Proofs of Knowledge", *Journal of Computer and System Sciences*, vol. 37 , 1988, pp. 156 – 189.
6. Damgård, I., personal communication, December 1998.
7. DiVincenzo, D.P., "Two-Bit Gates Are Universal For Quantum Computation", *Physical Review A*, vol. 51, no 2, February 1995, pp. 1015 – 1022.
8. Impagliazzo, R. and S. Rudich, "Limits on Provable Consequences of One-Way Permutations", *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, May 1989, pp. 44 – 61.
9. Lo, H.–K. and H.F. Chau, "Is quantum Bit Commitment Really Possible?", *Physical Review Letters*, vol. 78, no 17, April 1997, pp. 3410 – 3413.
10. Luby, M., *Pseudorandomness and Cryptographic Applications*, Princeton University Press, 1996.
11. Mayers, D., "The Trouble With Quantum Bit Commitment", `http://xxx.lanl.gov/abs/quant-ph/9603015`, March 1996.

12. Mayers, D., "Quantum Key Distribution and String Oblivious Transfer in Noisy Channels", *Advances in Cryptology : CRYPTO '96 : Proceedings*, Lecture Notes in Computer Science, no 1109, Springer–Verlag, August 1996, pp. 343 – 357.
13. Mayers, D., "Unconditionally Secure Quantum Bit Commitment is Impossible", *Physical Review Letters*, vol. 78, no 17, April 1997, pp. 3414 – 3417.
14. Mayers, D., "Unconditional Security in Quantum Cryptography", `http://xxx.lanl.gov/abs/quant-ph/9802025`, February 1998.
15. Naor, M., R. Ostrovsky, R. Ventkatesan and M. Young, "Perfect Zero-Knowledge Arguments For NP Using Any One-Way Permutation", *Journal of Cryptology*, vol. 11, no 2, 1998, pp. 87 – 108.
16. Salvail, L., "Quantum Bit Commitment From a Physical Assumption", *Advances in Cryptology : CRYPTO '98 : Proceedings*, Lecture Notes in Computer Science, no 1462, Springer–Verlag, August 1998, pp. 338 – 353.
17. Shor, P., "Algorithms for Quantum Computation: Discrete Logarithms and Factoring", *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, 1994, pp. 124 – 134.
18. van de Graaf, J., *Torwards a Formal Definition of Security for Quantum Protocols*, Ph.D. thesis, Département d'informatique et de recherche opérationelle, Université de Montréal, 1997.
19. Yao, A. C.–C., "Security of Quantum Protocols Against Coherent Measurements", *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, May 1995, pp. 67 – 75.

General Secure Multi-Party Computation from any Linear Secret-Sharing Scheme

Ronald Cramer*, Ivan Damgård**, and Ueli Maurer***

Abstract. We show that verifiable secret sharing (VSS) and secure multi-party computation (MPC) among a set of n players can efficiently be based on *any* linear secret sharing scheme (LSSS) for the players, provided that the access structure of the LSSS allows MPC or VSS at all. Because an LSSS neither guarantees reconstructability when some shares are false, nor verifiability of a shared value, nor allows for the multiplication of shared values, an LSSS is an apparently much weaker primitive than VSS or MPC.

Our approach to secure MPC is generic and applies to both the information-theoretic and the cryptographic setting. The construction is based on 1) a formalization of the special multiplicative property of an LSSS that is needed to perform a multiplication on shared values, 2) an efficient generic construction to obtain from any LSSS a multiplicative LSSS for the same access structure, and 3) an efficient generic construction to build verifiability into every LSSS (always assuming that the adversary structure allows for MPC or VSS at all).

The protocols are efficient. In contrast to all previous information-theoretically secure protocols, the field size is not restricted (e.g, to be greater than n). Moreover, we exhibit adversary structures for which our protocols are polynomial in n while all previous approaches to MPC for non-threshold adversaries provably have super-polynomial complexity.

1 Introduction

Secure *multi-party computation* (MPC) can be defined as the problem of n players to compute an agreed function of their inputs in a secure way, where security means guaranteeing the correctness of the output as well as the privacy of the players' inputs, even when some players cheat. A key tool for secure MPC, interesting in its own right, is *verifiable secret sharing* (VSS): a dealer distributes a secret value s among the players, where the dealer and/or some of the players may be cheating. It is guaranteed that if the dealer is honest, then the cheaters obtain no information about s, and all honest players are later able to reconstruct s. Even if the dealer cheats, a unique such value s will be determined and is reconstructible without the cheaters' help.

* Aarhus University, BRICS (Basic Reseach in Computer Science, center of the Danish National Research Foundation), work done while employed at ETH Zurich. Email: cramer@brics.dk. Supported by the Swiss SNF, grant no. 5003-045293.

** Aarhus University, BRICS. Email: ivan@daimi.aau.dk

*** ETH Zurich. Email: maurer@inf.ethz.ch. Supported by the Swiss SNF.

It is common to model cheating by considering an *adversary* who may corrupt some subset of the players. One can distinguish between passive and active corruption. *Passive* corruption means that the adversary obtains the complete information held by the corrupted players, but the players execute the protocol correctly. *Active* corruption means that the adversary takes full control of the corrupted players. It is (at least initially) unknown to the honest players which subset of players is corrupted. Trivially, secure MPC is impossible if any subset can be corrupted. The adversary's corruption capability is characterized by an *adversary structure* [25] $\mathcal{A}$, a family of subsets where the adversary can corrupt any subset in $\mathcal{A}$. This is called an $\mathcal{A}$-adversary. The adversary structure could for instance consist of all subsets with cardinality less than some threshold value t. Of course, an adversary structure must be monotone, i.e. if $A \in \mathcal{A}$ and $B \subset A$, then $B \in \mathcal{A}$.

Both passive and active adversaries may be *static*, meaning that the set of corrupted players is chosen once and for all before the protocol starts, or *adaptive* meaning that the adversary can at any time during the protocol choose to corrupt a new player based on all the information he has at the time, as long as the total corrupted set is in $\mathcal{A}$.

Two basic models of communication have been considered in the literature. In the *cryptographic* model [24], all players are assumed to have access to messages exchanged between players, and hence security can only be guaranteed in a cryptographic sense, i.e. assuming that the adversary cannot solve some computational problem. In the *information-theoretic* (abbreviated i.t., sometimes also called secure channels) model [5, 10], it is assumed that the players can communicate over pairwise secure channels, and security can then be guaranteed even when the adversary has unbounded computing power.

An MPC protocol simulates an ideal setting in which the players give their inputs to a trusted party who computes the result and gives it back to the players. Security means that whatever an adversary can do in the real protocol he could essentially also do in the ideal setting. This assures both privacy and correctness. There are several technically different proposals for formalizing this (see e.g. [1, 28, 8]). While either definition could be used for a formal security proof of the protocols in this paper, any such proof would by far exceed the space limitations. Instead, we include sketches of proofs, generic enough to fit any of the definitions.

The outline of the paper is as follows: In Section 2 we review come previous work. In Section 3 we introduce some terminology and concepts, state the results and explain the role they play in comparison with earlier results. The technical results on LSSS are proved in Section 4. The protocols we propose are described in sections 5, 6 and 7.

2 Previous Work

The classical MPC results in the information-theoretic model due to Ben-Or, Goldwasser and Wigderson [5] and Chaum, Crépeau and Damgård [10] who

showed that every function can be securely computed in presence of an adaptive, passive adversary, resp. an adaptive, active adversary if and only if the adversary corrupts less than $n/2$, resp. less than $n/3$ players. When a broadcast channel is available, and one accepts a non-zero probability that the protocol computes incorrect results, then one can tolerate less than $n/2$ active cheaters [30, 29].

The most general previous results for the cryptographic model are by Goldreich, Micali and Wigderson [24] who showed that, assuming trapdoor one-way permutations exist, any function can be securely computed in presence of a static, active adversary corrupting less than $n/2$ players and by Canetti et al. who show [9] that security against adaptive adversaries in the cryptographic model can also be obtained. VSS was introduced in [11].

All results mentioned so far only apply to threshold adversary structures. Gennaro [22] considered VSS in a non-threshold setting, and Hirt and Maurer [25] introduced the concept of an adversary structure and characterized exactly for which adversary structures VSS and secure MPC is possible. Let Q^2, resp. Q^3 be the conditions on an adversary structure that *no two*, resp. *no three* of the sets in the structure cover the full player set $\mathcal{P}$. The result of [25] can then be stated as follows: In the information-theoretic scenario, every function can be securely computed in presence of an adaptive, passive $\mathcal{A}$-adversary, resp. an adaptive, active $\mathcal{A}$-adversary if and only if $\mathcal{A}$ is Q^2, resp. $\mathcal{A}$ is Q^3. Beaver and Wool [2] propose a somewhat more efficient protocol for the passive case. The threshold results of [5], [10], [24] and [29] are special cases, where the adversary structure contains all sets of size less than $n/2$ or $n/3$.

This general model leads to strictly stronger results. For instance, in the case of 6 players $\{P_1, \ldots, P_6\}$ and active corruption, one can obtain a protocol secure against the structure with maximal sets $\{\{P_1\}, \{P_2, P_4\}, \{P_2, P_5, P_6\}, \{P_3, P_5\}, \{P_3, P_6\}, \{P_4, P_5, P_6\}\}$, whereas threshold type results tolerate only active cheating by a single player.

3 Results of this Paper

In this paper, we consider linear secret sharing schemes (LSSS). An LSSS is defined over a finite field K, and the secret to be distributed is an element in K. Each player receives from the dealer a share consisting of one or more field elements, each share is computed as a fixed linear function of the secret and some random field elements chosen by the dealer [1]. The *size* of an LSSS is the total number of field elements distributed. Only certain subsets of players, the *qualified* sets, can reconstruct the secret from their shares. *Unqualified* sets have no information about the secret. The collection of qualified sets is called the *access structure* of the LSSS, and the collection of unqualified sets is called the *adversary structure*.

Most proposed secret sharing schemes are linear, but the concept of an LSSS was first considered in its full generality by Karchmer and Wigderson who in-

[1] A seemingly weaker definition requires only that the reconstruction process be linear, however, this is essentially equivalent to the definition given here [3].

troduced the equivalent notion of Monotone Span Programs (MSP) which we describe in detail later. MSP's and LSSS's are in natural 1-1 correspondence.

The main goal in our paper is to provide an efficient construction which from any LSSS with adversary structure $\mathcal{A}$ builds MPC and VSS protocols secure against $\mathcal{A}$-adversaries (whenever this is possible). There are several motivations for this. First, basing VSS and MPC on as simple and weak a primitive as possible can help us design simpler and more efficient protocols because it is easier to come up with an implementation of a simpler primitive. Indeed, a wide range of general techniques for designing secret sharing schemes are known, e.g., Shamir [31], Benaloh-Leichter [4], Ito et al. [26], Bertilsson and Ingemarsson [6], Brickell [7] and van Dijk [16]. All these techniques result in LSSS's, and therefore are directly applicable to VSS and MPC by our results. Secondly, since LSSS's can be designed for any adversary structure, our approach allows us to build protocols handling any adversary structure for which VSS and MPC is possible at all. For some adversary structures this provably leads to an exponentially large efficiency improvement over known techniques, as we shall see.

We first give a brief overview of our basic approach: consider first the case where the adversary is passive. It is then trivial to add secrets securely: Each player holding an input shares it using the given LSSS, and each player adds up the shares he holds. By linearity of the LSSS, this results in a set of shares of the desired result.

Therefore, to do general MPC, it will suffice to implement multiplication of shared secrets. That is, we need a protocol where each player initially holds shares of secrets a and b, and ends up holding a share of ab. Such protocols are described for the threshold case in [24, 5, 10] and more recently in [23], based on Shamir's secret sharing scheme. We show below that the latter generalizes to work for any LSSS, provided that the LSSS is what we call *multiplicative*.

Loosely speaking, an LSSS is multiplicative if each player P_i can, from his shares of secrets a and b, compute a value c_i, such that the product ab can be computed as a linear combination of all the c_i's. It is *strongly* multiplicative if ab can be obtained using only values from honest players (we give a precise definition later).

With these techniques, using a multiplicative LSSS to implement passively secure MPC is quite straightforward. However, the multiplication property seems to require a very special structure in the LSSS. Nevertheless we show, perhaps somewhat surprisingly, that multiplicativity can be assumed without loss of generality: we give an efficient procedure that transforms any LSSS into a multiplicative LSSS of size at most twice that of the original one.

Finally, we consider the case of an active adversary. Basically, the same techniques as for the passive case will apply, provided we can build a linear *verifiable* secret sharing scheme from any given LSSS. We show that this can be done given a commitment scheme with certain convenient homomorphic properties. And we then build such a commitment scheme based also on the LSSS. With this VSS and the techniques described earlier for multiplication, an MPC protocol for active adversaries follows easily.

Thus, for the i.t. scenario, our main results are as follows:

THEOREM 1 *For any field K and any LSSS $\mathcal{M}$ with Q^3 adversary structure $\mathcal{A}$, there exists an error-free VSS protocol in the information-theoretic scenario, secure against any active and adaptive $\mathcal{A}$-adversary. The protocol has complexity polynomial in the size of $\mathcal{M}$ and* $\log|K|$.

THEOREM 2 *For any field K, any arithmetic circuit C over K, and any LSSS $\mathcal{M}$ with Q^2 adversary structure $\mathcal{A}$, there is an error-free MPC protocol computing C in the information theoretic scenario, secure against any adaptive and passive $\mathcal{A}$-adversary. The complexity of the protocol is polynomial in $|C|$, $\log|K|$, and the size of $\mathcal{M}$.*

THEOREM 3 *For any field K, any arithmetic circuit C over K, and any LSSS $\mathcal{M}$ with Q^3 adversary structure $\mathcal{A}$, there is an MPC protocol computing C in the information-theoretic scenario, secure against any adaptive and active $\mathcal{A}$-adversary. The complexity of the protocol is polynomial in $|C|$, $\log|K|$, the size of $\mathcal{M}$ and a security parameter k, where the error probability is exponentially small in k. If $\mathcal{M}$ is strongly multiplicative, there exists an error-free protocol for the same purpose, with complexity polynomial in $|C|$, $\log|K|$ and the size of $\mathcal{M}$.*

The statement of these results shows what can be done starting from a given LSSS. In practice, it may be that an adversary structure $\mathcal{A}$ is given by the application, and one wants the most efficient VSS or MPC possible for that structure. Our results show that we can build such protocols starting from any LSSS with a Q^3 (or Q^2) adversary structure containing $\mathcal{A}$. Such an LSSS always exists, by the results from Section 4. This leads naturally to a complexity measure for adversary structures, namely the size of the smallest LSSS that will work in this construction. From this perspective, our results show that the complexity of doing VSS/MPC secure for adversary structure $\mathcal{A}$ is upper bounded by the LSSS complexity of $\mathcal{A}$, up to a reduction polynomial in the number of players.

To compare our results to those of [25, 2] in terms of efficiency, we note that simple inspection of the protocols show that ours are more efficient by an additive polynomial amount for any non-threshold adversary structure. Moreover, the improvement can be much larger in some cases: we can show that there exists a family $\{\mathcal{A}_n\}_{n=1,2\ldots}$ of adversary structures (where $\mathcal{A}_n$ is a structure on n players) for which our results lead to protocols that are polynomial time in n whereas any construction based on [25] or [2] has super-polynomial complexity.

The proof of this result has been omitted for lack of space (but can be found in [32]). As an illustration, we describe a natural example of a family of structures, for which no previous solutions is known to work efficiently but for which linear size LSSS's can be built easily.

Suppose our player set is divided into two groups X and Y of m players each ($n = 2m$) where the players are on friendly terms within each group but tend to distrust players in the other group. Hence, a coalition of active cheaters might consist of almost all players from X or from Y, whereas a mixed coalition with players from both groups is likely to be quite small. Concretely, suppose we

assume that a group of active cheaters can consist of at most $9m/10$ players from only X or only Y, *or* it can consist of less than $m/5$ players coming from both X and Y. This defines a Q^3 adversary structure, and so multi-party computations are possible in this scenario. Nevertheless, no threshold solution exists, since the largest coalitions of corrupt players have size more than $n/3$. It can be shown that no weighted threshold solution exists either for this scenario.

Note that it is trivial to give linear size monotone formulæ characterizing these structures (when arbitrary threshold functions are allowed as operators), and hence efficient LSSS's for these structures follow immediately by results from [4]. Therefore, our techniques can be used to build efficient MPC in these scenarios. No efficient construction is known using the protocols from [25, 2].

It is natural to ask if the results can be improved, i.e., can we base VSS/MPC on a even weaker primitive, for example an arbitrary secret sharing (SS) scheme? This would be the best we could hope for since VSS trivially implies SS. Recently, Cramer, Damgård and Dziembowski [13] have shown that while VSS can indeed be based on arbitrary SS schemes (by an efficient black-box reduction), there exists no black-box reduction reducing MPC to SS that is efficient on all relevant adversary structures. Thus, any generally efficient reduction would have to rely on special properties of the SS scheme, such as linearity. Hence, improving our MPC results in this direction seems like a highly non-trivial problem.

Remarkably, the situation for the cryptographic scenario is quite different. We have the following generalization of the threshold result from [24] (where the complexity of an SS scheme is defined as the complexity of distributing and reconstructing a secret) [2]:

THEOREM 4 *Let C be an arithmetic circuit over a finite field K, let $\mathcal{A}$ be a Q^2 adversary structure, and let $\mathcal{S}$ be an SS scheme over K for which all sets in $\mathcal{A}$ are non-qualified and all complements of sets in $\mathcal{A}$ are qualified. If trapdoor one-way permutations exist, then there exists a secure MPC protocol computing C in the cryptographic scenario, secure against any active and static $\mathcal{A}$-adversary. It has complexity polynomial in $|C|$, the complexity of $\mathcal{S}$, and the security parameter k.*

The assumptions in this result are essentially minimal, but it does not lead to very practical protocols. However, if $\mathcal{S}$ is an LSSS and one-way group homomorphisms with specific extra properties exist, so-called q-one-way group homomorphisms [12], then very efficient protocols can be built. Particular assumptions sufficient for the existence of q-one way group homomorphisms include the RSA assumption, hardness of discrete logarithm in a group of prime order, or the decisional Diffie-Hellman assumption. As an example of what one obtains for the most efficient implementation of the primitives, we state the following:

THEOREM 5 *Let C be an arithmetic circuit over $K = GF(q)$ for a k-bit prime q, and let $\mathcal{A}$ be a Q^2 adversary structure. If Diffie-Hellmann based probabilistic encryption in a group of order q is semantically secure, then there exists an MPC*

[2] The proofs of the results in the cryptographic setting have been omitted for lack of space, they can be found in [32].

protocol computing C for the cryptographic scenario secure against any active and static $\mathcal{A}$-adversary. It has communication complexity $O(k \cdot |C| (\mu_{GF(q)}(\mathcal{A}))^2)$.

4 Multiplicative Monotone Span Programs

As mentioned earlier, Monotone Span Programs (MSP) are essentially equivalent to LSSS's (see e.g. [3]). It turns out to be convenient to describe our protocols in terms of MSP's, which we do for the rest of the paper. This section contains some basic definitions, notation and results relating to MSP's.

We first fix some notation for later use: The set of players in our protocols will be denoted by $\mathcal{P} = \{P_1, ..., P_n\}$. Consider any monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$. By identifying subsets of $\mathcal{P}$ with their characteristic vectors[3], we can also apply f to subsets of $\mathcal{P}$. A set $S \subset \mathcal{P}$ for which $f(S) = 0$ (or $f(S) = 1$) is said to be *rejected* (or *accepted*) by f. The function f hence defines naturally an adversary structure, denoted $\mathcal{A}_f$, consisting of the sets rejected by f. Conversely, an adversary structure $\mathcal{A}$ defines a monotone function $f_{\mathcal{A}}$ rejecting exactly the sets in $\mathcal{A}$.

For two vectors $\mathbf{x}$ and $\mathbf{y}$ over a field K, $\mathbf{x} \otimes \mathbf{y}$ denotes the matrix whose i-th column is $x_i\mathbf{y}$, where x_i is the i-th coordinate of $\mathbf{x}$. If $\mathbf{x}$ and $\mathbf{y}$ have the same length, then $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the standard scalar product. A $d \times e$ matrix M defines a linear map from K^e to K^d. Im M denotes the image of this map, i.e. the subspace of K^d spanned by the columns of M. Ker M denotes the kernel of M, i.e. Ker $M = \{\mathbf{x} \in K^e : M\mathbf{x} = \mathbf{0}\}$. For the subspace V of a finite-dimensional vector space over K, the dual $V^\perp$ is defined as the subspace of vectors whose scalar product is 0 with all vectors in V. It is a basic fact from linear algebra that for any field K, $(V^\perp)^\perp = V$ and this immediately implies that (Ker $M)^\perp =$ Im (M^T), which we refer to as *duality argument.*

DEFINITION 1 *A Monotone Span Program $\mathcal{M}$ is a triple (K, M, ψ), where K is a finite field, M is a matrix (with d rows and $e \leq d$ columns) over K and $\psi : \{1, \ldots, d\} \longrightarrow \{1, \ldots, n\}$ is a surjective function. The* size *of $\mathcal{M}$ is the number of rows (d).*

ψ labels each row with a number from $[1, \ldots, n]$ corresponding to a player in $\mathcal{P}$, so we can think of each player as being the "owner" of one or more rows.

In the following, if M is the matrix of an MSP, and A is a subset of the players, then M_A denotes M restricted to those rows i with $\psi(i) \in A$. Similarly, if $\mathbf{x}$ is a d-vector, then $\mathbf{x}_A$ denotes $\mathbf{x}$ restricted to the coordinates i with $\psi(i) \in A$.

$\mathcal{M}$ yields a linear secret sharing scheme as follows: to distribute $s \in K$ the dealer chooses a random vector $\boldsymbol{\rho} \in K^{e-1}$ and writes $\mathbf{b} := (s, \boldsymbol{\rho})$. For each row $\mathbf{x}$ in M, the scalar product $\langle \mathbf{x}, \mathbf{b} \rangle$ is given to the owner of $\mathbf{x}$. We will denote the d-vector thus distributed by $M(s, \boldsymbol{\rho})$. It turns out that a set of players can reconstruct s precisely if the rows they own contain in their linear span the *target*

[3] The characteristic vector of a set S is a vector in $\{0,1\}^n$ whose i-th component is 1 if and only if $P_i \in S$.

vector of $\mathcal{M}$ which we have here globally fixed to be $(1, 0, ..., 0)$ (without loss of generality). Otherwise they get no information on s (see [27] for a proof of this). We note that the size of $\mathcal{M}$ is also the size of the corresponding LSSS.

The function computed by $\mathcal{M}$ is the monotone function accepting precisely those subsets that can reconstruct the secret [27]. It is well-known that for every field K, every monotone Boolean function is computed by some MSP over K. For a monotone function f, $\mathrm{msp}_K(f)$ will denote the size of the smallest MSP over K computing f. We refer to [19] for a characterization of MSP complexity in terms of certain combinatorial structures.

We now look at doing multiplication of values shared using MSP's. If secrets a and b have been shared using Shamir's secret sharing scheme to obtain shares $(a_1, .., a_n)$ and $(b_1, ..., b_n)$, respectively, it is immediate (see [23]) that ab can be computed as a linear combination of the values $a_i b_i$, where each such value can be computed locally by a single player. This can be generalized to LSSS's based on MSP's:

Given two d-vectors $\mathbf{x} = (x_1, ...x_d), \mathbf{y} = (y_1, ..., y_d)$, we let $\mathbf{x} \diamond \mathbf{y}$ be the vector containing all entries of form $x_i \cdot y_j$, where $\psi(i) = \psi(j)$. Thus, if d_i is the number of rows owned by player i, then $\mathbf{x} \diamond \mathbf{y}$ has $m = \sum_i d_i^2$ entries. Hence if $\mathbf{x}$ and $\mathbf{y}$ are the share-vectors resulting from sharing two secrets using $\mathcal{M}$, then each component of the vector $\mathbf{x} \diamond \mathbf{y}$ can be computed locally by some player.

DEFINITION 2 *A* multiplicative *MSP is an MSP $\mathcal{M}$ for which there exists an m-vector $\mathbf{r}$, called a* recombination vector*, such that for any two secrets s, s' and any $\boldsymbol{\rho}$, $\boldsymbol{\rho}'$, it holds that*

$$s \cdot s' = \langle \mathbf{r}, M(s, \boldsymbol{\rho}) \diamond M(s', \boldsymbol{\rho}') \rangle.$$

We say that $\mathcal{M}$ is strongly multiplicative *if for any player subset A that is rejected by $\mathcal{M}$, $\mathcal{M}_{\overline{A}}$ is multiplicative.*

The case of strong multiplication generalizes the threshold case with at most t corrupted players where we share secrets using polynomials of degree $t < n/3$. After multiplying points on two polynomials, the honest players can reconstruct the product polynomial on their own.

We define $\mu_K(f)$ to be the size of the smallest multiplicative MSP over K with computing f (∞ if f cannot be computed). Similarly, $\mu_K^*(f)$ is the complexity of f using strongly multiplicative MSP's. By definition, we have $\mathrm{msp}_K(f) \leq \mu_K(f) \leq \mu_K^*(f)$. We now characterize the functions that (strongly) multiplicative MSP's can compute, and show that the multiplication property for an MSP can be assumed without loss of efficiency.

THEOREM 6 *For every finite field K and every monotone function f we have $\mu_K(f) < \infty$ if and only if f is Q^2, and $\mu_K^*(f) < \infty$ if and only if f is Q^3.*

THEOREM 7 *There exists an efficient algorithm which, on input an MSP $\mathcal{M}$ computing a Q^2 function f, outputs a multiplicative MSP $\mathcal{M}'$ (over the same field) computing f and of size at most twice that of $\mathcal{M}$. In particular $\mu_K(f) \leq 2 \cdot msp_K(f)$ for any K and f.*

We do not know if a similar result is true for strongly multiplicative MSP's. But results which can be found in [32] show some upper bounds on their size, and give methods for constructing strongly multiplicative MSP's.

PROOF OF THEOREM 7. We make some observations first. Let f_0 and f_1 be monotone functions, computed by MSP's $\mathcal{M}_0 = (K, M_0, \psi)$ and $\mathcal{M}_1 = (K, M_1, \psi)$, respectively, where M_0 and M_1 are $d \times e$ matrices, where the mapping ψ is identical for both MSPs, and where the target vector is $\boldsymbol{t} = (1, 0, \ldots, 0)$. Now suppose that the matrices M_0 and M_1 satisfy

$$M_0^T M_1 = E, \tag{1}$$

where E is $e_o \times e_1$ matrix that is zero everywhere, except in its upper-left corner where the entry is 1. Next we prove the following claim.

CLAIM: From MSP's $\mathcal{M}_0$ and $\mathcal{M}_1$ as defined above, one can construct a multiplicative MSP computing $f_0 \vee f_1$ of size at most $2d$.

PROOF OF CLAIM: Consider the following straightforward LSSS. The dealer shares the secret $s \in K$ using LSSS_0 *and* LSSS_1, given by $\mathcal{M}_0$ and $\mathcal{M}_1$, respectively. That is, he selects a pair of vectors $(\mathbf{b}_0, \mathbf{b}_1)$ at random, except that the first entries are both s: $\langle \boldsymbol{t}, \mathbf{b}_0 \rangle = \langle \boldsymbol{t}, \mathbf{b}_1 \rangle = s$. Then he computes the pair of vectors $(\mathbf{s}_0, \mathbf{s}_1) = (M_0\mathbf{b}_0, M_1\mathbf{b}_1)$, and sends for $i = 1, \ldots, n$ the i-th coordinates of $\mathbf{s}_0$ and $\mathbf{s}_1$ to player $P_{\psi(i)}$. It is clear that a subset A of the players can reconstruct s from their joint shares if and only if A is accepted by the function $f_0 \vee f_1$, i.e. A must be qualified w.r.t. either LSSS_0 or LSSS_1.

Now we look at multiplication. Assume that $s' \in K$ is a secret with full set of shares $(\mathbf{s}_0', \mathbf{s}_1') = (M_0\mathbf{b}_0', M_1\mathbf{b}_1')$, where $\langle \boldsymbol{t}, \mathbf{b}_0' \rangle = \langle \boldsymbol{t}, \mathbf{b}_1' \rangle = s'$. Let $\mathbf{s}_0 * \mathbf{s}_1'$ be the d-vector obtained by coordinate-wise multiplication of $\mathbf{s}_0$ and $\mathbf{s}_1'$. Then from (1) we have

$$\langle \mathbf{1}, \mathbf{s}_0 * \mathbf{s}_1' \rangle = \mathbf{s}_0^T \mathbf{s}_1' = \mathbf{b}_0^T M_0^T M_1 \mathbf{b}_1' = \mathbf{b}_0^T E \mathbf{b}_1' = ss', \tag{2}$$

where $\mathbf{1}$ denotes the all-one vector of appropriate length. Note that for each i, the shares in the i-th coordinate of $\mathbf{s}_0$ and the i-th coordinate of $\mathbf{s}_1'$ are held by the same player.

We now build an MSP $\mathcal{M}$ with a $2d$ by $2e$ matrix M as follows: first make a matrix M' filled in with M_0 in the upper left corner and M_1 in the lower right corner. Let $\mathbf{k}$ be the column in M' that passes through the first column of M_1. Add $\mathbf{k}$ to the first column of M' and delete $\mathbf{k}$ from the matrix. Let M be the result of this. The labeling of M is carried over in the natural way from $\mathcal{M}_0$ and $\mathcal{M}_1$. Clearly, $\mathcal{M}$ corresponds exactly to the LSSS we just constructed. It is clear that the vector $(\mathbf{s}_0, \mathbf{s}_1) \diamond (\mathbf{s}_0', \mathbf{s}_1')$ contains among its entries the entries of $\mathbf{s}_0 * \mathbf{s}_1'$. Thus the vector with 1's corresponding to these entries and 0's elsewhere can be used as recombination vector, which shows that $\mathcal{M}$ is multiplicative. This concludes the proof of the claim.

We are now ready to prove Theorem 7. Recall that the dual function f^* of f is defined by: $f^*(x) = \overline{f(\overline{x})}$. We assume in the following that f is Q^2, i.e. $f(x) = 0$ implies $f(\overline{x}) = 1$ and thus $f^*(x) = 0$. It follows that $f = f \vee f^*$.

Let $\mathcal{M} = (K, M, \psi)$ be an MSP computing f, with target vector equal to $\boldsymbol{t} = (1, 0, \ldots, 0)$. To build a multiplicative MSP for f, we apply the above claim. We set $f_0 = f$, $f_1 = f^*$ and $\mathcal{M}_0 = \mathcal{M}$. It is then sufficient to find $\mathcal{M}_1$ computing $f_1 = f^*$ so that the pair M_0, M_1 satisfies equation (1).

In [20] a construction is presented which, given an MSP $\mathcal{N} = (K, N, \psi)$ of size d computing f (and with target vector $(1, \ldots, 1)$), yields a "dual" MSP $\mathcal{N}^* = (K, N^*, \psi)$ computing f^* (also with target vector $(1, \ldots, 1)$). The construction is as follows. N^* has also d rows and the same labeling as N and consists of one column for each set A accepted by f, namely a (reconstruction) vector $\boldsymbol{\lambda}$ satisfying $\boldsymbol{\lambda}^T N = (1, \ldots, 1)$ and $\boldsymbol{\lambda}_{\overline{A}} = \mathbf{0}$. The matrix N^* has generally exponentially many columns, but it is easy to see that any linearly independent generating subset of them (at most d) will also constitute a matrix of an MSP for the same access structure. This construction process if used directly is not efficient, but the matrix N^* can be constructed efficiently, without enumerating all columns [17]. It follows from the construction that $N^T N^*$ is an all-one matrix, which we call U.

In our case the target vector of $\mathcal{M}$ is $\boldsymbol{t} = (1, 0, \ldots, 0)$ instead of $(1, \ldots, 1)$, but the target vector can be transformed by adding the first column of M to every other column of M. More formally, let H be the isomorphism that sends an e-(column) vector to an e-(column) vector by adding its first coordinate to each other coordinate. Write $N = MH^T$. Then the MSP $\mathcal{N} = (K, N, \psi)$ is as $\mathcal{M}$ except that the target vector is all-one. Now let $\mathcal{N}^* = (K, N^*, \psi)$ be its dual MSP as constructed above. Finally write $M^* = N^*(H^{-1})^T$. Then $\mathcal{M}^* = (K, M^*, \psi)$ has target vector $\boldsymbol{t}$ and computes f^*. Observe that $M^T M^* = H^{-1} U (H^{-1})^T = E$, as desired. Theorem 7 follows. $\triangle$

Proof of Theorem 6. Since MSP's compute all monotone functions, it follows directly from this fact and Theorem 7 that every Q^2-function is computed by a multiplicative MSP. This also follows from secret sharing scheme used in [2], and this argument can be extended to prove that every Q^3-function is computed by a strongly multiplicative MSP. We conclude the proof by showing that multiplicative MSP's compute Q^2-functions. The proof in the Q^3-case is similar, and is omitted.

Let $\mathcal{M} = (K, M, \psi)$ be an MSP with target vector $\boldsymbol{t}$ computing a monotone boolean function f on n input bits, and let $\mathcal{A}_f$ be the adversary structure associated with f. Suppose that $\mathcal{M}$ is multiplicative, but that $\mathcal{A}_f$ is *not* Q^2. Thus, there exists a set $A \subset \{1, \ldots, n\}$ such that $A \cup \overline{A} = \{1, \ldots, n\}$ and $A, \overline{A} \in \mathcal{A}_f$. The latter implies that neither the rows of M_A nor those of $M_{\overline{A}}$ span $\boldsymbol{t}$. Hence, by the duality argument there exist vectors $\boldsymbol{\kappa}$ and $\boldsymbol{\kappa}'$, both with first coordinate equal to 1, such that $M_A \boldsymbol{\kappa} = \mathbf{0}$ and $M_{\overline{A}} \boldsymbol{\kappa}' = \mathbf{0}$. By the multiplication property, on one hand it follows that $\langle \mathbf{r}, M\boldsymbol{\kappa} \diamond M\boldsymbol{\kappa}' \rangle = 1$, where $\mathbf{r}$ is the recombination vector. But on the other hand, $M\boldsymbol{\kappa} \diamond M\boldsymbol{\kappa}' = \mathbf{0}$, by the choice of $\boldsymbol{\kappa}$, $\boldsymbol{\kappa}'$, and the fact that $A \cup \overline{A} = \{1, \ldots, n\}$, so this scalar product must be equal to 0: a contradiction. $\triangle$

5 Homomorphic Commitments and VSS

5.1 Preliminaries

We introduce some conventions and notation to be used in the protocol descriptions throughout the rest of the paper. We assume throughout (without loss of generality, and in accordance with the previous literature) that the function to be computed by $\{P_1, \ldots, P_n\}$ is given as an *arithmetic circuit* C of size $|C|$ over some finite field K, consisting of addition and multiplication gates. Our protocols are described making use of a broadcast channel. But note that in the i.t. scenario with an active adversary, we do *not* assume that such a channel is given for free as part of the model, however it can efficiently be simulated using the protocol of [18] that is secure against any given Q^3 adversary structure.

Let $\mathcal{M}$ be an MSP computing a Q^2 (or Q^3) function f. We will assume for simplicity that ψ is $1-1$, i.e. *each player owns exactly one row in* $\mathcal{M}$. In this case, $(a_1, ..., a_n) \diamond (b_1, ..., b_n) = (a_1 b_1, ..., a_n b_n)$. The generalization to many rows per player is straightforward, but would lead to rather complicated notation.

5.2 Overview of Commitments and Related Protocols

To prove Theorem 1, it is sufficient to construct, for each MSP $\mathcal{M} = (K, M, \psi)$ computing a Q^3 function f, an efficient VSS that is secure against an active $\mathcal{A}_f$-adversary. We first discuss generic primitives sufficient to construct an efficient VSS protocol and conclude by providing concrete realizations of these primitives.

A *commitment scheme* (for a given adversary structure $\mathcal{A}$) consists of two protocols: the protocol COMMIT allows a player P_i (the dealer) to commit to a value a and the protocol OPEN allows him later to reveal the committed value. The total information stored by the players after the protocol COMMIT is called the *commitment* and will be denoted $[a]_i$. Both protocols may be interactive protocols among the players and result either in the players accepting the outcome, or disqualifying the dealer. A commitment scheme must hide the committed value in the presence of an $\mathcal{A}$-adversary, and it must bind the dealer to the committed value, i.e. there is at most one value that the dealer can get accepted during the OPEN protocol. Both these properties can hold unconditionally, or relative to a computational assumption, depending on the scenario.

The crucial property we need is that commitments are *homomorphic*, which means that from commitments $[a]_i$ and $[b]_i$ the players can compute *without interaction* a commitment $[a+b]_i$ by P_i to the sum of the values, and that for a constant m they can compute $[ma]_i$. Thus, any linear function on committed values can be computed non-interactively. Homomorphic commitments have been used before in the context of zero-knowledge (e.g. [12]) and are implicit in some MPC protocols (e.g. [10]). We need two auxiliary protocols:

- A *commitment transfer protocol* (CTP) allows player P_i to transfer a commitment to player P_j (who of course learns a in the process), i.e. to convert $[a]_i$ into $[a]_j$. It must be guaranteed that this protocol leaks no information

to the adversary if P_i and P_j are honest throughout the protocol, but also that the new commitment contains the same value as the old, even if P_i and P_j are both corrupt. It is therefore less trivial than one might expect.

- A *commitment sharing protocol* (CSP) allows player P_i to convert a committed value $[a]_i$ into a set of commitments to shares of a: $[a_1]_1, ..., [a_n]_n$, where $(a_1, ..., a_n) = M(a, \boldsymbol{\rho})$ for a random vector $\boldsymbol{\rho}$ chosen by P_i. This must hold even if P_i is corrupt, and must leak no information to the adversary if P_i is honest throughout the protocol.

The CSP protocol is easy to describe at a general level: starting from $[a]_i$, P_i chooses a random vector $(\rho_1, ..., \rho_{e-1})$ and commits to $\rho_1, ..., \rho_{e-1}$, resulting in $[\rho_1]_i, ..., [\rho_{e-1}]_i$. Let $(a_1, ..., a_n)$ be the shares resulting from sharing a using the ρ_i's as random choices. Each a_i is a linear function of the committed values, and hence the players can compute $[a_1]_i, ..., [a_n]_i$ non-interactively. Finally, P_i uses CTP to convert $[a_j]_j$ into $[a_j]_i$, for $j = 1, \ldots, n$.

Committing to a and then performing CSP is equivalent to verifiably secret sharing (VSS) of a: the commitments to shares prevent corrupted players from contributing false shares when the secret is reconstructed. It remains to give efficient realizations of commitments and the CTP.

5.3 Realization of Commitments

To have a player D commit to a one could have him secret share a using $\mathcal{M}$. However, D may be corrupt and so must be prevented from distributing inconsistent shares. In the special case of threshold secret sharing, this means ensuring that all uncorrupted players hold points on a polynomial of bounded degree. For this purpose, we propose a protocol that can be seen as a generalization of the BGW-protocol from [5] where a bivariate polynomial was used[4]:

1. To commit to $s \in K$, D chooses a symmetric $e \times e$ matrix R at random, except that R has s in the upper left corner.[5] Let $\mathbf{v}_i$ be the row in M assigned to P_i and let $\mathbf{v}_i^T$ be its transpose (a column vector). Then D sends to P_i the vector $\mathbf{u}_i = R \cdot \mathbf{v}_i^T$. The share s_i of s given to P_i is defined to be the first entry of $\mathbf{u}_i$. Hence the product $\langle \mathbf{v}_j, \mathbf{u}_i \rangle := s_{ij}$ can be thought of as a share of s_i given to P_j. Note that we have $\langle \mathbf{v}_j, \mathbf{u}_i \rangle = \langle \mathbf{v}_j R, \mathbf{v}_i^T \rangle = \langle \mathbf{v}_i, R\mathbf{v}_j^T \rangle = \langle \mathbf{v}_i, \mathbf{u}_j \rangle$.
2. P_i sends to each P_j the value $\langle \mathbf{v}_j, \mathbf{u}_i \rangle$, who compares this to $\langle \mathbf{v}_i, \mathbf{u}_j \rangle$ and broadcasts a message $complaint(i, j)$ if the values are not equal.
3. In response to $complaint(i, j)$, D must broadcast the correct value of s_{ij}.

[4] Apart from the threshold case [5], our protocol is a VSS, i.e. efficient reconstruction without the help of the dealer is possible, if for each set B whose complement is in $\mathcal{A}$, the matrix M_B has full rank. In this case, players should store all information received by the dealer to reconstruct efficiently. In general, however, we cannot guarantee efficient reconstruction, so we only use it here as a commitment scheme.

[5] One can think of step 1 in this protocol (choosing R) as corresponding to choosing a *symmetric* bivariate polynomial in the VSS protocol of [5].

4. If any player P_i finds that the information broadcast by D does not match what he received from D in step 1, he broadcasts an *accusation*, thus claiming that D is corrupt.
5. In response to an accusation by P_i, D must broadcast all information sent to P_i in step 1.
6. The information broadcast by D in the previous step may lead to further accusations. This process continues until the information broadcast by D contradicts itself, or he has been accused by a set of players not in $\mathcal{A}$, or no new complaints occur. In the first two cases, D is clearly corrupt and is disqualified. In the last case, the commit protocol is accepted by the honest players, and accusing players accept the share broadcast for them by D.

To open a commitment, D broadcasts s and the full set of shares $\{s_i\}$, and each player broadcasts a binary message ("agree" or "complain"). If the shares consistently determine s and only a set of players in $\mathcal{A}$ complained, then the opening is accepted.

We now explain why this commitment scheme works. First, assume D remains honest throughout the commit protocol. To show that the adversary obtains no information about s, note first that steps 2-6 of the commit protocol are designed such that the adversary learns nothing he was not already told in step 1. Now let A be any set in $\mathcal{A}$, and let $M_A R$ denote the information received by the players in A in the commit phase, finally let X be any symmetric matrix satisfying the equation $M_A X = M_A R$, and having some $\tilde{s} \in K$ in its upper-left corner. Since A is rejected by $\mathcal{M}$, it follows by the duality argument that there exists a vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_e) \in \mathrm{Ker} M_A$ with $\mu_1 = 1$. Consider the matrix $\boldsymbol{\mu} \otimes \boldsymbol{\mu}$. Note that this matrix is symmetric and that it has 1 in its upper-left corner. Then $X + (s - \tilde{s})\boldsymbol{\mu} \otimes \boldsymbol{\mu}$ satisfies the equation as well, has s in its upper left corner and is symmetric. Hence, for each possible $\tilde{s}$, the number of different solutions X with $\tilde{s}$ in the upper left corner is the same, and hence the adversary learns no information on s in step 1. Finally note that if D remains honest throughout, all honest players will agree with him, so the opening always succeeds.

Now assume that D is corrupt. We know that a set of players not in $\mathcal{A}$, i.e. large enough to uniquely determine a secret shared by $\mathcal{M}$, remain honest throughout the commit protocol (this is called a qualified set). Assume wlog that these are the first t players. The commit protocol ensures that if D is not disqualified then each pair of honest players agree on the value s_{ij} they have in common. Furthermore, if P_i is honest, all the s_{ij}'s known to him are consistent with $\mathbf{u}_i$. Define the symmetric $n \times n$ matrix S to be the matrix containing all the s_{ij}'s known to players $P_1, ..., P_t$ (this leaves entries s_{ij} with $i, j > t$ undefined). For $i \leq t$, the i'th column determines s_i uniquely, as a fixed linear combination of the first t entries (since the first t players form a qualified set). The coefficients in this linear combination depend only on $\mathcal{M}$ and so are the same for any column. It follows that the row of shares $(s_1, ..., s_n)$ is determined as a linear combination of the first t rows of S. Since each of these rows consistently determines a secret

(namely s_i for the i'th row), it follows by linearity of MSP secret sharing that the row $(s_1, ..., s_n)$ consistently determines some secret s.

It remains to be shown that opening the commitment must either reveal s or be rejected. Assume the opening is accepted. Then consider the full player set and subtract the set of corrupt players and the set of players who complained about the opening. The remaining set cannot be in $\mathcal{A}$ by the Q^3 property and so is qualified. It consists of honest players that did not complain, i.e. the shares revealed for them are the same as those received in the commitment phase. Hence the revealed value must be s.

5.4 Realization of the CTP

The following protocol converts $[a]_i$ into $[a]_j$:

1. Given a commitment $[a]_i$ realized as above with P_i in the role of D, P_i sends privately the shares determining a to P_j. If this information is not consistent, then P_j broadcasts a complaint, and the protocol continued at step 4.
2. P_j commits to a (independently), resulting in $[a]_j$.
3. Using linearity of commitments, P_j opens the difference $[a]_i - [a]_j$ to reveal 0, using the information from step 1 as if he created $[a]_i$ himself. If this succeeds, the protocol ends. Otherwise do Step 4.
4. If we arrive at this step, it is clear that at least one of P_i and P_j is corrupt, so P_i must then open $[a]_i$ in public, and we either disqualify P_i (if he fails) or continue with a default commitment to a assigned to P_j.

6 MPC Secure Against Passive Adversaries

To prove Theorem 2, it is sufficient to construct for each MSP $\mathcal{M} = (K, M, \psi)$ computing a Q^2 function f, an efficient protocol that is secure against a passive $\mathcal{A}_f$-adversary. By Theorem 7 we can assume without loss of generality (or efficiency) that $\mathcal{M}$ is multiplicative.

The protocol, which is a generalization of a threshold protocol appearing in [23], starts by letting each player share each of his inputs using $\mathcal{M}$ and send a share to each player. The given arithmetic circuit over K is then processed gate by gate, maintaining as invariant that all inputs and intermediate results are secret-shared, i.e. each such value $a \in K$ is shared (using $\mathcal{M}$) by shares $a_1, ..., a_n$, where P_i holds a_i. Moreover, if a depends on an input from an honest player, this must be a *random* set of shares with the only constraint that it determines a. At the beginning, only the input values are classified as having been computed. Once an output value x has been computed, it can be reconstructed in the obvious way by broadcasting the shares $x_1, ..., x_n$. It is therefore sufficient to show how addition and multiplication gates are handled. Assume the input values to a gate are a and b, determined by shares $a_1, ..., a_n$ and $b_1, ..., b_n$, respectively.

Addition For $i = 1, \ldots, n$, P_i computes $a_i + b_i$. The shares $a_1 + b_1, \ldots, a_n + b_n$ determine $a + b$ as required by the invariant.

Multiplication For $i = 1, \ldots, n$, P_i computes $a_i \cdot b_i = \tilde{c}_i$.
Resharing step: P_i secret shares $\tilde{c}_i$, resulting in shares $c_{i1}, ..., c_{in}$, and sends c_{ij} to player P_j.
Recombination step: For $j = 1, \ldots, n$, player P_j computes $c_j = \sum_{i=1}^{n} r_i c_{ij}$, where $(r_1, \ldots, r_n)$ is a fixed recombination vector of $\mathcal{M}$. The shares $c_1, \ldots, c_n$ determine $c = ab$ as required by the invariant.

We do not have space to prove formally the security of this protocol here. However, to get a feeling for why it is secure, note first that the addition and multiplication step compute correct results simply by linearity of the secret sharing, and by the multiplication property. To argue that privacy is maintained, the crucial point is to show that the sharing of a result c of the multiplication step starting from a, b is random with the only restriction that it determines c (the corresponding result for addition is trivial).

It is easily seen that the set of shares determining c can be written as $(c_1, ..., c_n) = M(c, \boldsymbol{\rho})$, where in fact $\boldsymbol{\rho} = \sum_{i=1}^{n} r_i \boldsymbol{\rho}_i$ and where $\boldsymbol{\rho}_i$ was chosen by P_i. Let $B = \{P_i |\ r_i \neq 0\}$. We claim that $B \notin \mathcal{A}$. Indeed, let $\mathcal{M}$ be an MSP with multiplication, and let $\mathbf{r}$ be a recombination vector. Then $B = \{P_i |\ r_i \neq 0\} \notin \mathcal{A}$. Towards a contradiction, suppose $B \in \mathcal{A}$. By the duality argument, choose $\boldsymbol{\kappa}$ such that $M_B \boldsymbol{\kappa} = \mathbf{0}$ and the first coordinate κ_1 of $\boldsymbol{\kappa}$ is 1. Then by definition of the multiplication property we have that $1 = \kappa_1^2 = \langle \mathbf{r}, M\boldsymbol{\kappa} \diamond M\boldsymbol{\kappa} \rangle$. But on the other hand, since $M_B \boldsymbol{\kappa} \diamond M_B \boldsymbol{\kappa} = \mathbf{0}$ and $\mathbf{r}_{\overline{B}} = \mathbf{0}$, this must be equal to 0, a contradiction. This proves the claim. Therefore, the choice of at least one $\boldsymbol{\rho}_i$ where $r_i \neq 0$ remains unknown to the adversary and is made randomly and independently of anything else. This can be used when building a simulator for an adversary: when he corrupts a player, what we have to do is essentially to come up with a random share for this player of each shared value. Each such share must be consistent with what the adversary already knows. By the above, this can be handled independently for each shared value, and so can be easily done by solving a system of linear equations.

7 MPC Secure Against Active Adversaries

To prove Theorem 3, it is sufficient to construct for each MSP $\mathcal{M} = (K, M, \psi)$ computing a Q^3 function f, an efficient protocol that is secure against an active $\mathcal{A}_f$-adversary. Since Q^3-functions are in particular Q^2, we can assume by Theorem 7 without loss of generality (or efficiency) that $\mathcal{M}$ is multiplicative.

Like in some previous protocols, the basic approach is to ensure that all players are committed to the values they hold, and to have them prove that they are performing their local operations correctly. In what follows, we use a generic commitment scheme and auxiliary protocols as in Section 5.

7.1 The CMP Protocol

We need an additional primitive, namely a *Commitment Multiplication Protocol* (CMP). Such a protocol starts from commitments $[a]_i, [b]_i, [c]_i$ and allows P_i

to convince the other players that $ab = c$. If P_i is corrupted, then the honest players should accept the proof only if $ab = c$ (in the cryptographic scenario, an negligible error probability is allowed). If P_i remains honest, it must leak no information to the adversary beyond the fact that $ab = c$. Moreover, in the event that $[c]_i$ is opened, the adversary must learn nothing about a, b beyond what is implied by c and the other information he holds. The following CMP protocol is a generalization of a protocol suggested in [14] and works for any homomorphic commitment scheme.

1. Inputs are commitments $[a]_i, [b]_i, [c]_i$ where P_i claims that $ab = c$. P_i chooses a random β and makes commitments $[\beta]_i, [\beta b]_i$.
2. The other players jointly generate a random challenge r using standard techniques.
3. P_i opens the commitments $r[a]_i + [\beta]_i$ to reveal a value r_1. P_i opens the commitment $r_1[b]_i - [\beta b]_i - r[c]_i$ to reveal 0.
4. If any of these opening fail, the proof is rejected, else it is accepted.

It is easy to show that if P_i is honest, then all values opened are random (or fixed to 0) and so reveal no extra information to the adversary. Furthermore, if after committing in step 2, P_i can answer correctly two different challenges, then $ab = c$. Thus the error probability is at most $1/|K|$, and the protocol can be iterated to reach any desired error probability. In [32], we show that an error-free CMP protocol can be built based on a strongly multiplicative MSP.

7.2 The General MPC Protocol

The general MPC protocol starts by asking each player to VSS each of his input values as described above: he commits to the value and then performs CSP. A player failing to execute this correctly is disqualified and we take default values for his inputs.

We then work our way through the given arithmetic circuit, maintaining as invariant that all inputs and intermediate results computed so far are VSS'ed as described above, i.e. each such value a is shared (using $\mathcal{M}$) by committed shares $[a_1]_1, ..., [a_n]_n$ where *all* these shares are correct, also those held by corrupted players. Moreover, if a depends on an input from an honest player, this must be a *random* set of shares with the only constraint that it determines a. At the beginning, only the input values are classified as having been computed.

Once an output value x has been computed, it can be reconstructed in the obvious way by opening commitments to the shares $x_1, ..., x_n$. This will succeed, as the honest players will contribute enough correct shares, and a corrupted player can only choose between contributing a correct share, or be disqualified by trying to open an incorrect value. It is therefore sufficient to show how addition and multiplication gates are handled. Assume the input values to a gate are a and b, determined by committed shares $[a_1]_1, ..., [a_n]_n$ and $[b_1]_1, ..., [b_n]_n$.

Addition For $i = 1, \ldots, n$, P_i computes $a_i + b_i$ and the players (non-interactively) compute $[a_i + b_i]_i$. By linearity of the secret sharing and commitments, $[a_1 + b_1]_1, ..., [a_n + b_n]_n$ determine $a + b$ as required by the invariant.

Multiplication For $i = 1..n$, P_i computes $a_i \cdot b_i = \tilde{c}_i$, commits to it, and performs CMP on inputs $[a_i]_i, [b_i]_i, [\tilde{c}_i]_i$.
Resharing step: P_i performs CSP on $[\tilde{c}_i]_i$, resulting in the commitments $[c_{i1}]_1, ..., [c_{in}]_n$. We describe below how to recover if P_i fails to execute this phase correctly.
Recombination step: For $j = 1..n$, player P_j computes $c_j = \sum_{i=1}^{n} r_i c_{ij}$, where $(r_1, ..., r_n)$ is a fixed recombination vector. Also all players compute (non-interactively) $[c_j]_j = \sum_{i=1}^{n} r_i [c_{ij}]_j = [\sum_{i=1}^{n} r_i c_{ij}]_j$. By the multiplication property and linearity of $\mathcal{M}$, the commitments $[c_1]_1, ..., [c_n]_n$ determine $c = ab$ as required by the invariant.

It remains to be described what should be done if a player P_i fails to execute the multiplication and resharing step above. In general, the simplest way to handle such failures is to go back to the start of the computation, open the input values of the players that have just been disqualified, and restart the computation, simulating openly the disqualified players. This allows the adversary to slow down the protocol by a factor at most linear in n. This protocol, together with the VSS and main MPC protocols described previously, are the basis for proving Theorem 3.

The described approach for dealing with cheaters can be used only for secure function evaluation, but not for an ongoing secure computation. For the latter, one can introduce an additional level of sharings: each value a player is committed to in the above description is shared again among the players, with each player being committed to his share.

Acknowledgments

We thank Serge Fehr, Mathias Fitzi, Anna Gál, Rosario Gennaro, Martin Hirt, Peter Bro Miltersen, and Tal Rabin for interesting discussions and comments.

References

1. D. Beaver, *Foundations of secure interactive computing*, Proc. CRYPTO '91, Springer Verlag LNCS, vol. 576, pp. 377–391.
2. D. Beaver and A. Wool, *Quorum-based multi-party computations*, Proc. EUROCRYPT '98, Springer Verlag LNCS, vol. 1403, pp. 375–390.
3. A. Beimel, *Secure schemes for secret sharing and key distribution*, Ph.D.-thesis, Technion, Haifa, June 1996.
4. J. Benaloh, J. Leichter, *Generalized secret sharing and monotone functions*, Proc. CRYPTO '88, Springer Verlag LNCS, vol. 403, pp. 25–35.
5. M. Ben-Or, S. Goldwasser, A. Wigderson, *Completeness theorems for non-cryptographic fault-tolerant distributed computation*, Proc. ACM STOC '88, pp. 1–10.
6. M. Bertilsson, I. Ingemarsson, *A construction of practical secret sharing schemes using linear block codes*, Proc. AUSCRYPT '92, Springer Verlag LNCS, vol. 718, pp. 67–79.

7. E. F. Brickell, *Some ideal secret sharing schemes*, J. Combin. Maths. & Combin. Comp. 9 (1989), pp. 105–113.
8. R. Canetti, *Studies in secure multi-party computation and applications*, Ph. D. thesis, Weizmann Institute of Science, Rehovot, 1995.
9. R. Canetti, U. Feige, O. Goldreich, M. Naor, *Adaptively secure multi-party computation*, Proc. ACM STOC '96, pp. 639–648.
10. D. Chaum, C. Crépeau, I. Damgård, *Multi-party unconditionally secure protocols*, Proc. ACM STOC '88, pp. 11–19.
11. B. Chor, S. Goldwasser, S. Micali, B. Awerbuch, *Verifiable secret sharing and achieving simultaneity in the presence of faults*, Proc. FOCS '85, pp. 383–395.
12. R. Cramer, I. Damgård, *Zero Knowledge for Finite Field Arithmetic or: Can Zero Knowledge be for Free?*, Proc. CRYPTO'98, Springer Verlag LNCS, vol. 1462, pp. 424–441.
13. R. Cramer, I. Damgård, S. Dziembowski, *On the complexity of verifiable secret sharing and multi-party computation*, Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC '00), Portland, Oregon, May 2000.
14. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt and T. Rabin, *Efficient multi-party computations secure against an adaptive adversary*, Proc. EUROCRYPT '99, Springer Verlag LNCS, vol. 1592, pp. 311–326.
15. C. Crépeau, J. van de Graaf and A. Tapp, *Committed oblivious transfer and private multi-party computation*, proc. CRYPTO '95, Springer Verlag LNCS, vol. 963, pp. 110–123.
16. M. van Dijk, *Secret key sharing and secret key generation*, Ph.D. Thesis, Eindhoven University of Technology, 1997.
17. S. Fehr, *Efficient construction of dual MSP*, manuscript 1999.
18. M. Fitzi, U. Maurer, *Efficient Byzantine agreement secure against general adversaries*, Proc. 12th Int. Symp. on Distributed Computing (DISC '98), Springer Verlag LNCS, vol. 1499, pp. 134–148.
19. A. Gál, *A characterization of span program size and improved lower bounds for monotone span programs*, Proceedings of the 30th ACM Symposium on the Theory of Computing, 1998, pp. 429–437.
20. A. Gál, *Combinatorial methods in Boolean function complexity*, Ph.D.-thesis, University of Chicago, 1995.
21. Z. Galil, S. Haber and M. Yung, *Cryptographic computation: Secure fault-tolerant protocols and the public-key model*, Proc. CRYPTO'87, Springer Verlag LNCS, vol. 293, pp. 135–155.
22. R. Gennaro, *Theory and practice of verifiable secret sharing*, Ph.D. thesis, MIT, 1996.
23. R. Gennaro, M. Rabin, T. Rabin, *Simplified VSS and fast-track multi-party computations with applications to threshold cryptography*, Proc. ACM PODC'98.
24. O. Goldreich, S. Micali and A. Wigderson, *How to play any mental game or a completeness theorem for protocols with honest majority*, Proc. ACM STOC '87, pp. 218–229.
25. M. Hirt, U. Maurer, *Player simulation and general adversary structures in perfect multi-party computation*, Journal of Cryptology, vol. 13, no. 1, pp. 31–60, 2000. (Preliminary version in Proc. ACM PODC'97, pp. 25–34.)
26. M. Ito, A. Saito and T. Nishizeki, *Secret sharing schemes realizing general access structures*, Proc. IEEE GlobeCom '87 Tokyo, pp. 99–102.
27. M. Karchmer, A. Wigderson, *On span programs*, Proc. of Structure in Complexity '93, pp. 102–111.

28. S. Micali and P. Rogaway, *Secure computation*, Manuscript, Preliminary version in Proc. CRYPTO '91, Springer Verlag LNCS, vol. 576, pp. 392–404
29. T. Rabin, M. Ben-Or, *Verifiable secret sharing and multi-party protocols with honest majority*, Proc. ACM STOC '89, pp. 73–85.
30. T. Rabin, *Robust sharing of secrets when the dealer is honest or cheating*, J. ACM, 41(6):1089-1109, November 1994.
31. A. Shamir, *How to share a secret*, Communications of the ACM 22 (1979) 612–613.
32. Technical report, full version of this paper. Will be posted on the Web and is available from the authors. Obsolete are the earlier versions: *Span programs and general secure multi-party computation*, BRICS Report RS-97-28, Nov. 1997, and *Enforcing the multiplication property on MSPs, with only constant overhead*, Jan. 1999.

Minimal-Latency Secure Function Evaluation

Donald Beaver *

CertCo, Inc.

Abstract. Sander, Young and Yung recently exhibited a protocol for computing on encrypted inputs, for functions computable in NC^1. In their variant of secure function evaluation, Bob (the "CryptoComputer") accepts homomorphically-encrypted inputs (x) from client Alice, and then returns a string from which Alice can extract $f(x, y)$ (where y is Bob's input, or *e.g.* the function f itself). Alice must not learn more about y than what $f(x, y)$ reveals by itself. We extend their result to encompass NLOGSPACE (nondeterministic log-space functions).
In the domain of multiparty computations, constant-round protocols have been known for years [BB89,FKN95]. This paper introduces novel parallelization techniques that, coupled with the [SYY99] methods, reduce the constant to 1 with preprocessing. This resolves the conjecture that NLOGSPACE subcomputations (including log-slices of circuit computation) can be evaluated with latency 1 (as opposed to just $O(1)$).

1 Introduction

We consider variants of the now-classic problem raised by Yao [Y86] in which Alice and Bob wish to compute $f(x, y)$ while keeping their respective inputs x and y private. Roughly speaking, their computation should be as "secure" as if a trusted third party had accepted their inputs and provided nothing but the final output. This problem, *Secure Function Evaluation*, has a rich history of investigation, with a great deal of attention given to minimizing needed assumptions and communication complexity.

CEI. One particular variant is that of *Computing on Encrypted Inputs (CEI)*, in which Alice provides Bob with encryptions of x (or its bits), and Bob must enable Alice to determine $C(x)$ without revealing his "program" C. Mathematically, C can itself be encoded as an input to a universal circuit, hence this variant can be subsumed in general secure function evaluation. But the ground rules for CEI are somewhat different, in that Alice provides her input in the form of encryptions rather than through an inventor's flexibly chosen alternative (such as indirectly through oblivious transfer [R81]).

This is somewhat different than the general 2-party setting, in which encryptions can be used as an implementation tool but are not required. Moreover, the encryptions used in "Yao gates" and other earlier techniques are usually meant to encrypt random secrets that indirectly represent secret bits, as opposed to

* CertCo, 55 Broad St., New York, NY 10004; beaverd@certco.com.

encrypting the secret bits themselves. (A concrete hint if this is confusing: Alice often gets to learn one secret x_0 or another secret x_1, each of which is itself random; but the actual value represented by this process is 0 if Alice learns x_0, or 1 if Alice learns x_1.)

In the general SFE setting for two parties, preprocessing obviates the need for "encrypted inputs" and other extra work, since a "scrambled universal circuit" can be prepared in advance and then applied in one round as soon as the actual inputs become available. The challenge is therefore to achieve a one-round protocol without preprocessing (other than public-key initialization and the like).

Recently, Sander, Young and Yung provided a novel construction that enables *non-interactive* computing on encrypted inputs for functions in NC^1, namely functions computed by bounded-fan-in log-depth circuits. ("Non-interactive" means that Bob can complete his computation and return the result to Alice without conversation beyond receiving the initial message from Alice; obviously, Alice's inputs must be communicated to Bob in some form.) Alice simply drops off her input, Bob processes it, and Alice picks up the results. Dubbed *"crypto-computing"* by [SYY99], this methodology has applications to mobile computing and other settings.

Our contribution to non-interactive CEI (cryptocomputing) is to extend the class of functions to NLOGSPACE, *i.e.* non-deterministic logspace, a superclass of NC^1. This extension relies on matrix techniques from Feige, Kilian and Naor [FKN94], but also employs a newly contributed *inversion-free reduction* (§5.5) to compute products of secret group elements in one pass. With these methods, functions in NLOGSPACE can be evaluated in 1 round from scratch, answering a challenge left open by [SYY99], namely whether complexity beyond NC^1 is attainable for non-interactive computing on encrypted inputs.

MSC. Another twist on Secure Function Evaluation introduces some number n of parties, each holding a private input x_i, who wish to compute some function $f(x_1, \ldots, x_n)$ [GMW86,GMW87,BGW88,CCD88]. This version, known as *Multiparty Secure Computation (MSC)*, has also been the subject of extensive analysis.

When "computational" security is considered (as opposed to information theoretic), it is in fact possible to reduce any poly-depth circuit to a protocol with $O(1)$ rounds [BMR90]. With preprocessing and complexity-theoretic assumptions, those methods enable the results to be ready in one round after the inputs are provided, as mentioned above for the 2-party case.

Instead, we focus on the challenge of information theoretic security. For efficient solutions (polynomial message size and local computation), the number of rounds of communication is generally related to the circuit depth for f. Bar-Ilan and Beaver introduced techniques to reduce the computation by log-factors [BB89]; thus functions in NC^1 can be computed in a constant expected number of rounds. In fact, the methods from [FKN94] extend this to functions in NLOGSPACE. But the $O(1)$ constants, while small, still exceed 1.

Unlike the CEI setting, we do focus here on minimizing the *latency* of the computation, namely the number of rounds from when the inputs are supplied to when the output is ready. In [B91] it was shown that 1-round latency for secret multiplication (among other things, such as multiplicative inversion) is achievable.

Applying that work in a brute-force fashion to the [BB89,FKN94] solutions still gives a constant latency exceeding 1, because of the need to compute multiplicative inverses prior to evaluating components of a large product. We apply the methods of [SYY99] to reduce the latency to 1 for NC^1. The final construction provides a particularly elegant view of multiparty computation expressed as a secret linear combination of inputs.

With the inversion-free reduction described in this work, we also show how to achieve a latency of 1 for NLOGSPACE secret computations by avoiding the two-phase process in earlier works. Especially attractive is the fact that, apart from the preprocessing, no broadcast is needed. Thus a single dissemination (broadcast message without agreement) from each honest participant suffices for each NLOGSPACE-subcomputation.

2 Background and Definitions

We consider two different cases for function evaluation: the two-party case and the multiparty case. In the two-party case, hidden values can be represented through encryption, through oblivious transfer, or other such constructs. In the multiparty case, values can be represented through encryption, of course, but more interestingly through secret sharing [S79,B79].

It should be noted that the manipulations of these fundamental representations – encryptions or shares – are quite similar. Thus we may speak of "secret addition" to mean a homomorphic encryption (*e.g.* $E(a)E(b) = E(a \oplus b)$) or to mean a homomorphic additive sharing (*e.g.* $h(x) = f(x) + g(x)$ from $h(i) = f(i) + g(i)$). In general, "secret $\langle$*operation*$\rangle$" can be interpreted according to context, whereby the representation of the result is calculated from the representations of the inputs – be it encryption or sharing or otherwise.

Likewise, the "reconstruction" or "revelation" will refer to interpolation of shares, or decrypting of values and propagation through trees, *etc.*.

We note these observations explicitly to avoid doubling the size of the exposition, since, for example, the use of multiplicative inverses will be discussed both in the context of encrypted-representations and shared-representations.

2.1 Secret Sharing and Multiparty Computation

We refer the reader to [S79,B79,GMW87,BGW88,CCD88] for more detailed exposition. A secret value x can be shared among n parties, at most t of whom are colluding, by selecting t random coefficients $a_t, \ldots, a_1$ and defining $f(u) = a_t u^t + \cdots + a_1 u + x$. Player i receives the **share** $f(i)$. With $t+1$ correct shares,

$f(0) = x$ can be determined. With t or fewer shares, no information about x is revealed.

If $f(u)$ represents $f(0) = x$ and $g(u)$ represents $g(0) = y$, then $h(u) = f(u) + g(u)$ represents $x + y$, and shares $h(i)$ are easily calculated without interaction as $f(i) + g(i)$. Multiplication protocols are more model-dependent, but generally within a small number of rounds of interaction, a polynomial representation of xy can be obtained as well.

There are a variety of models to which our results apply – for example, $t < n/2$ covers one class, and $t < n/3$ covers another. (One can also withstand general adversary structures such as "Q2" and "Q3.") Broadcast is assumed (at least for preprocessing), but we place no restrictions on the computational power of attackers.

For simplicity, we consider t-adversaries who make a static choice of whom they will corrupt at the outset, and we investigate independence rather than simulatability. (Generalizations are possible.) Let $f(x_1, \ldots, x_n)$ be a function on n inputs, each of size m, producing a result of size m, and described by a boolean circuit family C_f. A multiparty protocol for f is a collection $\{P_1, \ldots, P_n\}$ of interactive Turing machines, each taking input m and a private m-bit argument x_i, and producing a final output y_i. A t-adversary is allowed to substitute (and coordinate) up to t of the programs. Two inputs are T-equivalent if they are identical on inputs in T and evaluate under f to the same result.

Definition 1. *A protocol $\Pi = \{P_1, \ldots, P_n\}$ for f is (information theoretically)* t**-secure** *if, for any coalition $T \subseteq \Pi$ of size $|T| \leq t$, and for any T-equivalent input $(x_1, \ldots, x_n)$, the view obtained by T is identically distributed.*

To complicate the analysis, we may allow the inputs x_i to be supplied at some round ρ after the protocol starts. The number of rounds of preprocessing (independent of inputs) is then ρ, and the **latency** is the total number of rounds less ρ. When considering protocols that divide the computation of f into "slices" (*i.e.*, subcomputations), we also consider the latency of computing each slice as the maximal number of rounds from when the previous slice is completed to when the current slice is done.

2.2 Computing on Encrypted Inputs

In CEI, we would like to capture the challenge of dropping off encrypted inputs which are then manipulated in a somewhat black-box fashion to produce a result for the client. This requires a bit more than postulating a homomorphic encryption scheme, as we now discuss.

One of the earliest and most fundamental techniques for two-party circuit evaluation is due to Yao [Y86]. In this method, Bob prepares a scrambled circuit in which each gate is represented by an encrypted table (called a "Yao gate"), and each wire w is represented by a pair of hidden triggers, *i.e.* numbers w_0 and w_1. The value of wire w is 0 if Alice discovers w_0; it is w_1 if Alice discovers w_1; in other cases it is undefined. By propagating the discovered triggers through each encrypted table, Alice is able to calculate the trigger for the output wire

of the gate. She is not told how to interpret the triggers, *i.e.* to which bit the trigger corresponds – except for the final output wire.

The needed interaction is minimal, on the same order as the [SYY99] setting. Alice must obtain initial wire triggers that correspond to her secret input bits. This is achieved through chosen-1-out-of-2 Oblivious Transfer [R81]: Bob supplies w_{i0} and w_{i1} for an input wire w_i; Alice learns precisely one, namely w_{i,x_i}; but Bob does not learn which Alice chose. Bob also sends Alice the scrambled circuit. Subsequently, Alice can calculate the output value without further interaction.

Given a homomorphic encryption scheme, one quick way to implement the OT is by way of a generalization of Den Boer's method [dB90]. Alice sends $E(c)$ where c is a bit describing her choice. Bob responds with $\{E(b_0), E(a)/E(b_0)\}$, $\{E(b_1), E(a)E(1)/E(b_1)\}$, with sets and members permuted randomly. Given proper behavior, Alice decrypts the sets to $\{0,1\}$ and $\{b_c, b_c\}$, hence she obtains b_c. The authors of [SYY99] invoke a variety of options to demonstrate good behavior without introducing interaction; those options apply here as well. Note that Bob can send the scrambled circuit along with his OT reply, making the whole interaction non-interactive, so to speak.

Thus, if Bob can employ a homomorphic encryption secure against Alice, an immediate solution is possible for any polynomial-time f, not just one in NC^1. This solution makes an end run around the spirit of the problem. Since it is hard to provide a formal test that captures whether Bob's computations are nothing "more" than a manipulation of encrypted values (there are a lot of clever and indirect things he can do), we turn to a simple requirement: the protocol must be information-theoretically secure against Alice.

Definition 2. *A* **CEI protocol** *for function f represented by circuit C_f is a two-party protocol consisting of a message from Alice to Bob followed by one in return. The protocol is* **correct** *if for all inputs (x, y), Alice's output is $f(x, y)$ except with negligible probability. A CEI protocol is* **private** *if it is computationally-private against Bob and information-theoretically private against Alice.*

Concrete Examples of Encryptions A couple of common encryptions make suitable candidates. One is the Goldwasser-Micali encryption [GM84] in which N is a public Blum integer with a private key consisting of its factors P, Q. Bit b is encrypted as $(-1)^b r^2$ for a random r. This is secure assuming that quadratic residues are indistinguishable from non-residues (the Quadratic Residuosity Assumption, or QRA).

A second candidate is a variant of El-Gamal encryption with primes P, Q satisfying $P = 2Q + 1$, and a generator g of Z/QZ. Corresponding to private key x is the public key $y = g^x$. To encrypt message m taken from the space of quadratic residues, compute $\{g^r, y^r m\}$. Encryption of 0 and 1 uses two fixed, public quadratic residues m_0, m_1. The security of this method is equivalent to Decision Diffie-Hellman [TY98,NR97].

In each of these cases, given some $E(b)$, it is easy to see how to sample random encryptions uniformly from the set of all encryptions of b, or of $1-b$, even without knowing b.

3 Pyramid Representation

The foundation for the recent 1-round protocol of Sander, Young and Yung is an ingenious tree representation for a circuit output. We will build multiparty protocols around their architecture, thus we give details here; the familiar reader can skip to the next section.

Let us coin the term **pyramid representation** to describe the data structure employed in [SYY99]: a complete 4-2 tree, *i.e.* a tree with degree 4 at root and even-level nodes, and with degree 2 at all odd-level nodes. We take the root to be level $2d$ and the leaves to be at level 0. There are 8^d leaves.

There are three important aspects to the SYY construction. First, the nodes can be evaluated in terms of a given circuit, resulting in the root being assigned a value equal to the output of the circuit. Second, the pyramid representation can be constructed from encrypted leaf values without knowing what the cleartext bits are. Third, the pyramid representation can be randomized so that it appears chosen uniformly at random from all such representations that evaluate to the given root value.

The authors of [SYY99] refer to the construction and randomizing as *inattentive* computing, suggesting that the party who performs the tasks need not pay attention to the actual values themselves. The manipulations are oblivious to the contents.

Decoding. In slightly more detail for completeness, we first summarize how evaluation/decoding takes place, given bit assignments to the leaves. (Ultimately, each leaf corresponds to an encryption, and the value of the leaf node is the decrypted bit.) Propagating upward, a node at level $2k+1$ has two children, (a, b), and is assigned the value $a \oplus b$. A node at level $2k$ has four children (a, b, c, d), and is assigned the value 0 if three are labelled 0 and one is 1, or respectively 1 if three children are labelled 1 and one is 0. (All other cases are syntactically unacceptable and are given an `undef` label.) This three-of-one-kind representation is critical.

Construction. To construct a pyramid representation of the value of some function f applied to input bits $x_1, \ldots,$ one must apply the gates of a circuit C_f for f to the nodes in the representation. Inputs and constants lie at the leaves. Without loss of generality, express C_f as NOT and OR gates. We briefly summarize the SYY construction using the following procedures, which depend on the level of the node in the tree:

- NOT(x:level 0) gives y:level 0.
  - set $y = x \oplus 1$. (Later, 0 and 1 may be encoded by $(0,1)$ and $(1,0)$, in which case this is operation is instead a swap.)

- NOT(x:level $2k+2$) gives y:level $2k+2$.
  - return $((\text{NOT}(a_1), a_2), (\text{NOT}(b_1), b_2), (\text{NOT}(c_1), c_2), (\text{NOT}(d_1), d_2))$, where $x = ((a_1, a_2), (b_1, b_2), (c_1, c_2), (d_1, d_2))$.
- OR(x:level $2k$,y:level $2k$) gives z:level $2k+2$.
  - return $((x, 0), (y, 0), (x, y), 1')$ where 0 denotes a level $2k$ zero, and $1'$ denotes a level $2k+1$ one.

The ingenious motivation behind the three-of-one-kind representation is now more clear. Negating each individual bit in the multiset $\{0, 0, 0, 1\}$ provides a three-of-one-kind result $\{1, 1, 1, 0\}$, and vice versa. More importantly, the results of the OR routine are always in a three-of-one-kind configuration, when interpreted at a higher level. Explicitly:

x	y	$(x,0)$	$(y,0)$	(x,y)	1'	OR
0	0	0'	0'	0'	1'	{0',0',0',1'}
0	1	0'	1'	1'	1'	{0',1',1',1'}
1	0	1'	0'	1'	1'	{1',0',1',1'}
1	1	1'	1'	0'	1'	{1',1',0',1'}

where the primed values are interpreted at the next layer up ($(0,0)$ and $(1,1)$ are written 0', *etc.*).

Randomization. [SYY99] show that the following straightforward method turns a particular pyramid representation of some result z into a randomly-chosen valid pyramid representation of z, thereby hiding the inattentive steps used to construct the original representation. The randomization method is itself inattentive to the contents of the pyramid.

- RANDOMIZE(x:level $2k+2$) gives y:level $2k+2$.
  - let x be $((x_{11}, x_{12}), (x_{21}, x_{22}), (x_{31}, x_{32}), (x_{41}, x_{42}))$;
  - for $i = 1..4$ and $j = 1..2$, set $b_{ij} \leftarrow \text{RANDOMIZE}(x_{ij})$;
  - for $i = 1..4$, set c_i by random choice to be (b_{i1}, b_{i2}) or $(\text{NOT}(b_{i1}), \text{NOT}(b_{i2}))$;
  - choose random permutation $\sigma \in S_4$ and return $(c_{\sigma(1)}, c_{\sigma(2)}, c_{\sigma(3)}, c_{\sigma(4)})$.

3.1 Non-Interactive Computing on Encrypted Inputs

In the [SYY99] paper, this construction is applied to encrypted inputs. That is, Alice presents CryptoComputer Bob with encryptions $E(x_i)$ of each of her input bits x_i, along with their inverses $E(1 - x_i)$. This enables Bob to create the level 0 leaf labels. Note that Bob can also encrypt his secret inputs x_j, as well as any known constants, thereby filling in any other needed labels.

Now, without knowing the contents of the encryptions, Bob can invoke the NOT and OR routines, and finally the RANDOMIZE routine. The result is a pyramid representation whose root value is $f(x, y)$. Bob sends this to Alice.

Alice is able to decrypt the labels on the leaves and can subsequently evaluate the root value.

4 Multiparty Secure Computation

With the pyramid data structure in place, we are now ready to give a multiparty secure computation for NC^1.

4.1 Latency vs. Cost: Circuit Randomization

When calculating from scratch, our MSC results will generally incur a minimal cost of one secret multiplication. While still better than previously published results, this falls short of the most desirable bound of 1 round, period.

Instead, we focus on **latency**, defined as the number of rounds from when the inputs to a computation phase are provided until the output (whether secret or public) is complete. Preprocessing is acceptable (and likely required), but it must be independent of the inputs to be used.

Latency is particularly important when evaluating a depth-D circuit using $(\log n)$-slices to speed up the number of rounds. A brute-force approach would normally require $C\dot{D}/\log n)$ multiplications with C much larger than 1 (and even including our results below, it would be at least $D/\log n$ multiplications). If, however, the later slices benefit from preprocessing that is performed during the first slice, then the net running time can be drastically reduced. That is, one multiplication plus $D-1$ rounds is far better than D sequential multiplications.

One way to improve latency was shown by Beaver, using a technique called *circuit randomization* [B91]. With appropriate preprocessing, this enables each secret multiplication to finish in one round, an order of magnitude faster than the cost of a secret multiplication from scratch.

The preprocessing is simple, consisting of computing secret products on secret, random inputs. Thus, for example, secrets a, b, c with $c = ab$ are created in advance. When x and y are ready to be multiplied, the differences ("corrections") $\Delta x = x - a$ and $\Delta y = y - b$ are published. The "correction" to c, namely $\Delta z = xy - c$, then becomes a straightforward linear combination with public coefficients (the Δx, Δy values). The bottom line is that secret multiplication has a latency of 1 round.

We shall see below that the same conclusion applies to NC^1 (and to NLOGSPACE): secret NC^1 computations have a latency of 1 round. Interestingly, the following result can be derived in different ways, with or without the recent SYY methods.

Claim. Let f be represented by a circuit C_f of polynomial size. There exists a secure MSC protocol to compute NC^1 slices of C_f with a latency of 1 round.

4.2 NC1 Via Secret Quadratic Forms and SYY

The first of two ways to achieve Claim 4.1 employs [SYY99] with secretly shared values in place of encrypted bits. The "inattentive" creation of a pyramid representation on secrets is done as a multiparty computation in a straightforward manner.

The calculation of NOT at level 0 is simple: non-interactively compute $1 - x$ secretly. Second, the RANDOMIZE step can be calculated using a secret quadratic form applied to the inputs – or in other words, a "linear" combination of input values in which the coefficients are themselves secrets. These coefficients are chosen randomly but with certain restrictions.

There are only two steps in RANDOMIZE in which random choices are made. In the 2-party Computing on Encrypted Inputs setting, the "CryptoComputer" would make these choices and ensure that they remain secret. In the MSC application, these choices are also kept secret. We must ensure that they can be selected and applied efficiently.

Referring to §3, there are two main steps for applying random choices. First is the choice between (b_{i1}, b_{i2}) and $(\text{NOT}(b_{i1}), \text{NOT}(b_{i2}))$. This choice can be executed by creating a new secret bit d_i, then setting (at leaves):

$$b_{i1} = d_i x_{i1} + (1 - d_i)(1 - x_{i1})$$
$$b_{i2} = d_i x_{i2} + (1 - d_i)(1 - x_{i2})$$

The manipulation at higher level nodes is similar: the multiplication by d_i is propagated to the children.

Similarly, the random selection of a permutation from S_4 can be modelled by a secret permutation matrix $A = [a_{ij}]$, so that the resulting quadruple is (y_1, y_2, y_3, y_4) where $y_i = \sum_j a_{ij} c_j$.

At each odd-level node in the pyramid representation, then, a secret random bit is generated. At each even-level node above 0, a secret random S_4 permutation is generated.

If these operations are composed, the result is a collection of coefficients C_{ij} such that leaf i is $C_{i0} + \sum_j C_{ij} x_j$. These coefficients are products of the coefficients assigned on the path down to leaf i. Thus they can be efficiently calculated (secretly, of course) in a preprocessing phase.

Noting that [B91] enables quadratic forms on secrets to be evaluated with 1-round latency, Claim 4.1 is satisfied.

4.3 Some Details

For concreteness, here are some ugly programming steps for the protocol. The SYY construction induces at each node a tree with formulas in it. One can apply a syntactic NOT operation to a leaf label s by replacing s by $1 - s$. One can apply a NOT to a higher node by applying NOT recursively to each of the left grandchildren (as in SYY). One can also perform linear combinations recursively on two trees of formulas in a direct manner:

$$a((t_1, t_2), (t_3, t_4), (t_5, t_6), (t_7, t_8)) + b((u_1, u_2), (u_3, u_4), (u_5, u_6), (u_7, u_8)) =$$
$$((at_1 + bu_1, at_2 + bu_2), (at_3 + bu_3, at_4 + bu_4),$$
$$(at_5 + bu_5, at_6 + bu_6), (at_7 + bu_7, at_8 + bu_8)).$$

The first (non-interactive) preparation creates a raw tree of formulas:

1. Start with circuit C_f which is applied to input bits $x_1, \ldots, x_m$.

2. Create a *raw pyramid program*: Each node contains a tree of formulas using x_i's and constants. Place x_i's and constants at the leaves according to C_f. Propagating upward, create a formula tree at each node according to the Construction in 3. (For example, at level 2, OR(NOT(x_1),x_2) would be labelled with the formula tree $((1-x_1,0),(x_2,0),(1-x_1,x_2),(0,1)$.)

The second (non-interactive) preparation adds symbols that correspond to the randomization:

1. For each odd-level node v, create symbol $d(v)$. For each even-level node v, create 16 symbols $a(v,i,j)$ for $1 \le i,j \le 4$.
2. Create a *randomized pyramid program*: Propagating upwards from leaves, apply randomization symbols.
   2A. Replace the current T by $T' = d(v)T + (1-d(v))NOT(T)$. (This involves recursively applying $d(v)$ symbols and NOT's; the result is a tree of formulas over inputs, constants, and d_i's.)
   2B. Now say the current T is $((b_{11},b_{12}),(b_{21},b_{22}),(b_{31},b_{32}),(b_{41},b_{42}))$. Replace b_{im} with $\sum_j a(v,l,j)b_{jm}$. (Again, the $a(v,l,j)$ symbols trickle to the leaves.)

The result is a pyramid of formulas in which each formula can be written as $C_{i0} + \sum_j C_{ij}x_j$, where the C's are formulas on the randomization symbols and constants alone.

This gives an $O(8^d)$-sized "program" for the preprocessing phase, where d is the circuit depth of C_f. Generate random secret bits for each of the $d(v)$ symbols. Generate random secret permutation matrices for each set $\{a(v,i,j)\}$. Evaluate each C_{i0} and C_{ij} *secretly*. The preprocessing takes constant rounds.

We now have a pyramid in which each leaf i contains an expression $C_{i0} + \sum_j C_{ij}x_j$. Following the approach of [B91], these results can be precomputed at random values $\hat{x}_i$. When the x_j inputs are provided, "corrections" $(x_i - \hat{x}_i)$ are announced, corrections to the pyramid entries are disseminated (no broadcast/agreement needed), and each player then calculates the entries himself. Each player then evaluates the pyramid according to the instructions of [SYY99] (see §3).

Preparing the Coefficients We digress with a few remarks on alternatives for obtaining the C_{ij} coefficients. Several avenues present themselves:

- generation by precomputation;
- generation by Trusted Third Party (TTP) or Server;
- generation by composition.

The previous section considered precomputation.

In a hybrid model more akin to [SYY99], one can rely on a TTP who supplies secret shares of the coefficients to the participants. While zero-knowledge proofs can ensure correctness (*i.e.* the coefficients are a proper permutation), one must trust that the TTP does not leak the coefficients. This trust model is similar to the CryptoComputer model of [SYY99]; secrecy relies on maintaining secrecy of the RANDOMIZE step.

Finally, verified sets of coefficients from the TTP's can be composed. This corresponds to allowing each TTP to execute the RANDOMIZE step. As long as

one TTP maintains discretion, the conclusions of [SYY99] will apply and the results will be secure. Of course, if the TTP's are taken to be the participants themselves (*eg.* $t+1$ of them), then a secret matrix product on several matrices is required, which gets us back to the initial problem.

5 Matrix Representations

We present a background for matrix-based computing and finish the section with our new inversion-free reduction.

5.1 Secret Group Operations

The following subroutines are applicable to 2-party and to multiparty settings. Note that the group need not be abelian, thus matrices are perfectly fine candidates. The costs are $O(1)$ multiplications; hence if secret multiplication takes $O(1)$ rounds, the net cost is $O(1)$ rounds. (As described later, secret multiplication generally has 1-round *latency* after preprocessing, so these routines are very short in terms of latency.)

Inverses. The authors of [BB89] demonstrated how to compute a secret inverse of a secret group element X in $O(1)$ multiplications using the following trick: choose secret element U; secretly calculate $Y = XU$ and reveal Y; publicly calculate Y^{-1}; secretly multiply $Z = UY^{-1}$. Clearly, $Z = X^{-1}$, yet Y is distributed uniformly at random, revealing nothing about X (as long as U remains uncompromised).

Polynomial-Length Products. Let $M_1, \ldots, M_N$ be secret group elements. The goal is to calculate $M = \prod_i M_i$ secretly. The following application (with minor differences) arose in [K88,K90] and [BB89]:

$$M = R_0^{-1} R_0 M_1 R_1^{-1} R_1 M_2 R_2^{-1} \cdots R_{N-1} M_N R_N^{-1} R_N$$

where $R_0, \ldots, R_N$ are secret, random, invertible group elements (R_0 can be set to the identity). Let $S_i = R_{i-1} M_i R_i^{-1}$. Then the set $\{S_i\}$, if made public, reveals nothing about $\{M_i\}$; it appears uniformly random, subject to producing the same overall product.

A protocol that follows this structure (compute R_i's and inverses, compute and reveal S_i's) will incur $O(1)$ multiplications plus the cost of generating random invertible elements. It will nevertheless exceed 1 round.

5.2 3×3 Products for NC1

Building on a result of Barrington [B86], Ben-Or and Cleve [BC88] showed that NC^1 computations are equivalent to products of polynomially-many 3×3 matrices. In their representation, inputs are supplied as an identity matrix with the

top right $(1,3)$ zero replaced by the input value. The final result is also read from the $(1,3)$ entry of a specified product of such "input" matrices interspersed with certain constant matrices. In fact, the final product is simply an identity matrix with the top right zero replaced by $f(x_1,\ldots,x_n)$.

Without going into further detail, we mention simply that the number of matrices involved in a depth-d computation will be some $N = O(4^d)$, and that each matrix is either a well-known constant or simply contains an input variable (possibly negated) in the $(1,3)$ entry as above.

5.3 $N \times N$ Products for NLOGSPACE

More recently, Feige, Kilian and Naor [FKN94] described how to formulate NLOGSPACE computations as a product of $N \times N$ matrices, where N is polynomial in the input size. In their setup, the top right $(1,N)$ entry of the final product M indicates the final output: 0 if the entry is zero, or 1 if the entry is nonzero.

Because [FKN94] used the $N \times N$ construction to solve a slighly different task, in which Alice and Bob provide sufficient data to a Combiner so that the Combiner can calculate $f(x,y)$ without learning x and y, they also focused on leaving $f(x,y)$ (and nothing else) in the output. While this occurs automatically in the 3×3 matrix case (for NC^1), [FKN94] had to provide additional secret matrices Q_L and Q_R to randomize the final product matrix. With Q_L and Q_R of a particular, randomized form, they showed that $Q_L M Q_R$ was uniformly random subject to the entry at $(1,N)$ being either zero if the output was 0 or random and nonzero if the output was 1.

It is not hard to verify that secret Q_L and Q_R matrices can be generated in a constant expected number of rounds.

5.4 Direct Output or Slice Output

There is a distinction between producing the final result of a function in some public fashion (known to one or more parties) and producing a secret representation of the final result. The latter can be used to speed up larger circuit evaluations by slicing them [K88,K90,BB89] into (for example) log-depth layers.

In any case, it is often simple to convert a direct-output computation to one that preserves the output as a secret for input to further computation. Simply create an additional secret r, directly output the result of $f() - r$, and implicitly add the public value $f() - r$ to the secretly represented r.

(This does not obviate the use of Q_L and Q_R in [FKN94], however, since there are a host of other entries ($N^2 - 1$ of them, in fact) whose public revelation may compromise sensitive information. Their approach was to open the final matrix completely.)

5.5 Multiplication Without Secret Inverses

One of the difficulties with using the matrix multiplication methods described in §5.1 is that they are *prima facie* interactive in nature. To calculate an inverse,

one must publicly reveal the randomized product, which is then interactively fed back into another pass. To calculate a long product of elements, one first reveals the intermediate products of triples, then calculates their product and feeds it back into another phase (multiplying by secrets on left and/or right).

Here, we propose an *inversion-free reduction* from a product to a list of publicized matrices which can be combined to calculate the original product. (While no inversions are needed in the reduction, some of the resulting matrices must be inverted before multiplying them together.)

Starting with a polynomial-length product $M = \prod M_i$, we create secret, invertible elements $R_0, \ldots, R_N$ as before. But now, also create secret, invertible elements $\hat{R}_0, \ldots, \hat{R}_N$. Write:

$$M = (\hat{R}_0)(R_0\hat{R}_0)^{-1}(R_0M_1\hat{R}_1)(R_1\hat{R}_1)^{-1}(R_1M_2\hat{R}_2)\cdots \\ \cdots(R_{N-1}\hat{R}_{N-1})^{-1}(R_{N-1}M_N\hat{R}_N)(R_N\hat{R}_N)^{-1}(R_N).$$

Let $S_i = R_{i-1}M_i\hat{R}_i$, and let $\hat{S}_i = R_i\hat{R}_i$. Then:

$$M = \hat{R}_0\hat{S}_0^{-1}S_1\hat{S}_1^{-1}S_2\cdots\hat{S}_{N-1}^{-1}S_N\hat{S}_N^{-1}R_N.$$

It is not hard to generalize [K88,K90,BB89] to show that each S_i and $\hat{S}_i$ leaks no information. Define $S = \hat{S}_0^{-1}S_1\hat{S}_1^{-1}S_2\cdots\hat{S}_{N-1}^{-1}S_N\hat{S}_N^{-1}$. Then $M = \hat{R}_0SR_N$. While inverses are applied to the public values ($\hat{S}_i^{-1}$), no inversion is required to reduce the original product secretly to the list of public multiplicands.

6 Multiparty Secure Computation Revisited

6.1 Achieving NC1 for Multiparty Secure Computation

Claim 6.2 can now be demonstrated by an alternative approach. The inversion-free reduction of §5.5 enables MSC protocols with 1-round latency for NC^1 without relying on [SYY99], as the following indicates. Precompute the R_i and $\hat{R}_i$ matrices and reveal the $\hat{S}_i$ values.

Let $I(i)$ be the index of the secret input variable appearing in matrix M_i (if any). When each $R_iM_i\hat{R}_{i+1}$ product is expanded, each of the nine entries $\{s_{ikl}\}_{1\le k,l\le 3}$ in S_i is of the form $\alpha_{ikl}+\beta_{ikl}x_{I(i)}$. (If no variable appears, $\beta_{ikl} = 0$.) Secretly precompute the α_{ikl} and β_{ikl} values.

Finally, when the input variables are supplied, it remains to publish each $\alpha_{ikl} + \beta_{ikl}x_{I(i)}$ in order to reveal the S_i matrices. This involves a single multiplication, which the methods of [B91] reduce to latency 1. (The product is precomputed on random inputs; the single round consists of disseminating an adjustment to the precomputed result.)

At this point, the $\hat{S}_i$ and S_i matrices have been revealed. The overall result can be evaluated without further interaction, or fed secretly into the next layer of computation.

6.2 Achieving NLOGSPACE for Multiparty Secure Computation

The generation of secret nonsingular $N \times N$ matrices, and appropriate secret Q_L and Q_R matrices, can be done in expected $O(1)$ rounds. Thus we find (as already claimed in [FKN94]) that there is a secure multiparty protocol for any NLOGSPACE function, using expected $O(1)$ rounds. But we can now strengthen that conclusion by applying the methods of the previous section to $N \times N$ matrices:

Claim. Let f be represented by a composition of D NLOGSPACE-computable functions each with output size polynomial in the size of f's input. There exists a secure MSC protocol to compute each NLOGSPACE-computable subfunction with a latency of 1 round. The overall protocol incurs $D + O(1)$ rounds.

7 Two Parties: Computing on Encrypted Inputs

In the case of Computing on Encrypted Inputs, we do not have the flexibility to allow preprocessing. Instead, we turn back to the [SYY99] for bootstrapping the product of $N \times N$ matrices.

The selection of random, secret, nonsingular matrices, and the individual computation of each of the Q_L, Q_R, S_i and $\hat{S}_i$ matrices can be performed in NC^1. Note that input bits and extra random bits are re-used in different, parallel sub-executions.

Thus, on a higher level, the protocol for NLOGSPACE consists of some number N of executions of various NC^1 calculations. These calculations provide Alice with the values for Q_L, Q_R, S_i and $\hat{S}_i$, which in turn enable her to compute the final bit. According to the proofs presented in [FKN94], these matrices provide no extra information. More details are below.

7.1 Computing NLOGSPACE on Encrypted Inputs

For a given function f in NLOGSPACE, the construction in [FKN94] produces a pair of adjacency matrices, A and B. The binary entries in A depend only on Alice's inputs (or on no inputs at all), and the entries in B depend only on Bob's inputs. The $(1, N)$ entry of $(AB)^N$ will be nonzero if and only if the result of f is 1; otherwise f is 0. To hide the other entries in $(AB)^N$, which may leak information, two extra secret matrices Q_L and Q_R are used, and the desired product is $M = Q_L(AB)^N Q_R$. Bob will enable Alice to find the product of these $2N + 2$ $N \times N$ matrices.

In our application, only Alice will learn the S_i and $\hat{S}_i$ matrices. Unlike other settings, this permits us to have Bob learn or set the randomizing matrices himself, as long as Alice doesn't.

1. For each input bit x_i held by Alice, Alice encrypts and sends $E(x_i)$ to Bob.

2. Bob selects $2N+1$ random R_i matrices and $2N+1$ random $\hat{R}_i$ matrices (set $R_{2N+2} = \hat{R}_{2N+2} = I$). Bob selects Q_L and Q_R at random according to the constraints in [FKN94]. He sets matrix B according to the inputs to f that he holds. Let $M_1 = Q_L$, $M_{2N+2} = Q_R$, and for $i = 1..N$ let $M_{2i} = A$ (values unknown to Bob) and $M_{2i+1} = B$.
3. Bob invokes N instances of the [SYY99] protocol. In instance i he uses Alice's encryptions to evaluate (for Alice) the result $S_{2i} = R_{2i-1}M_{2i}\hat{R}_{2i}$. In addition, Bob directly sends the following results to Alice: $S_1 = Q_L\hat{R}_1$, $S_{2N+2} = R_{2N+1}Q_R\hat{R}_{2N+2}$, $S_{2i+1} = R_{2i}B\hat{R}_{2i+1}$ for $1 \leq i \leq N$, and $\hat{S}_j = R_j\hat{R}_j$ for $1 \leq j \leq 2N+2$.
4. Alice receives pyramids for S_{2i} $(1 \leq i \leq N)$ and calculates S_{2i} accordingly. She then calculates $M = S_1\hat{S}_1^{-1} \cdots \hat{S}_{2N+1}^{-1}S_{2N+2}\hat{S}_{2N+2}^{-1}$. If entry $(1, N)$ in M is nonzero, Alice outputs 1, else she outputs 0.

By inspection, the protocol takes one round. By arguments in [FKN94] and [SYY99], Alice's view of the pyramids and the direct matrices provides her no greater knowledge than the final result itself (from which she can construct the view). The product is clearly correct.

8 Closing Remarks

We have extended the reach of earlier results by applying new parallelization constructs. Two results obtain. Multiparty Secure Computation can be speeded up by creating subtasks of complexity NLOGSPACE, where the latency of computing each subtask is not just $O(1)$ but exactly 1. Likewise, Computing on Encrypted Inputs can be achieved non-interactively for functions in NLOGSPACE, not just NC^1.

We presented two approaches to achieving NC^1 computations for MSC with 1-round latency. One, based on [SYY99], has message size complexity of $O(8^d)$ (where d is circuit depth). The other requires $O(4^d)$. On closer inspection, the culprit seems to be the use of $(0,1)/(1,0)$ representations. In the MSC application, it can be removed, collapsing the SYY pyramid to size $O(4^d)$. It is remarkable that two distinct constructions converge to the same complexity, which may suggest a deeper relationship.

Acknowledgements The author gratefully acknowledges helpful discussions and inspirations from Moti Yung. Several referees made extremely helpful comments on content and presentation.

References

[BB89] J. Bar-Ilan, D. Beaver. "Non-Cryptographic Fault-Tolerant Computing in a Constant Expected Number of Rounds of Interaction." *Proceedings of PODC,* ACM, 1989, 201–209.

[B86] D. Barrington. "Bounded Width Polynomial Size Branching Programs Recognize Exactly those Languages in NC^1." *Proceedings of the* 18th *STOC,* ACM, 1986, 1–5.

[B91] D. Beaver. "Efficient Multiparty Protocols Using Circuit Randomization." *Advances in Cryptology – Crypto '91 Proceedings,* Springer–Verlag LNCS 576, 1992, 420–432.

[BMR90] D. Beaver, S. Micali, P. Rogaway. "The Round Complexity of Secure Protocols." *Proceedings of the* 22nd *STOC,* ACM, 1990, 503–513.

[BC88] M. Ben-Or, R. Cleve. "Computing Algebraic Formulas Using a Constant Number of Registers." *Proceedings of the* 20th *STOC,* ACM, 1988, 254–257.

[BGW88] M. Ben-Or, S. Goldwasser, A. Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation." *Proceedings of the* 20th *STOC,* ACM, 1988, 1–10.

[CCD88] D. Chaum, C. Crépeau, I. Damgaard. "Multiparty Unconditionally Secure Protocols." *Proceedings of the* 20th *STOC,* ACM, 1988, 11–19.

[CDG87] D. Chaum, I. Damgard, J. van de Graaf. "Multiparty Computations Ensuring Secrecy of Each Party's Input and Correctness of the Output." *Advances in Cryptology – Crypto '87 Proceedings,* Springer–Verlag LNCS 293, 1988.

[B79] G.R. Blakley. "Safeguarding Cryptographic Keys." *Proceedings of AFIPS 1979 National Computer Conference*, NY **48**, 1979, 313–317.

[dB90] B. den Boer. "Oblivious Transfer Protecting Secrecy." *Advances in Cryptology – Eurocrypt '90 Proceedings,* Springer–Verlag LNCS 547, 1990, 31–45.

[FiSh86] A. Fiat, A. Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." Proc. of Crypto '86.

[FFS88] U. Feige, A. Fiat, A. Shamir. "Zero Knowledge Proofs of Identity." *J. Cryptology* **1**:2, 1988, 77–94.

[FKN94] U. Feige, J. Kilian, M. Naor. "A Minimal Model for Secure Computation." *Proceedings of the* 26th *STOC,* ACM, 1994, 554–563.

[GMW86] O. Goldreich, S. Micali, A. Wigderson. "Proofs that Yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design." *Proceedings of the* 27th *FOCS,* IEEE, 1986. pages 174–187. IEEE, 1986.

[GMW87] O. Goldreich, S. Micali, A. Wigderson. "How to Play Any Mental Game, or A Completeness Theorem for Protocols with Honest Majority." *Proceedings of the* 19th *STOC,* ACM, 1987, 218–229.

[GM84] S. Goldwasser, S. Micali. Probabilistic Encryption. *JCCS*, **28**:2, 270–299, 1984.

[K88] J. Kilian. "Founding Cryptography on Oblivious Transfer." *Proceedings of the* 20th *STOC,* ACM, 1988, 20–29.

[K90] J. Kilian. Uses of Randomness in Algorithms and Protocols. Cambridge, MIT Press, 1990.

[NR97] M. Naor, O. Reingold. "Number-Theoretic Consturctions of Efficient Pseudo-Random Functions." *Proceedings of the* 38th *FOCS,* IEEE, 1997.

[R81] M.O. Rabin. "How to Exchange Secrets by Oblivious Transfer." TR-81, Harvard, 1981.

[SYY99] T. Sander, A. Young, M. Yung. "Non-Interactive CryptoComputing for NC1." *Proceedings of the* 40th *FOCS,* IEEE, 1999.

[S79] A. Shamir. "How to Share a Secret." *Communications of the ACM,* **22**, 1979, 612–613.

[TY98] Y. Tsiounis, M. Yung. "On the Security of ElGamal-based Encryption." Proceedings of PKC '98, LNCS, Springer-Verlag, 1998.

[Y82] A. Yao. "Protocols for Secure Computations." *Proceedings of the* 23rd *FOCS,* IEEE, 1982, 160–164.

[Y86] A. Yao. "How to Generate and Exchange Secrets." *Proceedings of the* 27th *FOCS,* IEEE, 1986, 162–167.

Information-Theoretic Key Agreement: From Weak to Strong Secrecy for Free

Ueli Maurer[1] and Stefan Wolf[1]

Computer Science Department, Swiss Federal Institute of Technology (ETH Zurich), CH-8092 Zurich, Switzerland
{maurer,wolf}@inf.ethz.ch

Abstract. One of the basic problems in cryptography is the generation of a common secret key between two parties, for instance in order to communicate privately. In this paper we consider information-theoretically secure key agreement. Wyner and subsequently Csiszár and Körner described and analyzed settings for secret-key agreement based on noisy communication channels. Maurer as well as Ahlswede and Csiszár generalized these models to a scenario based on correlated randomness and public discussion. In all these settings, the secrecy capacity and the secret-key rate, respectively, have been defined as the maximal achievable rates at which a highly-secret key can be generated by the legitimate partners. However, the privacy requirements were too weak in all these definitions, requiring only the ratio between the adversary's information and the length of the key to be negligible, but hence tolerating her to obtain a possibly substantial amount of information about the resulting key in an absolute sense. We give natural stronger definitions of secrecy capacity and secret-key rate, requiring that the adversary obtains virtually no information about the entire key. We show that not only secret-key agreement satisfying the strong secrecy condition is possible, but even that the achievable key-generation rates are equal to the previous weak notions of secrecy capacity and secret-key rate. Hence the unsatisfactory old definitions can be completely replaced by the new ones. We prove these results by a generic reduction of strong to weak key agreement. The reduction makes use of extractors, which allow to keep the required amount of communication negligible as compared to the length of the resulting key.

1 Introduction and Preliminaries

1.1 Models of Information-Theoretic Secret-Key Agreement

This paper is concerned with *information-theoretic security* in cryptography. Unlike *computationally-secure cryptosystems*, the security of which is based on the assumed yet unproven hardness of a certain problem such as integer factoring, a proof without any computational assumption, based on information theory rather than complexity theory, can be given for the security of an information-theoretically (or unconditionally) secure system.

A fundamental problem is the generation of a mutual key about which an adversary has virtually no information. Wyner [18] and later Csiszár and Körner [10] considered the natural message-transmission scenarios in which the legitimate partners Alice and Bob, as well as the adversary Eve, are connected by noisy channels. In Csiszár and Körner's setting, Alice sends information (given by the random variable X) to Bob (receiving Y) and to the opponent Eve (who obtains Z) over a noisy broadcast channel characterized by the conditional distribution $P_{YZ|X}$. Wyner's model corresponds to the special case where $X \to Y \to Z$ is a Markov chain.

The *secrecy capacity* $C_S(P_{YZ|X})$ of the channel $P_{YZ|X}$ has been defined as the maximal rate at which Alice can transmit a secret string to Bob by using only the given noisy (one-way) broadcast channel such that the rate at which the eavesdropper receives information about the string can be made arbitrarily small. More precisely, the secrecy capacity is the maximal asymptotically-achievable ratio between the number of generated key bits and the number of applications of the noisy broadcast channel such that Eve's per-letter information about the key is small.

As a natural generalization of these settings, Maurer [13] and subsequently Ahlswede and Csiszár [1] have considered the model of secret-key agreement by public discussion from correlated randomness. Here, two parties Alice and Bob, having access to specific dependent information, use authentic public communication to agree on a secret key about which an adversary, who also knows some related side information, obtains only a small fraction of the total information. More precisely, it is assumed in this model that Alice and Bob and the adversary Eve have access to repeated independent realizations of random variables X, Y, and Z, respectively. A special example is the situation where all the parties receive noisy versions of the outcomes of some random source, e.g., random bits broadcast by a satellite at low signal power.

The *secret-key rate* $S(X;Y||Z)$ has, in analogy to the secrecy capacity, been defined in [13] as the maximal rate at which Alice and Bob can generate a secret key by communication over the noiseless and authentic but otherwise insecure channel in such a way that the opponent obtains information about this key only at an arbitrarily small rate.

Note that Maurer's model is a generalization of the earlier settings in the sense that only the correlated information, but not the insecure communication is regarded as a resource. In particular, the communication can be interactive instead of only one-way, and the required amount of communication has no influence on the resulting secret-key rate. These apparently innocent modifications have dramatic consequences for the possibility of secret-key agreement.

1.2 The Secrecy Capacity and the Secret-Key Rate

The precise definitions of $C_S(P_{YZ|X})$ and of $S(X;Y||Z)$ will be given later, but we discuss here some of the most important bounds on these quantities. Roughly speaking, the possibility of secret-key agreement in Wyner's and Csiszár and Körner's models is restricted to situations for which Alice and Bob have an

initial advantage in terms of $P_{YZ|X}$, whereas interactive secret-key generation can be possible in settings that are initially much less favorable for the legitimate partners.

It was shown [10] that $C_S(P_{YZ|X}) \geq \max_{P_X} (I(X;Y) - I(X;Z))$, where the maximum is taken over all possible distributions P_X on the range $\mathcal{X}$ of X, and that equality holds whenever $I(X;Y) - I(X;Z)$ is non-negative for all distributions P_X. On the other hand, it is clear from the above bound that if $U \to X \to YZ$ is a Markov chain, then $C_S(P_{YZ|X}) \geq I(U;Y) - I(U;Z)$ is also true. If the maximization is extended in this way, then equality always holds:

$$C_S(P_{YZ|X}) = \max_{P_{UX} : U \to X \to YZ} (I(U;Y) - I(U;Z)) \tag{1}$$

is the main result of [10]. It is a consequence of equality (1) that Alice and Bob can generate a secret key by noisy one-way communication exactly in scenarios that provide an advantage of the legitimate partners over the opponent in terms of the broadcast channel's conditional distribution $P_{YZ|X}$.

The secret-key rate $S(X;Y||Z)$, as a function of P_{XYZ}, has been studied intensively. Lower and upper bounds on this quantity were derived, as well as necessary and sufficient criteria for the possibility of secret-key agreement [13], [15]. The lower bound

$$S(X;Y||Z) \geq \max\left[I(X;Y) - I(X;Z)\,,\, I(Y;X) - I(Y;Z)\right] \tag{2}$$

follows from equality (1) [13]. The important difference to the previous settings however is that secret-key agreement can even be possible when the right-hand side of inequality (2) is zero or negative. A special protocol phase, called *advantage distillation*, requiring feedback instead of only one-way communication, must be used in this case. On the other hand, it was shown in [15] that

$$S(X;Y||Z) \leq I(X;Y{\downarrow}Z) := \min_{P_{\overline{Z}|Z}} [I(X;Y|\overline{Z})]$$

holds, where $I(X;Y{\downarrow}Z)$ is called the *intrinsic conditional information between X and Y, given Z*. It has been conjectured in [15], based on some evidence, that $S(X;Y||Z) = I(X;Y{\downarrow}Z)$ holds for all P_{XYZ}, or at least that $I(X;Y{\downarrow}Z) > 0$ implies $S(X;Y||Z) > 0$. Most recent results suggest that the latter is true if $|\mathcal{X}| + |\mathcal{Y}| \leq 5$, but false in general [11].

1.3 Contributions of this Paper and Related Work

In all the mentioned scenarios, the conditions on the resulting secret key were too weak originally. As it is often done in information theory, all the involved quantities, including the information about the key the adversary is tolerated to obtain, were measured in terms of an *information rate*, which is defined as the ratio between the information quantity of interest and the number of independent repetitions of the underlying random experiment. Unfortunately, the total information the adversary gains about the resulting secret key is then, although

arbitrarily small in terms of the rate, not necessarily bounded, let alone negligibly small. The reason is that for a given (small) ratio $\epsilon > 0$, key agreement with respect to the security parameter ϵ is required to work only for strings of length N exceeding some bound $N_0(\epsilon)$ which can depend on ϵ. In particular, $N_0(\epsilon) \cdot \epsilon \to \infty$ for $\epsilon \to 0$ is possible. Clearly, this is typically unacceptable in a cryptographic scenario. For instance, the generated key cannot be used for a one-time-pad encryption if all parts of the message must be protected.

Motivated by these considerations, stronger definitions of the rates at which a secret key can be generated are given for the different scenarios. More specifically, it is required that the information the adversary obtains about the entire key be negligibly small in an *absolute* sense, not only in terms of a rate. In the setting of secret-key agreement by noiseless public discussion from common information it is additionally required that the resulting secret key, which must be equal for Alice and Bob with overwhelming probability, is perfectly-uniformly distributed.

The main result of this paper is a generic reduction from strong to weak key agreement with low communication complexity. As consequences of this, Theorems 1 and 2 state that both for the secrecy capacity and for the secret-key rate, strengthening the security requirements does not reduce the achievable key-generation rates. This is particularly interesting for the case of the secrecy capacity because in this model, all the communication must be carried out over the noisy channel. Recent advances in the theory of extractors allow for closing the gap between weak and strong security in this case.

An important consequence is that all previous results on $C_S(P_{YZ|X})$ and on $S(X;Y||Z)$, briefly described in Section 1.2, immediately carry over to the strong notions although they were only proved for the weaker definitions. The previous definitions were hence unnecessarily weak and can be entirely replaced by the new notions.

A basic technique used for proving the mentioned reduction is privacy amplification, introduced in [3], where we use both universal hashing and, as a new method in this context, extractors. A particular problem to be dealt with is to switch between (conditional) Shannon-, Rényi-, and min-entropy of random variables or, more precisely, of blocks of independent repetitions of random variables, and the corresponding probability distributions. A powerful tool for doing this are typical-sequences techniques.

Similar definitions of strong secrecy in key agreement have been proposed already by Maurer [14] (for the secret-key rate) and by Csiszár [9] (for the secrecy capacity). The authors have learned about the existence of the paper [9] (in Russian) only a few days before submitting this final version. In [14], the lower bound (3) on a slightly weaker variant of the strong secret-key rate than the one studied in this paper was proven. We present a substantially simplified proof here. In [9], a result similar to Theorem 2 was shown, using methods different from ours. More precisely, it was proved that the technique of [10] actually leads to a stronger secrecy than stated. In contrast to this, we propose a generic procedure for amplifying the secrecy of any information-theoretic key

agreement, requiring an amount of communication which is negligible compared to the length of the resulting key.

1.4 Entropy Measures and Variational Distance

We recall the definitions of some entropy measures needed in this paper. Let R be a discrete random variable with range $\mathcal{R}$. Then the *(Shannon) entropy* $H(R)$ is defined as[1] $H(R) := -\sum_{r\in\mathcal{R}} P_R(r)\cdot\log(P_R(r))$. The *Rényi entropy* $H_2(R)$ is $H_2(R) := -\log(\sum_{r\in\mathcal{R}} P_R^2(r))$. Finally, the *min-entropy* $H_\infty(R)$ is $H_\infty(R) := -\log\max_{r\in\mathcal{R}}(P_R(r))$. For two probability distributions P_X and P_Y on a set $\mathcal{X}$, the *variational distance* between P_X and P_Y is defined as $d(P_X, P_Y) := (\sum_{x\in\mathcal{X}} |P_X(x) - P_Y(x)|)/2$.

2 Secret-Key Agreement from Correlated Randomness

In this section we define a stronger variant of the secret-key rate of a distribution P_{XYZ} and show that this new quantity is equal to the previous, weak secret-key rate as defined in [13]. The protocol for strong key agreement consists of the following steps. First, weak key agreement is repeated many times. Then, so-called information reconciliation (error correction) and privacy amplification are carried out. These steps are described in Section 2.2. Of central importance for all the arguments made are typical-sequences techniques (Section 2.3). The main result of this section, the equality of the secret-key rates, is then proven in Section 2.4.

2.1 Definition of Weak and Strong Secret-Key Rates

Definition 1 **[13]** The *(weak) secret-key rate of X and Y with respect to Z*, denoted by $S(X;Y||Z)$, is the maximal $R \geq 0$ such that for every $\epsilon > 0$ and for all $N \geq N_0(\epsilon)$ there exists a protocol, using public communication over an insecure but authenticated channel, such that Alice and Bob, who receive $X^N = [X_1, \ldots, X_N]$ and $Y^N = [Y_1, \ldots, Y_N]$, can compute keys S and S', respectively, with the following properties. First, $S = S'$ holds with probability at least $1-\epsilon$, and second,

$$\frac{1}{N} I(S; C Z^N) \leq \epsilon \qquad \text{and} \qquad \frac{1}{N} H(S) \geq R - \epsilon$$

hold. Here, C denotes the collection of messages sent over the insecure channel by Alice and Bob. ○

As pointed out in Section 1.3, the given definition of the secret-key rate is unsatisfactorily and, as shown later, unnecessarily weak. We give a strong definition which bounds the information leaked to the adversary in an absolute sense and additionally requires the resulting key to be perfectly-uniformly distributed.

[1] All the logarithms in this paper are to the base 2, unless otherwise stated.

Definition 2 The *strong secret-key rate of X and Y with respect to Z*, denoted by $\overline{S}(X;Y||Z)$, is defined in the same way as $S(X;Y||Z)$ with the modifications that Alice and Bob compute strings S_A and S_B which are with probability at least $1-\epsilon$ both equal to a string S with the properties

$$I(S;CZ^N) \leq \epsilon \qquad \text{and} \qquad H(S) = \log|\mathcal{S}| \geq N \cdot (R-\epsilon) \; . \qquad \bigcirc$$

Obviously, $\overline{S}(X;Y||Z) \leq S(X;Y||Z)$ holds. It is the goal of this section to show equality of the rates for every distribution P_{XYZ}. Thus the attention can be totally restricted to the strong notion of secret-key rate.

2.2 Information Reconciliation and Privacy Amplification

In this section we analyze the two steps, called *information reconciliation* and *privacy amplification*, of a protocol allowing *strong* secret-key agreement whenever $I(X;Y) - I(X;Z) > 0$ or $I(Y;X) - I(Y;Z) > 0$ holds. More precisely, we show

$$\overline{S}(X;Y||Z) \geq \max\{\, I(X;Y) - I(X;Z)\,,\; I(Y;X) - I(Y;Z)\,\}\;. \qquad (3)$$

Assume $I(X;Y) > I(X;Z)$. The information-reconciliation phase of interactive error correction consists of the following step. For some suitable function $h : \mathcal{X}^N \to \{0,1\}^L$, Alice sends $h(X^N)$ to Bob for providing him (who knows Y^N) with a sufficient amount of information about X^N that allows him to reconstruct X^N with high probability. The existence of such a function (in a fixed universal class, see Definition 3) for L on the order of $N \cdot H(X|Y)$ is stated in Lemma 1, a weaker variant of which was formulated already in [14]. Note that this type of (one-way) information-reconciliation protocol is optimal with respect to the amount of exchanged information and efficient with respect to communication complexity, but not with respect to computational efficiency of Bob. There exist efficient interactive methods, which however leak more information to the adversary (see [4] for various results on information reconciliation).

Definition 3 [7] A class G of functions $g : \mathcal{A} \longrightarrow \mathcal{B}$ is *universal* if, for any distinct x_1 and x_2 in $\mathcal{A}$, the probability that $g(x_1) = g(x_2)$ holds is at most $1/|\mathcal{B}|$ when g is chosen at random from G according to the uniform distribution. $\bigcirc$

Example 1. [7] Let $1 \leq M \leq N$, let a be an element of $GF(2^N)$, and interpret $x \in \{0,1\}^N$ as an element of $GF(2^N)$ with respect to a fixed basis of the extension field over the prime field $GF(2)$. Consider the function $h_a : \{0,1\}^N \to \{0,1\}^M$ assigning to an argument x the first M bits (with respect to this basis representation) of the element ax of $GF(2^N)$, i.e., $h_a(x) := \mathrm{LSB}_M(a \cdot x)$. The class $\{h_a : a \in GF(2^N)\}$ is a universal class of 2^N functions mapping $\{0,1\}^N$ to $\{0,1\}^M$.

Lemma 1 *Let X and Y be random variables, and let $[(X_1,Y_1),\ldots,(X_N,Y_N)]$ be a block of N independent realizations of X and Y. Then for every $\epsilon > 0$ and*

$\epsilon' > 0$, for sufficiently large N, for every L satisfying $L/N > (1+\epsilon)H(X|Y)$, and for every universal class $\mathcal{H}$ of functions mapping $\mathcal{X}^N$ to $\{0,1\}^L$, there exists a function h in $\mathcal{H}$ such that $X^N = [X_1, \ldots, X_N]$ can be decoded from Y^N and $h(X^N)$ with error probability at most ϵ'.

The proof of Lemma 1 is omitted. See [4] for the proof of a closely related result.

In the second protocol phase, privacy amplification, Alice and Bob compress the mutual but generally highly-insecure string X^N to a shorter string S with virtually-uniform distribution and about which Eve has essentially no information. (Note that Eve's total information about X^N consists of Z^N and $h(X^N)$ at this point.) Bennett *et. al.* [2] have shown that universal hashing allows for distilling a virtually-secure string whose length is roughly equal to the Rényi entropy of the original string in Eve's view.

Lemma 2 [2] *Let W be a random variable with range $\mathcal{W}$, and let G be the random variable corresponding to the random choice, according to the uniform distribution, of a function out of a universal class of functions mapping $\mathcal{W}$ to $\{0,1\}^M$. Then $H(G(W)|G) \geq H_2(G(W)|G) \geq M - 2^{M-H_2(W)}/\ln 2$.*

Lemma 2 states that if Alice and Bob share a particular string S and Eve's information about S leads to the distribution $P_{S|U=u}$ (where u denotes the particular value of her information U) about which Alice and Bob know nothing except a lower bound t on the Rényi entropy, i.e., $H_2(S|U=u) \geq t$, then Alice and Bob can generate a secret key S' of roughly t bits. More precisely, if Alice and Bob compress S to a $(t-s)$-bit key for some security parameter $s > 0$, then Eve's total information about this key is exponentially small in s (see Figure 1).

A natural problem that arises when combining information reconciliation and privacy amplification with universal hashing is to determine the effect of the error-correction information (leaked also to the adversary) on the Rényi entropy of the partially-secret string, given Eve's information. The following result, which was shown by Cachin [5] as an improvement of an earlier result by Cachin and Maurer [6], states that leaking r physical bits of arbitrary side information about a string cannot reduce its Rényi entropy by substantially more than r, except with exponentially small probability.

Lemma 3 [5] *Let X and Q be random variables, and let $s > 0$. Then with probability at least $1 - 2^{-(s/2-1)}$, we have $H_2(X) - H_2(X|Q=q) \leq \log|\mathcal{Q}| + s$.*

2.3 Typical Sequences

In the following proofs we will make use of so-called typical-sequences arguments. Such arguments are based on the fact that if a large number of independent realizations of a random variable U is considered, then the actual probability of the particular outcome sequence is, with overwhelming probability, close to a certain "typical probability." There exist various definitions of typical sequences. The definition given below corresponds to a weak notion of typicality, dealing only with probabilities and not with the number of occurrences of the outcome symbols of the original random variable U in the sequence.

Definition 4 Let U be a random variable with probability distribution P_U and range $\mathcal{U}$, and let $N \geq 0$. Then a sequence $u = (u_1, u_2, \ldots, u_N) \in \mathcal{U}^N$ is called (weakly) δ-typical if $2^{-N(H(U)+\delta)} \leq P_{U^N}(u) \leq 2^{-N(H(U)-\delta)}$. ○

Lemma 4 states that if N is large enough, then U^N, distributed according to $P_{U^N} = P_U^N$ which corresponds to N independent realizations of U, is δ-typical with high probability. More precisely, the probability of the "non-typicality" event tends to zero faster than $1/N^2$. This follows immediately from Theorem 12.69 in [8].

Lemma 4 [8] *For all $\delta, \epsilon > 0$, we have $N \cdot (\mathrm{Prob}\,[U^N$ is not δ-typical$])^{1/2} < \epsilon$ for sufficiently large N.*

As a first step towards proving equality of the secret-key rates with respect to the weak and strong definitions, we show that the weak definition can be extended by an additional condition requiring that the resulting key is close-to-uniformly distributed. More precisely, Lemma 5 states that the condition

$$\frac{1}{N} H(S) \geq \frac{1}{N} \log |\mathcal{S}| - \epsilon \tag{4}$$

can be included into the definition of $S(X;Y||Z)$ without effect on its value. (Note that the condition (4) is much weaker than the uniformity condition in the definition of $\overline{S}(X;Y||Z)$.)

Lemma 5 *Let the* uniform *(weak) secret-key rate $S_u(X;Y||Z)$ be defined similarly to $S(X;Y||Z)$, but with the additional condition (4). Then $S_u(X;Y||Z) = S(X;Y||Z)$ holds.*

Proof. The idea is to carry out the key-generation procedure independently many times and to apply data compression. More precisely, secret-key agreement with respect to the definition of $S(X;Y||Z)$ is repeated M times. Clearly, we can assume that the resulting triples $[S_i, S_i', (Z^N C)_i]$ are independent for different values of i and can be considered as the random variables in a new random experiment. When repeating this experiment for a sufficient number of times and applying data compression to the resulting sequence of keys, thereby using that with high probability both $[S_1, S_2, \ldots]$ and $[S_1', S_2', \ldots]$ are typical sequences, one finally obtains key agreement that ends up in a highly-uniformly distributed key.

Let $R := S(X;Y||Z)$. We show that for any $\epsilon > 0$ (and for a sufficiently large number of realizations of the random variables) secret-key agreement at a rate at least $R - \epsilon$ is possible even with respect to the stronger definition which includes the uniformity condition (4).

For parameters $\epsilon' > 0$ and $N > 0$, both to be determined later, let secret-key agreement (not necessarily satisfying the new condition) be carried out M times independently. Let S_i and S_i', $i = 1, \ldots, M$, be the generated keys, and let C_i and $(Z^N)_i$ be the corresponding collection of messages sent over the public channel and the realizations of Z that Eve obtains, respectively. Then the

triples $[S_i, S_i', (Z^N C)_i]$, $i = 1, \ldots, M$, are statistically independent and identically distributed. According to the definition of $S(X;Y||Z)$, we can achieve for every i

$$H(S_i)/N \geq R - \epsilon' \ , \quad \text{Prob}\,[S_i \neq S_i'] < \tilde{\epsilon} \ , \quad \text{and} \quad I(S_i; (Z^N C)_i)/N < \epsilon' \ , \qquad (5)$$

where the constant $\tilde{\epsilon}$ will be specified later. (Note that in order to make only $\tilde{\epsilon}$ smaller and to leave ϵ' unchanged, it is not necessary to increase N because the second condition in (5) is stricter for larger N: The key can be subdivided into smaller pieces at the end, and for every such piece, the error probability is at most $\tilde{\epsilon}$.)

Using the fact that for all $\alpha > 0$ and $\delta > 0$, the event $\mathcal{E}(\delta)$ that the sequence $[S_1, S_2, \ldots, S_M]$ is δ-typical has probability at least $1 - \alpha$ for sufficiently large M, we can transform the key vector $[S_1, \ldots, S_M]$ into an almost-uniformly distributed key T as follows. If $\mathcal{E}(\delta)$ occurs, then let $T := [S_1, \ldots, S_M]$, otherwise $T := \Delta$ for some failure symbol Δ. The key T' is computed from $[S_1', \ldots, S_M']$ analogously. Then, T and T' have the following properties. First, $\log|\mathcal{T}| \leq M(H(S) + \delta) + 1$ and $H(T) \geq (1 - \alpha)M(H(S) - \delta)$ follow from the definitions of T and of δ-typical sequences. For the quantities occurring in the definition of $S_u(X;Y||Z)$, we hence obtain

$$H(T)/MN \geq (1 - \alpha)(R - \epsilon' - \delta/N) \ , \qquad (6)$$

$$\text{Prob}\,[T \neq T'] < M\tilde{\epsilon} \ , \qquad (7)$$

$$I(T; (Z^N C)_{i=1,\ldots,M})/MN < \epsilon' \ , \qquad (8)$$

$$(\log|\mathcal{T}| - H(T))/MN \leq \alpha R + 2\delta/N \ . \qquad (9)$$

Because of Lemma 4 one can choose, for every sufficiently large N, constants α, δ, and ϵ' such that $\text{Prob}\,[\overline{\mathcal{E}(\delta)}] < \alpha$ (where $\overline{\mathcal{E}(\delta)}$ stands for the complementary event of $\mathcal{E}(\delta)$) for this choice of M, and such that the expressions on the right-hand sides of (8) and (9) are smaller than ϵ, whereas the right-hand side of (6) is greater than $R - \epsilon$. Finally, $\tilde{\epsilon}$ can be chosen as ϵ/M, such that the condition (7) is also satisfied.

We conclude that the uniform secret-key rate $S_u(X;Y||Z)$ is at least $R = S(X;Y||Z)$. This concludes the proof. □

Lemma 6 links Rényi entropy with typicality of sequences (and hence Shannon entropy). More precisely, the conditional Rényi entropy of a sequence of realizations of random variables is close to the length of the sequence times the conditional Shannon entropy of the original random variables, given a certain typicality event which occurs with high probability. Related arguments already appeared in [12] and [5].

Lemma 6 *Let P_{XZ} be the joint distribution of two random variables X and Z, let $0 < \delta \leq 1/2$, and let N be an integer. The event $\mathcal{F}(\delta)$ is defined as follows: First, the sequences x^N and $(x,z)^N$ must both be δ-typical, and second, z^N must be such that the probability, taken over $(x')^N$ according to the distribution $P_{X^N|Z^N=z^N}$, that $(x',z)^N$ is δ-typical is at least $1 - \delta$. Then we have*

$N \cdot \mathrm{Prob}[\overline{\mathcal{F}(\delta)}] \to 0$ *for* $N \to \infty$, *and* $H_2(X^N|Z^N = z^N, \mathcal{F}(\delta)) \geq N(H(X|Z) - 2\delta) + \log(1-\delta)$.

Proof. Because of Lemma 4, the event, denoted by $\mathcal{E}(\delta)$, that both x^N and $(x,z)^N$ are δ-typical has probability at least $1-\delta^2$ for some $N = N(\delta)$ with $N(\delta)\cdot\delta \to 0$. For this value of N, z^N has with probability at least $1-\sqrt{\delta^2} = 1-\delta$ the property that $(x',z)^N$ is δ-typical with probability at least $1-\sqrt{\delta^2} = 1-\delta$, taken over $(x')^N$ distributed according to $P_{X^N|Z^N=z^N}$. Hence the probability of the complementary event $\overline{\mathcal{F}(\delta)}$ of $\mathcal{F}(\delta)$ is at most $\delta^2+\delta$, thus $N\cdot\mathrm{Prob}[\overline{\mathcal{F}(\delta)}] \to 0$.

On the other hand, given that z^N and $(x',z)^N$ are δ-typical, we can conclude that

$$2^{-N(H(X|Z)+2\delta)} \leq P_{X^N|Z^N}((x')^N, z^N) \leq 2^{-N(H(X|Z)-2\delta)}$$

holds. For a fixed value z^N, the Rényi entropy of X^N, given the events $Z^N = z^N$ and $\mathcal{F}(\delta)$, is lower bounded by the Rényi entropy of a uniform distribution over a set with $(1-\delta)\cdot 2^{N(H(X|Z)-2\delta)}$ elements: $H_2(X^N|Z^N = z^N, \mathcal{F}(\delta)) \geq N(H(X|Z) - 2\delta) + \log(1-\delta)$. □

2.4 Equality of Weak and Strong Rates

In this section we prove the lower bound (3) on $\overline{S}(X;Y||Z)$ and the first main result, stating that the weak and strong secret-key rates are equal for every distribution. A result closely related to Lemma 7 was proved as the main result in [14]. We give a much shorter and simpler proof based on the results in Sections 2.2 and 2.3.

Lemma 7 *For all* P_{XYZ}, $\overline{S}(X;Y||Z) \geq \max\{\, I(X;Y) - I(X;Z)\,,\, I(Y;X) - I(Y;Z)\,\}$ *holds.*

Proof. We only prove that $I(X;Y) - I(X;Z) = H(X|Z) - H(X|Y)$ is an achievable rate. The statement then follows by symmetry.

Let $\epsilon > 0$, and let $\Delta > 0$ be determined later. We show that for the parameter ϵ, and for sufficiently large N, there exists a protocol which achieves the above rate (reduced by ϵ). Let $\delta < \epsilon/4$ and $\alpha < \Delta/(2H(X))$ be constants, and let $\mathcal{F}(\delta)$ be the event as defined in Lemma 6. Because of Lemma 6 we have for sufficiently large N that $N \cdot \mathrm{Prob}[\overline{\mathcal{F}(\delta)}] < \alpha$. On the other hand,

$$H_2(X^N|Z^N = z^N, \mathcal{F}(\delta)) \geq N \cdot (H(X|Z) - 2\delta) + \log(1-\delta)$$

holds.

The protocol now consists of two messages sent from Alice to Bob, one for information reconciliation and the other one for privacy amplification (see Section 2.2). Let $\beta < \epsilon/(2H(X|Y))$ be a positive constant. According to Lemma 1 there exists for sufficiently large N a function $h : \mathcal{X}^N \to \{0,1\}^L$, where $L := \lceil (1+\beta)NH(X|Y) \rceil$, such that X^N can be determined from Y^N and $h(X^N)$ with probability at least $1-\epsilon/2$ (using the optimal strategy). Clearly, the value

$h(X^N)$ reduces Eve's uncertainty in terms of Rényi entropy about X^N. We conclude from Lemma 3 for $s := 2\log(2NH(X)/\Delta) + 2$ that with probability at least $1 - 2^{-(s/2-1)}$,

$$\begin{aligned} H_2(X^N|Z^N = z^N, h(X^N) = h(x^N), \mathcal{F}(\delta)) \qquad (10)\\ \geq N \cdot (H(X|Z) - 2\delta) + \log(1-\delta) - [(1+\beta) \cdot N \cdot H(X|Y) + 1 + s] \\ = N \cdot (H(X|Z) - H(X|Y)) - 2\delta N - \beta N H(X|Y) - 1 - s + \log(1-\delta) \\ =: \ Q \, . \end{aligned}$$

Finally, Alice and Bob use privacy amplification to transform their mutual information X^N into a highly-secret string $\tilde{S}$. Let $r := \lceil \log N \rceil$, and let $M := Q - r$ be the length of the resulting string $\tilde{S}$. If G is the random variable corresponding to the random choice of a universal hash function mapping $\mathcal{X}^N \to \{0,1\}^M$, and if $\tilde{S} := G(X^N)$, then we have $H(\tilde{S}|Z^N = z^N, h(X^N) = h(x^N), G, \mathcal{F}(\delta)) \geq M - 2^{-r}/\ln 2$ under the condition that inequality (10) holds. Hence we get for sufficiently large N

$$\begin{aligned} H(\tilde{S}|Z^N, h(X^N), G) &\geq (\mathrm{Prob}\,[\mathcal{F}(\delta)] - 2^{-(s/2-1)})(M - 2^{-r}/\ln 2) \\ &\geq M - 2^{-r}/\ln 2 - (\mathrm{Prob}\,[\overline{\mathcal{F}(\delta)}] + 2^{-(s/2-1)}) \cdot N \cdot H(X) \\ &> \log|\tilde{\mathcal{S}}| - \Delta \end{aligned}$$

by definition of r, α, and s. Let now S be a "uniformization" of $\tilde{S}$ (i.e., a random variable S with range $\mathcal{S} = \tilde{\mathcal{S}} = \{0,1\}^M$ that is generated by sending $\tilde{S}$ over some channel characterized by $P_{S|\tilde{S}}$, that is uniformly distributed, and that minimizes $\mathrm{Prob}\,[S \neq \tilde{S}]$ among all random variables with these properties). For $C = [h(X^N), G]$ and sufficiently small Δ, we can then conclude that

$$I(S; Z^N C) < \epsilon \ , \quad H(S) = \log|\mathcal{S}| \ , \quad \text{and} \quad \mathrm{Prob}\,[S' \neq S] < \epsilon$$

holds because of $H(\tilde{S}) \geq H(\tilde{S}|Z^N, h(X^N), G)$. The achievable key-generation rate with this protocol is hence at least

$$H(X|Z) - H(X|Y) - 2\delta - \beta H(X|Y) \geq I(X;Y) - I(X;Z) - \epsilon \, .$$

Thus we obtain $\overline{S}(X;Y||Z) \geq I(X;Y) - I(X;Z)$, and this concludes the proof. □

Theorem 1 is the main result of this section and states that the strong secret-key rate $\overline{S}(X;Y||Z)$ is always equal to the weak secret-key rate $S(X;Y||Z)$.

Theorem 1 *For all distributions P_{XYZ}, we have $\overline{S}(X;Y||Z) = S(X;Y||Z)$.*

Proof. Clearly, $\overline{S}(X;Y||Z) \leq S(X;Y||Z)$ holds. Let $R := S(X;Y||Z)$, and let $\epsilon > 0$. According to the definition of the secret-key rate $S(X;Y||Z)$ (and because of Lemma 5), there exists, for sufficiently large N, a protocol with the following properties: Alice and Bob know, at the end of the protocol, strings S and S' such that $H(S) \geq NR - N\epsilon$, $\mathrm{Prob}\,[S \neq S'] < \epsilon$, $I(S; Z^N C) \leq N\epsilon$, and $H(S) \geq$

$\log|\mathcal{S}| - N\epsilon$ hold. From these equations, we can conclude by Fano's inequality [8] that

$$\begin{aligned} I(S;S') = H(S) - H(S|S') &\geq H(S) - h(\text{Prob}\,[S \neq S']) - \text{Prob}\,[S \neq S'](H(S) + N\epsilon) \\ &> H(S)(1-\epsilon) - h(\epsilon) - N\epsilon^2 \geq NR - NR\epsilon - N\epsilon - h(\epsilon) \end{aligned}$$

holds (where h is the binary entropy function), hence $I(S;S') - I(S;Z^N C) \geq NR - NR\epsilon - 2N\epsilon - h(\epsilon)$. Let us now consider the random experiment $[S, S', Z^N C]$ (where we assume that the realizations are independent). By applying Lemma 7 to the new distribution, we get

$$\overline{S}(X;Y||Z) \geq S(S;S'||Z^N C)/N \geq (I(S;S') - I(S;Z^N C))/N \geq R - R\epsilon - 2\epsilon - h(\epsilon)/N$$

for every $\epsilon > 0$, thus $\overline{S}(X;Y||Z) \geq S(X;Y||Z)$. □

3 Strengthening the Secrecy Capacity

This section is concerned with the model introduced by Wyner [18] and the generalization thereof by Csiszár and Körner [10], which served as a motivation for Maurer's [13] scenario treated in Section 2. In analogy to the weak definition of the secret-key rate, the original definition of the secrecy capacity is not satisfactory because the total amount of information about the resulting key that the adversary obtains can be unbounded. We show that also the definition of the secrecy capacity can be strengthened, without any effect on the actual value of this quantity, in the sense that the total amount of information the adversary obtains about the secret key is negligibly small. More precisely, we develop a generic reduction of strong to weak key agreement by one-way communication and such that the total length of the additional messages is negligible compared to the length of the resulting string. The low-communication-complexity condition is necessary because in this model, in contrast to the model of Section 2, no communication is "for free." More precisely, the noisy broadcast channel must be used for the entire communication (i.e., for the exchange of all the error-correction and privacy-amplification information), which at first sight appears to reduce the maximal achievable key-generation rate. However, the use of extractors (see Section 3.2) instead of universal hashing for privacy amplification allows to keep the fraction of channel uses for communicating the error-correction and privacy-amplification messages arbitrarily small.

3.1 Definition of the Secrecy Capacity $C_S(P_{YZ|X})$

Assume that the parties Alice and Bob, and the adversary Eve, are connected by a noisy broadcast channel with conditional output distribution $P_{YZ|X}$ [10]. (Wyner's wire-tap channel corresponds to the special case where $P_{YZ|X} = P_{Y|X} \cdot P_{Z|Y}$ holds.) The ability of generating mutual secret information was quantified as follows.

Definition 5 [18], [10] Consider a memoryless broadcast channel characterized by the conditional joint distribution $P_{YZ|X}$. The *secrecy capacity* $C_S(P_{YZ|X})$ of the channel is the maximal real number $R \geq 0$ such that for every $\epsilon > 0$, for sufficiently large N, and for $K := \lfloor (R-\epsilon)N \rfloor$, there exists a possibly probabilistic (i.e., additionally depending on some random bits) encoding function $e : \{0,1\}^K \to \mathcal{X}^N$ together with a decoding function $d : \mathcal{Y}^N \to \{0,1\}^K$ such that if S is uniformly distributed over $\{0,1\}^K$, we have for $X^N = e(S)$ and $S' := d(Y^N)$ that $\mathrm{Prob}[S' \neq S] < \epsilon$ and

$$\frac{1}{K} H(S|Z^N) > 1 - \epsilon \tag{11}$$

hold. ○

3.2 Privacy Amplification with Extractors

In order to show that the notion of secrecy used in the definition of C_S can be strengthened without reducing the secrecy capacity of the broadcast channel, we need a different technique for privacy amplification, requiring less information to be transmitted, namely only an asymptotically arbitrarily small fraction of the number of bits of the partially-secure string to be compressed. (Otherwise, the channel applications needed for sending this message would reduce the achievable key-generation rate.) We show that such a technique is given by so-called *extractors*. Roughly speaking, an extractor allows to efficiently isolate the randomness of some source into virtually-random bits, using a small additional number of perfectly-random bits as a catalyst, i.e., in such a way that these bits reappear as a part of the almost-uniform output. Extractors are of great importance in theoretical computer science, where randomness is often regarded as a resource. They have been studied intensively in the past years by many authors. For an introduction and some constructions, see [16], [17], and the references therein.

Recent results, described below, show that extractors allow, using only a small amount of true randomness, to distill (almost) the entire randomness, measured in terms of H_∞, of some string into an almost-uniformly distributed string. A disadvantage of using extractors instead of universal hashing for privacy amplification is that a string of length only roughly equal to the *min*-entropy instead of the generally greater *Rényi* entropy of the original random variable can be extracted. However, this drawback has virtually no effect in connection with typical sequences, i.e., almost-uniform distributions, for which all the entropy measures are roughly equal.

Definition 6 A function $E : \{0,1\}^N \times \{0,1\}^d \to \{0,1\}^r$ is called a (δ', ϵ')-extractor if for any random variable T with range $\mathcal{T} \subseteq \{0,1\}^N$ and min-entropy $H_\infty(T) \geq \delta' N$, the variational distance of the distribution of $[V, E(T,V)]$ to the uniform distribution over $\{0,1\}^{d+r}$ is at most ϵ' when V is independent of T and uniformly distributed in $\{0,1\}^d$. ○

The following theorem was proved in [17]. It states that there exist extractors which distill virtually all the min-entropy out of a weakly-random source, thereby

requiring only a small (i.e., "poly-logarithmic") number of truly-random bits. Note that Definition 6, and hence the statement of Lemma 8, is formally slightly stronger than the corresponding definition in [17] because it not only requires that the length of the extractor output is roughly equal to the min-entropy of the source plus the number of random bits, but that these bits even reappear as a part of the output. It is not difficult to see that the extractors described in [17] have this additional property.

Lemma 8 [**17**] *For every choice of the parameters N, $0 < \delta' < 1$, and $\epsilon' > 0$, there exists a (δ', ϵ')-extractor $E : \{0,1\}^N \times \{0,1\}^d \to \{0,1\}^{\delta' N - 2\log(1/\epsilon') - O(1)}$, where $d = O((\log(N/\epsilon'))^2 \log(\delta' N))$.*

Lemma 9, which is a consequence of Lemma 8, is what we need in the proof of Theorem 2. The statement of Lemma 9 is related to Lemma 2, where universal hashing is replaced by extractors, and min-entropy must be used instead of Rényi entropy (see Figure 1).

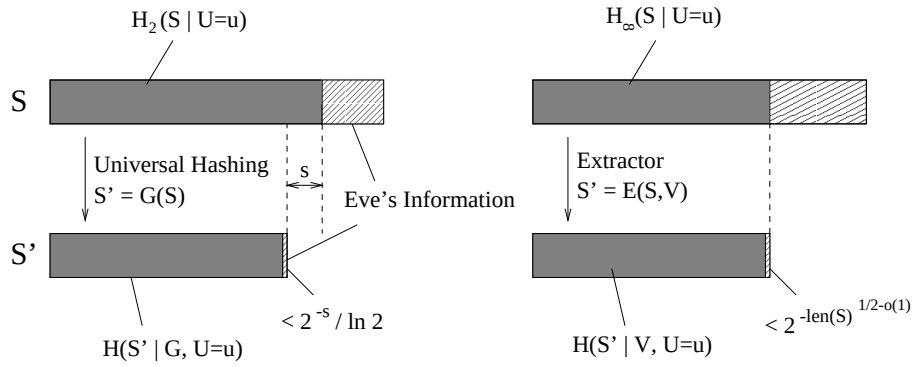


Fig. 1. Privacy Amplification: Universal Hashing Versus Extractors

Lemma 9 *Let $\delta', \Delta_1, \Delta_2 > 0$ be constants. Then there exists, for all sufficiently large N, a function $E : \{0,1\}^N \times \{0,1\}^d \to \{0,1\}^r$, where $d \leq \Delta_1 N$ and $r \geq (\delta' - \Delta_2)N$, such that for all random variables T with $\mathcal{T} \subseteq \{0,1\}^N$ and $H_\infty(T) > \delta' N$, we have*

$$H(E(T,V)|V) \geq r - 2^{-N^{1/2-o(1)}} . \tag{12}$$

Proof. Let $\epsilon'(N) := 2^{-\sqrt{N}/\log N}$. Then there exists N_0 such that for all $N \geq N_0$ we have a (δ', ϵ')-extractor E, mapping $\{0,1\}^{N+d}$ to $\{0,1\}^r$, where $d \leq \Delta_1 N$ (note that $d = O(N/\log N)$ holds for this choice of ϵ') and $r \geq (\delta' - \Delta_2)N$. By definition, this means that for a uniformly distributed d-bit string V and if

$H_\infty(T) \geq \delta' N$, the distance of the distribution of $[V, E(T,V)]$ to the uniform distribution U_{d+r} over $\{0,1\}^{d+r}$ is at most $\epsilon' = 2^{-\sqrt{N}/\log N}$. Because

$$d([V, E(T,V)], U_{d+r}) = \mathrm{E}_V[d(E(T,V), U_r)] \leq \epsilon'$$

holds for uniformly distributed V, the distance of the distribution of $E(T,v)$ to the uniform distribution U_r (over $\{0,1\}^r$) is at most $\sqrt{\epsilon'}$ with probability at least $1-\sqrt{\epsilon'}$ over v, i.e.,

$$P_V\left[d\left(E(T,V), U_r\right) \leq 2^{-\sqrt{N}/2\log N}\right] \geq 1 - 2^{-\sqrt{N}/2\log N} \ . \tag{13}$$

Inequality (12) follows from (13) in a straight-forward way. □

Lemma 3 gives an upper bound on the effect of side information on the Rényi entropy of a random variable, and thus links information reconciliation and privacy amplification with universal hashing. We now need a similar result with respect to min-entropy H_∞. The proof of Lemma 10 is straight-forward and therefore omitted.

Lemma 10 *Let X and Q be random variables, and let $s > 0$. Then with probability at least $1-2^{-s}$, we have $H_\infty(X) - H_\infty(X|Q=q) \leq \log|\mathcal{Q}| + s$.*

3.3 The Strong Secrecy Capacity $\overline{C_S}(P_{YZ|X})$

In this section we show that the definition of secrecy capacity in Csiszár and Körner's, hence also in Wyner's, model can be strengthened similarly to the weak and strong notions of secret-key rate: Not the rate, but the total amount of leaked information is negligible. Note that an additional uniformity condition is not necessary here since already the definition of C_S requires the key to be perfectly-uniformly distributed. Theorem 2 is the main result of this section.

Definition 7 For a distribution $P_{YZ|X}$, the *strong secrecy capacity* $\overline{C_S}(P_{YZ|X})$ is defined similarly to $C_S(P_{YZ|X})$, where the secrecy condition (11) is replaced by the stronger requirement $H(S|Z^N) > K - \epsilon$. ○

Theorem 2 *For all distributions $P_{YZ|X}$, we have $\overline{C_S}(P_{YZ|X}) = C_S(P_{YZ|X})$.*

Proof. The idea of the proof is to repeat the (weak) key generation a number of times and to compute from the block of resulting weak keys a secure string satisfying the stronger definition of secrecy capacity. More precisely, this is done by information reconciliation as described in Section 2.2, and by privacy amplification with extractors. Since the parties have, in contrast to the public-discussion model, no access to a noiseless public channel, all the error-correction and privacy-amplification information must be sent over the noisy channel specified by the conditional marginal distribution $P_{Y|X}(y,x) = \sum_{z\in\mathcal{Z}} P_{YZ|X}(y,z,x)$. However, the use of extractors instead of universal hashing for privacy amplification allows to keep the fraction of channel uses required for this communication

negligibly small. This is precisely what is needed for showing equality of C_S and $\overline{C_S}$.

Let $R := C_S(P_{YZ|X})$. For a constant $\epsilon' > 0$ and integers M and N to be determined later, assume that the key-generation procedure, with respect to the (weak) secrecy capacity C_S and parameters ϵ' and N, is repeated independently M times. Let $S^M := [S_1, \ldots, S_M]$ and $(S')^M := [S'_1, \ldots, S'_M]$ be the generated keys of Alice and Bob, respectively, and let $K = \lfloor (R - \epsilon')N \rfloor$ be the length of (the binary strings) S_i and S'_i. From the fact that $\mathrm{Prob}\,[S_i \neq S'_i] < \epsilon'$ holds we conclude, by Fano's inequality, $H(S_i|S'_i) \leq \epsilon' K + 1$ for all i, hence $H(S^M|(S')^M) \leq M(\epsilon' K + 1)$.

For constants $\Delta_1, \Delta_2 > 0$, we conclude from Lemma 1 that there exists an error-correction-information function $h : (\{0,1\}^K)^M \longrightarrow \{0,1\}^{\lceil (1+\Delta_1)M(\epsilon' K+1)\rceil}$ such that S^M can be determined from $(S')^M$ and $h(S^M)$ with probability at least $1 - \Delta_2$ for sufficiently large M. Hence $\lceil (1 + \Delta_1)M(\epsilon' K + 1)\rceil$ message bits have to be transmitted over the channel $P_{Y|X}$ for error correction (see below).

According to the definition of the (weak) secrecy capacity C_S, we have $H(S_i|Z_i^N) \geq K(1 - \epsilon')$. For $\delta > 0$, let the event $\mathcal{F}(\delta)$, with respect to the random variables S and Z^N, be defined as in Lemma 6. For every $\alpha > 0$ we can achieve, for arbitrarily large (fixed) N and M, $MK \cdot \mathrm{Prob}\,[\overline{\mathcal{F}(\delta)}] < \alpha$ and

$$H_\infty(S^M|(Z^N)^M = (z^N)^M, \mathcal{F}(\delta)) \geq M(K(1 - \epsilon') - 2\delta) + \log(1 - \delta) \ .$$

The reason is that the statement of Lemma 6 also holds for the min-entropy H_∞ instead of H_2. The proof of this variant is exactly the same because it is ultimately based on uniform distributions, for which H_2 and H_∞ (and also H) are equal.

Let us now consider the effect of the error-correction information (partially) leaked to the adversary. According to Lemma 10, we have for $s > 0$ with probability at least $1 - 2^{-s}$

$$\begin{aligned} H_\infty(S^M|(Z^N)^M = (z^N)^M, h(S^M) = h(s^M), \mathcal{F}(\delta)) \\ \geq\ & M(K(1 - \epsilon') - 2\delta) + \log(1 - \delta) - \lceil (1 + \Delta_1)M(\epsilon' K + 1)\rceil - s \\ \geq\ & MK(1 - \Delta_3) \end{aligned} \tag{14}$$

for some constant Δ_3 that can be made arbitrarily small by choosing N large enough, $s := \lceil \log M \rceil$, and Δ_1 as well as ϵ' small enough.

Let now for constants $\Delta_4, \Delta_5 > 0$ and sufficiently large M an extractor function E be given according to Lemma 9, i.e., $E : \{0,1\}^{MK} \times \{0,1\}^d \to \{0,1\}^r$ with $d \leq \Delta_4 MK$ and $r \geq MK(1 - \Delta_3 - \Delta_5)$ such that, for $\tilde{S} := E(S^M, V)$, the inequality

$$H(\tilde{S}|(Z^N)^M = (z^N)^M, h(S^M) = h(s^M), V, \mathcal{F}(\delta)) \geq r - 2^{-(MK)^{1/2 - o(1)}}$$

holds if V is uniformly distributed in $\{0,1\}^d$. Let S' be the key computed in the same way by Bob (where the random bits V are sent over to him by Alice using the channel $P_{Y|X}$ with an appropriate error-correcting code).

The resulting key $\tilde{S}$ of Alice is now close-to-uniformly, but not perfectly-uniformly distributed. Given the events $\mathcal{F}(\delta)$ and that inequality (14) holds, we have $H(\tilde{S}) \geq r - 2^{-(MK)^{1/2-o(1)}}$.

Let now, as in the proof of Lemma 7, S be the "uniformization" of $\tilde{S}$ (the random variable which is uniformly distributed in $\{0,1\}^r$ and jointly distributed with $\tilde{S}$ in such a way that $\mathrm{Prob}\,[S \neq \tilde{S}]$ is minimized). It is clear that for any $\Delta_6 > 0$, $\mathrm{Prob}\,[S \neq \tilde{S}] < \Delta_6$ can be achieved for sufficiently large M.

Let us finally consider the number of channel uses necessary for communicating the information for information reconciliation and privacy amplification. The number of bits to be transmitted is, according to the above, at most $\lceil (1+\Delta_1)M(\epsilon' K + 1)\rceil + \Delta_4 MK$. It is an immediate consequence of Shannon's channel-coding theorem (see for example [8]) that for arbitrary $\Delta_7, \Delta_8 > 0$ and sufficiently large M, the number of channel uses for transmitting these messages can be less than

$$\frac{MK((1+\Delta_1)\epsilon' + \Delta_4) + (1+\Delta_1)M + 1}{C(P_{Y|X}) - \Delta_7}$$

(where $C(P_{Y|X})$ is the capacity of the channel $P_{Y|X}$ from Alice to Bob), keeping the probability of a decoding error below Δ_8. Note that $C(P_{Y|X}) > 0$ clearly holds when $C_S(P_{YZ|X}) > 0$. (If $C(P_{Y|X}) = 0$, the statement of the theorem is hence trivially satisfied.) Thus the total number of channel uses for the entire key generation can be made smaller than $MN(1+\Delta_9)$ for arbitrarily small $\Delta_9 > 0$ and sufficiently large N.

From the above we can now conclude that S is a perfectly-uniformly distributed string of length $r = (1-o(1))RL$, where $L = (1+o(1))MN$ is the total number of channel uses. Furthermore, we have by construction $\mathrm{Prob}\,[S' \neq S] = o(1)$ and finally

$$\begin{aligned} H(S|Z^L) = H(S) - I(S;Z^L) &\geq H(S) - I(\tilde{S};Z^L) \\ &= r - 2^{-(MK)^{1/2-o(1)}} - r\cdot(2^{-s} + \mathrm{Prob}\,[\overline{\mathcal{F}(\delta)}]) = r - o(1)\,. \end{aligned} \tag{15}$$

The inequality in (15) holds because $Z^L \to \tilde{S} \to S$ is a Markov chain and because of the data-processing lemma [8]. Hence the achievable rate with respect to the strong secrecy-capacity definition is of order $(1-o(1))R = (1-o(1))C_S(P_{YZ|X})$, thus $\overline{C_S}(P_{YZ|X}) = C_S(P_{YZ|X})$ holds. □

4 Concluding Remarks

The fact that previous security definitions of information-theoretic key agreement in the noisy-channel models by Wyner [18] and Csiszár and Körner [10] and the correlated-randomness settings of Maurer [13] and Ahlswede–Csiszár [1] are unsatisfactory is a motivation for studying much stronger definitions which tolerate the adversary to obtain only a negligibly small amount of information

about the generated key. We have shown, by a generic reduction with low communication complexity and based on extractor functions, that in all these models, the achievable key-generation rates with respect to the weak and strong definitions are asymptotically identical. Therefore, the old notions can be entirely replaced by the new definitions.

References

1. R. Ahlswede and I. Csiszár, Common randomness in information theory and cryptography – Part I: secret sharing, *IEEE Transactions on Information Theory*, Vol. 39, No. 4, pp. 1121–1132, 1993.
2. C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer, Generalized privacy amplification, *IEEE Transactions on Information Theory*, Vol. 41, No. 6, pp. 1915–1923, 1995.
3. C. H. Bennett, G. Brassard, and J.-M. Robert, Privacy amplification by public discussion, *SIAM Journal on Computing*, Vol. 17, pp. 210–229, 1988.
4. G. Brassard and L. Salvail, Secret-key reconciliation by public discussion, *Advances in Cryptology - EUROCRYPT '93*, Lecture Notes in Computer Science, Vol. 765, pp. 410–423, Springer-Verlag, 1994.
5. C. Cachin, *Entropy measures and unconditional security in cryptography*, Ph. D. Thesis, ETH Zurich, Hartung-Gorre Verlag, Konstanz, 1997.
6. C. Cachin and U. M. Maurer, Linking information reconciliation and privacy amplification, *Journal of Cryptology*, Vol. 10, No. 2, pp. 97–110, 1997.
7. J. L. Carter and M. N. Wegman, Universal classes of hash functions, *Journal of Computer and System Sciences*, Vol. 18, pp. 143–154, 1979.
8. T. M. Cover and J. A. Thomas, *Elements of information theory*, Wiley Series in Telecommunications, 1992.
9. I. Csiszár, Almost independence and secrecy capacity (in Russian), in *Problems of Information Transmission (PPI)*, Vol. 32, No. 1, pp. 48–57, 1996.
10. I. Csiszár and J. Körner, Broadcast channels with confidential messages, *IEEE Transactions on Information Theory*, Vol. 24, No. 3, pp. 339–348, 1978.
11. N. Gisin and S. Wolf, Linking classical and quantum key agreement: is there "bound information"?, manuscript, 2000.
12. J. Håstad, R. Impagliazzo, L. Levin, and M. Luby, *Construction of a pseudo-random generator from any one-way function*, ICSI Tech. Rep. 91–068, 1991.
13. U. M. Maurer, Secret key agreement by public discussion from common information, *IEEE Transactions on Information Theory*, Vol. 39, No. 3, pp. 733–742, 1993.
14. U. M. Maurer, The strong secret key rate of discrete random triples, in *Communication and Cryptography – Two Sides of One Tapestry*, Kluwer Academic Publishers, pp. 271–285, 1994.
15. U. M. Maurer and S. Wolf, Unconditionally secure key agreement and the intrinsic conditional information, *IEEE Transactions on Information Theory*, Vol. 45, No. 2, pp. 499–514, 1999.
16. L. Trevisan, Construction of Extractors Using Pseudorandom Generators, *Proc. of the 31st Symposium on Theory of Computing (STOC)*, ACM, pp. 141–148, 1999.
17. S. P. Vadhan, Extracting all the randomness from a weakly random source, *Electronic Colloquium on Computational Complexity*, Tech. Rep. TR98–047, 1998.
18. A. D. Wyner, The wire-tap channel, *Bell System Technical Journal*, Vol. 54, No. 8, pp. 1355–1387, 1975.

New Attacks on PKCS#1 v1.5 Encryption

Jean-Sébastien Coron[1,3], Marc Joye[2], David Naccache[3], and Pascal Paillier[3]

[1] École Normale Supérieure
45 rue d'Ulm, 75005 Paris, France
coron@clipper.ens.fr
[2] Gemplus Card International
Parc d'Activités de Gémenos, B.P.100, 13881 Gémenos, France
marc.joye@gemplus.com
[3] Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
{jean-sebastien.coron, david.naccache, pascal.paillier}@gemplus.com

Abstract. This paper introduces two new attacks on PKCS#1 v1.5, an RSA-based encryption standard proposed by RSA Laboratories. As opposed to Bleichenbacher's attack, our attacks are chosen-plaintext only, *i.e.* they do *not* make use of a decryption oracle. The first attack applies to small public exponents and shows that a plaintext ending by sufficiently many zeroes can be recovered efficiently when two or more ciphertexts corresponding to the same plaintext are available. We believe the technique we employ to be of independent interest, as it extends Coppersmith's low-exponent attack to certain length parameters. Our second attack is applicable to *arbitrary* public exponents, provided that most message bits are zeroes. It seems to constitute the first chosen-plaintext attack on an RSA-based encryption standard that yields to practical results for any public exponent.

1 Introduction

PKCS stands for *Public-Key Cryptography Standards*. It is a large corpus of specifications covering RSA encryption [13], Diffie-Hellman key agreement, password-based encryption, syntax (extended-certificates, cryptographic messages, private-key information and certification requests) and selected attributes. Historically, PKCS was developed by RSA Laboratories, Apple, Digital, Lotus, Microsoft, MIT, Northern Telecom, Novell and Sun. The standards have been regularly updated since. Today, PKCS has become a part of several standards and of a wide range of security products including Internet Privacy-Enhanced Mail.

Amongst the PKCS collection, PKCS#1 v1.5 describes a particular encoding method for RSA encryption called `rsaEncryption`. In essence, the enveloped data is first encrypted under a randomly chosen key K using a symmetric block-cipher (*e.g.* a triple DES in CBC mode) then K is RSA-encrypted with the recipient's public key.

In 1998, Bleichenbacher [2] published an adaptive chosen-ciphertext attack on PKCS#1 v1.5 capable of recovering arbitrary plaintexts from a few hundreds of

thousands of ciphertexts. Although active adversary models are generally viewed as theoretical issues,[1] Bleichenbacher's attack makes use of an oracle that only detects conformance with respect to the padding format, a real-life assumption leading to a practical threat. PKCS#1 was subsequently updated in the release 2.0 [15] and patches were issued to users wishing to continue using the old version of the standard.

Independently, there exist several well-known chosen-plaintext attacks on RSA-based encryption schemes [8, 5]. These typically enable an attacker to decrypt ciphertexts at moderate cost without requiring to factor the public modulus. The most powerful cryptanalytic tool applicable to low exponent RSA is probably the one based on a theorem due to Coppersmith [6]. As a matter of fact, one major purpose of imposing a partially random padding form to messages, besides attempting to achieve a proper security level such as indistinguishability, is to render the whole encryption scheme resistant against such attacks.

This paper shows that, despite these efforts, chosen-plaintext attacks are actually sufficient to break PKCS#1 v1.5 even in cases when Coppersmith's attack does not apply. We introduce new cryptanalytic techniques allowing an attacker to retrieve plaintexts belonging to a certain category, namely messages ending by a required minimum number of zeroes. The first attack requires two or more ciphertexts corresponding to the same plaintext. Although specific, our attacks only require a *very* small amount of ciphertexts (say ten of them), are completely independent from the public modulus given its size and, moreover, are fully practical for usual modulus sizes.

The rest of this paper is divided as follows. Section 2 introduces a new low-exponent attack for which we provide a comparison with Coppersmith's attack in Section 3. Section 4 shows how to deal with arbitrary public exponents while staying within the chosen-plaintext attack model. Counter-measures are discussed in Section 5. For completeness, Appendix A reports practical experiments of our technique performed on 1024-bit ciphertexts.

2 Our Low-Exponent Chosen-Plaintext Attack

We briefly recall the PKCS#1 v1.5 encoding procedure [14]. Let $\{n, e\}$ be an RSA public key and d be the corresponding secret key. Denoting by k the byte-length of n, we have $2^{8(k-1)} \leq n < 2^{8k}$. A message m of size $|m|$ bytes with $|m| \leq k-11$ is encrypted as follows. A padding r' consisting of $k-3-|m| \geq 8$ nonzero bytes is generated at random. Then the message m gets transformed into:

$$\text{PKCS}(m, r') = 0002_{16} \| r' \| 00_{16} \| m \,,$$

and encrypted to form the ciphertext:

[1] Chosen-ciphertext attacks require the strong assumption that the adversary has a complete access to a decryption oracle.

$$c = \text{PKCS}(m, r')^e \bmod n \ .$$

Letting $r = (0002_{16} \| r')$, we can write $\text{PKCS}(m, r') = r\,2^\beta + m$ with $\beta = 8|m| + 8$. Now assume that m has its least Z significant bits equal to zero. Hence, we can write $m = \bar{m}\,2^Z$ and subsequently:

$$\text{PKCS}(m, r') = 2^Z (r\,2^{\beta - Z} + \bar{m}) \ .$$

From two encryptions of the same message m, (*i.e.* $c_i = [2^Z (r_i 2^{\beta-Z} + \bar{m})]^e \bmod n$ for $i = 1, 2$), the attacker evaluates:

$$\begin{aligned} \Delta &:= \frac{c_1 - c_2}{2^{eZ}\,2^{\beta - Z}} \bmod n \\ &\equiv \underbrace{(r_1 - r_2)}_{:=\omega} \underbrace{\left[\sum_{j=0}^{e-1} (r_1\,2^{\beta-Z} + \bar{m})^{e-1-j} (r_2\,2^{\beta-Z} + \bar{m})^j\right]}_{:=v} \pmod n \ . \end{aligned} \tag{1}$$

The attack consists in the following: assuming that $r_1 > r_2$ and the number of zeroes Z to be large enough so that $0 < \omega\, v < n$, relation (1) holds over the integers, and $\omega = r_1 - r_2$ must divide Δ. Therefore, by extracting the small factors of Δ one expects to reconstruct a candidate for ω. The correct guess for ω will lead to the message m using the low-exponent attack described in [7].

Letting R the bit-size of random r' (the standard specifies $R \geq 64$), M the bit size of $\bar{m}$, and N the bit size of modulus n, the condition $w \cdot v < n$ is satisfied whenever:

$$eR + (e-1) \times (M + 10) < N \ . \tag{2}$$

With $N = R + M + Z + 24$, equation (2) is equivalent to:

$$(e-1)R + (e-2)M + 10e - 34 < Z$$

2.1 Determining the Factors of Δ Smaller than a Bound B

The first step of our attack consists in computing a set $\mathcal{D}$ of divisors of Δ by extracting the primes $\mathcal{P} = \{p_1, \ldots, p_i\}$ that divide Δ and are smaller than a bound B. If all the prime factors of ω are smaller than B (in this case, ω is said to be B-smooth), then $\omega \in \mathcal{D}$. Since only a partial factorization of Δ is required, only factoring methods which complexity relies on the size of the prime factors are of interest here. We briefly recall four of these: trial division, Pollard's ρ method, $p - 1$ method and Lenstra's elliptic curve method (ECM) and express for each method the asymptotic complexity $C(p)$ of extracting a factor p from a number n.

Trial division method: Trial division by primes smaller than a bound B demands a complexity of $p + \log n$ for extracting p.

Pollard's ρ-method [4]: Let p be a factor of n. Pollard's ρ-method consists in iterating a polynomial with integer coefficients f (*i.e.* computing $f(x) \bmod n$, $f(f(x)) \bmod n$, and so on) until a collision modulo p is found (*i.e.* $x \equiv x' \pmod p$). Then with high probability $\gcd(x - x' \pmod n, n)$ yields p. The complexity of extracting a factor p is $\mathcal{O}(\sqrt{p})$. In practice, prime factors up to approximately 60 bits can be extracted in reasonable time (less than a few hours on a workstation).

$p-1$ method: If $p-1$ is B-smooth then $p-1$ divides the product $\ell(B)$ of all primes smaller than B. Since $a^{p-1} \bmod p = 1$, we have $a^{\ell(B)} \bmod p = 1$ and thus $\gcd(a^{\ell(B)} - 1 \bmod n, n)$ gives p.

Lenstra's elliptic curve method (ECM) [11]: ECM is a generalization of the $p-1$ factoring method. Briefly, a point P of a random elliptic curve $\mathcal{E}$ modulo n is generated. If $\#\mathcal{E}/(p)$ (*i.e.* the order of the curve modulo p) is B-smooth, then $[\ell(B)]P = \mathcal{O}$, the point at infinity. This means that an illegal inversion modulo n has occurred and p is revealed. ECM extracts a factor p of n in $\exp((\sqrt{2} + \mathrm{o}(1))\sqrt{\log p \log\log p})$ expected running time. In practice, prime factors up to 80 bits can be pulled out in reasonable time (less than a few hours on a workstation).

Traditionally, $\psi(x, y)$ denotes the number of integers $z \leq x$ such that z is smooth with respect to the bound y. The theorem that follows gives an estimate for $\psi(x, y)$.

Theorem 1 ([9]). *For any non-negative real u, we have:*

$$\lim_{x \to \infty} \psi(x, x^{1/u})/x = \rho(u)\,,$$

where $\rho(u)$ is the so-called Dickman's function and is defined as:

$$\rho(t) = \begin{cases} 1 & \text{if } 0 \leq t < 1 \\ \rho(n) - \displaystyle\int_n^t \frac{\rho(v-1)}{v}\,dv & \text{if } n \leq t < n+1 \end{cases}.$$

Theorem 1 shows that a *uniformly distributed* random integer z between 1 and x is $x^{1/u}$-smooth with probability $\rho(u)$. However, the integers referred to in the sequel are not uniformly distributed. Consequently, the probability and complexity estimates must be considered to be heuristic.

The probability that ω is B-smooth is approximately $\rho(R/\log_2 B)$. Thus using two ciphertexts, the probability of finding all factors of ω is $\rho(R/\log_2 B)$. When using k ciphertexts, $k \times (k-1)/2$ paired combinations can be obtained. Assuming statistical independence between the factorization of the corresponding w, approximately

$$k = \sqrt{2/\rho(R/\log_2 B)}$$

ciphertexts are required to compute the factorization of at least one ω in complexity:

$$C(B)/\rho(R/\log_2 B) \ .$$

In practice, a factorization algorithm starts with trial division up to some bound B' (we took $B' = 15000$), then Pollard's ρ-method and the $p-1$ method are applied, and eventually the ECM. In Table 1 we give the running times obtained on a Pentium 233-MHz to extract a prime factor of size L bits with the ECM, using the arithmetic library MIRACL [12].

Table 1. Running times for extracting a prime factor of L bits using the ECM

L	32	40	48	56	64	72
time in seconds	6	15	50	90	291	730

This clearly shows that for $R \leq 72$, the factors of ω can be recovered efficiently. For $R > 72$ we estimate in Table 2 the execution time and the number of required ciphertexts, when only factors up to 72 bits are to be extracted.

Table 2. Running time and approximate number of ciphertexts needed to recover the factorization of at least one ω

L	128	160	192	224	256
time in seconds	1719	3440	7654	19010	51127
number of ciphertexts	3	4	5	8	12

2.2 Identifying the Candidates for ω

From the previous section we obtain a set of primes $\mathcal{P} = \{p_1, \ldots, p_i\}$ dividing Δ, such that the primes dividing ω are in $\mathcal{P}$. From $\mathcal{P}$ we derive a set $\mathcal{D} = \{\Delta_j\}$ of divisors of Δ, which contains ω. Denoting by $d(k)$ the number of divisors of an integer k, the following theorem [10] provides an estimate of the number of divisors of a random integer. We say that an arithmetical function $f(k)$ is of the *average order* of $g(k)$ if

$$f(1) + f(2) + \ldots + f(k) \sim g(1) + \ldots + g(k) \ .$$

We state:

Theorem 2. *The average order of $d(k)$ is $\log k$. More precisely, we have:*

$$d(1) + d(2) + \cdots + d(k) = k \log k + (2\gamma - 1)k + \mathrm{O}(\sqrt{k}) \,,$$

where γ is Euler's constant.

Theorem 2 shows that if Δ was uniformly distributed between 1 and n then its number of divisors and consequently the average number of candidates for ω would be roughly $\log n$. Since Δ is not uniformly distributed this only provides an heuristic argument to show that the average number of candidates for ω should be polynomially bounded by $\log n$.

In practice, not all divisors Δ_j need to be tested since only divisors of length close to or smaller than R are likely to be equal to ω. Moreover, from Eq. (1) and letting $\bar{m}_2 = r_2\, 2^{\beta - Z} + \bar{m}$, we have:

$$\begin{aligned}
\Delta &= \omega \sum_{j=0}^{e-1} (\omega\, 2^{\beta-Z} + \bar{m}_2)^{e-1-j}\, \bar{m}_2^j \\
&= \omega \sum_{j=0}^{e-1} \sum_{k=0}^{e-1-j} \binom{e-1-j}{k} (\omega\, 2^{\beta-Z})^{e-1-j-k}\, \bar{m}_2^{j+k} \\
&= \omega \sum_{h=0}^{e-1} \left[\sum_{i=0}^{h} \binom{e-1-i}{h-i} \right] (\omega\, 2^{\beta-Z})^{e-1-h}\, \bar{m}_2^h \,,
\end{aligned}$$

whence, noting that $\sum_{i=0}^{h} \binom{e-1-i}{h-i} \equiv 0 \pmod{e}$ for $1 \leq h \leq e-1$,

$$\Delta \equiv \omega\, (\omega\, 2^{\beta-Z})^{e-1} \pmod{e} \,.$$

In particular, when e is prime, this simplifies to

$$\Delta \equiv \omega^e\, 2^{(\beta-Z)(e-1)} \equiv \omega \pmod{e} \,.$$

This means that only a Δ_j satisfying $\Delta \equiv \Delta_j\, (\Delta_j\, 2^{\beta-Z})^{e-1} \pmod{e}$ (or $\Delta \equiv \Delta_j \pmod{e}$ if e is prime) is a valid candidate for ω.

2.3 Recovering m Using the Low-Exponent RSA with Related Messages Attack

The low-exponent attack on RSA with related messages described in [7] consists in the following: assume that two messages m_1, m_2 verify a known polynomial relation $\mathcal{P}$ of the form

$$m_2 = \mathcal{P}(m_1) \quad \text{with } \mathcal{P} \in \mathbb{Z}_n[z] \text{ and } \deg(\mathcal{P}) = \delta \,,$$

and suppose further that the two corresponding ciphertexts c_1 and c_2 are known. Then $z = m_1$ is a common root of polynomials $\mathcal{Q}_1, \mathcal{Q}_2 \in \mathbb{Z}_n[z]$ given by

$$\mathcal{Q}_1(z) = z^e - c_1 \quad \text{and} \quad \mathcal{Q}_2(z) = (\mathcal{P}(z))^e - c_2 \,,$$

so that with high probability one recovers m_1 by

$$\gcd(\mathcal{Q}_1, \mathcal{Q}_2) = z - m_1 \pmod{n} \,.$$

From the previous section we obtain a set of divisors Δ_j of Δ, among which one is equal to ω. Letting $m_1 = \text{PKCS}(m, r_1)$ and $m_2 = \text{PKCS}(m, r_2)$ we have:

$$c_1 = m_1^e \pmod{n}, \quad c_2 = m_2^e \pmod{n}, \quad \text{and } m_2 = m_1 - 2^\beta \omega \,.$$

For a divisor Δ_j of Δ, the attacker computes:

$$\mathcal{R}_j(z) = \gcd(z^e - c_1, (z - 2^\beta \Delta_j)^e - c_2) \,.$$

If $\Delta_j = \omega$ then, with high probability, $\mathcal{R}_j(z) = z - m_1 \pmod{n}$, which yields the value of message m, as announced.

3 Comparison with Coppersmith's Attacks on Low-exponent RSA

Coppersmith's method is based on the following theorem [6]:

Theorem 3 (Coppersmith). *Let $\mathcal{P} \in \mathbb{Z}_n[x]$ be a univariate polynomial of degree δ modulo an integer n of unknown factorization. Let X be the bound on the desired solution. If $X < \frac{1}{2} n^{1/\delta - \varepsilon}$, one can find all integers x_0 with $\mathcal{P}(x_0) = 0 \pmod{n}$ and $|x_0| \leq X$ in time polynomial in $(\log n, \delta, 1/\varepsilon)$.*

Corollary 1 (Coppersmith). *Under the same hypothesis and provided that $X < n^{1/\delta}$, one can find all integers x_0 such that $\mathcal{P}(x_0) = 0 \pmod{n}$ and $|x_0| \leq X$ in time polynomial in $(\log n, \delta)$*

Theorem 3 applies in the following situations:

Stereotyped messages: Assume that the plaintext m consists of a known part $B = 2^k b$ and an unknown part x. The ciphertext is $c = m^e = (B + x)^e \pmod{n}$. Using Theorem 3 with the polynomial $\mathcal{P}(x) = (B + x)^e - c$, one can recover x from c if $|x| < n^{1/e}$.

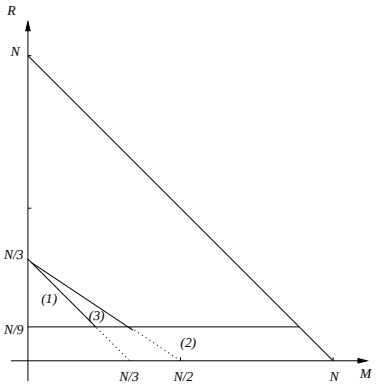


Fig. 1. Domains of validity for $e = 3$ of Coppersmith's stereotyped attack (1), Coppersmith's random padding attack (2) and our attack (3).

Random padding: Assume that two messages m and m' satisfy an affine relation $m' = m + r$ with a small but unknown r. From the RSA-encryptions of the two messages:

$$c = m^e \bmod n \quad \text{and} \quad c' = (m + r)^e \bmod n\,,$$

we eliminate m from the two above equations by taking their resultant, which gives a univariate polynomial in r modulo n of degree e^2. Thus, if $|r| < n^{1/e^2}$, r can be recovered, wherefrom we derive m as in Section 2.3.

In our case of interest, for a message ending with Z zeroes, the stereotyped messages attack works for $e(M + R) < N$ and the random padding attack works for $e^2 R < N$. Neglecting constant terms, our method of Section 2 is effective for

$$eR + (e-1)M < N\ .$$

Consequently, as illustrated in Figure 1, for $e = 3$, our method improves Coppersmith's method whenever

$$\begin{cases} \dfrac{N}{e^2} < R < \dfrac{N}{e} & \text{and} \\ \dfrac{N}{e} - R < M < \dfrac{N}{e-1} - \dfrac{e}{e-1}R \end{cases}.$$

4 A Chosen Plaintext Attack for Arbitrary Exponents

4.1 Description

In this section we describe a chosen plaintext attack against PKCS#1 v1.5 encryption for an arbitrary exponent e. The attack makes use of a known flaw in

ElGamal encryption [3] and works for very short messages only. As in Section 2 we only consider messages ending by Z zeroes:

$$m = \bar{m} \| 0 \ldots 0_2 \ .$$

For a random r' consisting of nonzero bytes, the message m is transformed using PKCS#1 v1.5 into:

$$\text{PKCS}(m, r') = 0002_{16} \| r' \| 00_{16} \| \bar{m} \| 0 \ldots 0_2$$

and encrypted into $c = \text{PKCS}(m, r')^e \bmod n$. Letting $x = 0002_{16} \| r' \| 00_{16} \| \bar{m}$, we can write

$$\text{PKCS}(m, r') = x\, 2^Z \ .$$

We define $y = c/2^{eZ} = x^e \pmod n$, M the bit-size of $\bar{m}$, and X the bit-size of x. Hence, we have $X = M + R + 10$. Assuming that $x = x_1\, x_2$ where x_1 and x_2 are integers smaller than a bound B, we construct the table:

$$\frac{y}{i^e} \bmod n \quad \text{for } i = 1, \ldots, B$$

and for each $j = 0, \ldots, B$ we check whether $j^e \bmod n$ belongs to the table, in which case we have $y/i^e = j^e \bmod n$. Hence, from $\{i, j\}$ we recover $x = i \cdot j$, which leads to the message m.

4.2 Analysis

The attack requires $\mathcal{O}\big(B(\log n)((\log n)^3 + \log B)\big)$ operations. Let $\phi(x, y)$ denote the number of integers $v < x$ such that v can be written as $v = v_1\, v_2$ with $v_1 < y$ and $v_2 < y$. The following theorem gives a lower bound for $\phi(x, y)$.

Theorem 4. *For $x \to \infty$ and $1/2 < \alpha < 1$,*

$$\liminf \phi(x, x^\alpha)/x \geq \log \frac{\alpha}{1-\alpha} \ . \tag{3}$$

Proof: For $y > \lceil \sqrt{x} \rceil$, we note:

$$\mathcal{T}(x, y) = \{v < x, \text{such that } v \text{ is } y\text{-smooth and not } \lceil x/y \rceil\text{-smooth}\} \ .$$

Any integer $v \in \mathcal{T}(x, y)$ has a prime factor p standing between $\lceil x/y \rceil$ and y, and so $v = p\, r$ with $p < y$ and $r < y$. Consequently,

$$\phi(x, y) \geq \#\mathcal{T}(x, y) \ . \tag{4}$$

From Theorem 1 and $\rho(t) = 1 - \log t$ for $1 \leq t \leq 2$, we have:

$$\lim_{x \to \infty} \#\mathcal{T}(x, x^\alpha)/x = \log \frac{\alpha}{1-\alpha},$$

which, using Eq. (4) gives (3). □

Since x is not uniformly distributed between zero and 2^X, Theorem 4 only provides a heuristic argument to show that when taking $B = 2^{\alpha X}$ with $\alpha > 1/2$, then with probability greater than

$$\log \frac{\alpha}{1-\alpha},$$

the attack recovers x in complexity $2^{\alpha X + o(1)}$.

Thus, an eight-bit message encrypted with PKCS#1 v1.5 with a 64-bit random padding string can be recovered with probability $\simeq 0.16$ in time and space complexity approximately 2^{44} (with $\alpha = 0.54$).

5 Experiments and Counter-measures

A number of counter-measures against Bleichenbacher's attack are listed on RSA Laboratories' web site (`http://www.rsa.com/rsalabs/`). A first recommendation is a rigorous format check of all decrypted messages. This has no effect on our attack since we never ask the legitimate receiver to decrypt anything. A second quick fix consists in asking the sender to demonstrate knowledge of m to the recipient which is done by disclosing some additional piece of information. This also has no effect on our attack. The same is true for the third correction, where a hash value is incorporated in m, if the hash value occupies the most significant part of the plaintext *i.e.*

$$\text{PKCS}(m, r') = 0002_{16} \| r' \| 00_{16} \| \text{SHA}(m) \| m \ .$$

A good way to thwart our attack is to limit Z. This can be very simply achieved by forcing a constant pattern τ in $\text{PKCS}(m, r')$:

$$\text{PKCS}(m, r') = 0002_{16} \| r' \| 00_{16} \| m \| \tau \ .$$

This presents the advantage of preserving compatibility with PKCS#1 v1.5 and being very simple to implement. Unfortunately, the resulting format is insufficiently protected against [2]. Instead, we suggest to use:

$$\text{PKCS}(m, r') = 0002_{16} \| r' \| 00_{16} \| m \| \text{SHA}(m, r'),$$

which appears to be an acceptable short-term choice (r' was added in the hash function to better resist [2] at virtually no additional cost). For long-term permanent solutions, we recommend OAEP (PKCS#1 v2.0) [1].

6 Extensions and Conclusions

We proposed two new chosen-plaintext attacks on the PKCS#1 v1.5 encryption standard. The first attack applies to small public exponents and shows how messages ending by sufficiently many zeroes can be recovered from the ciphertexts corresponding to the same plaintext. It is worth seeing our technique as a cryptanalytic tool of independent interest, which provides an extension of Coppersmith's low-exponent attack. Our second attack, although remaining of exponential complexity in a strict sense, shows how to extend the weakness to any public exponent in a practical way.

The attacks can, of course, be generalized in several ways. For instance, one can show that the padding format:

$$\mu(m_1, m_2, r') = 0002_{16} \| m_1 \| r' \| 00_{16} \| m_2$$

(where the plaintext $m = m_1 \| m_2$ is spread between two different locations), is equally vulnerable to the new attack: re-defining $r'' = m_1 \| r'$, we can run the attack (as is) on $\mathrm{pkcs}(m, r'')$ and notice that the size of ω will still be R' given that the most significant part of r'' is always constant.

We believe that such examples illustrate the risk induced by the choice of *ad hoc* low-cost treatments as message paddings, and highlights the need for carefully scrutinized encryption designs, strongly motivating (once again) the search for provably secure encryption schemes.

7 Acknowledgements

We wish to thank the referees for their valuable remarks and improvements to this work.

References

1. M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption*, Advances in Cryptology — EUROCRYPT '94, vol. 950 of Lecture Notes in Computer Science, pp. 92–111, Springer-Verlag, 1994.
2. D. Bleichenbacher, *Chosen ciphertext attacks against protocols based on the* RSA *encryption standard* PKCS#1, Advances in Cryptology — CRYPTO '98, vol. 1462 of Lecture Notes in Computer Science, pp. 1–12, Springer-Verlag, 1998.
3. D. Boneh, Personal communication.
4. R. Brent, *An improved Monte Carlo factorization algorithm*, Nordisk Tidskrift för Informationsbehandling (BIT) vol. 20, pp. 176–184, 1980.
5. D. Coppersmith, *Finding a small root of a univariate modular equation*, Advances in Cryptology — EUROCRYPT '96, vol. 1070 of Lecture Notes in Computer Science, pp. 155–165, Springer-Verlag, 1996.
6. D. Coppersmith, *Small solutions to polynomial equations, and low exponent* RSA *vulnerabilities*, J. of Cryptology, 10(4), pp. 233-260, 1997.

7. D. Coppersmith, M. Franklin, J. Patarin and M. Reiter, *Low exponent* RSA *with related messages*, Advances in Cryptology — EUROCRYPT '96, vol. 1070 of Lecture Notes in Computer Science, pp. 1–9, Springer-Verlag, 1996.
8. Y. Desmedt and A. Odlyzko. *A chosen text attack on the* RSA *cryptosystem and some discrete logarithm schemes*, Advances in Cryptology — CRYPTO '85, vol. 218 of Lecture Notes in Computer Science, pp. 516–522, Springer-Verlag, 1986.
9. K. Dickman, *On the frequency of numbers containing prime factors of a certain relative magnitude*, Arkiv för matematik, astronomi och fysik, vol. 22A, no. 10, pp. 1–14, 1930.
10. G.H. Hardy and E.M. Wright, *An Introduction to the theory of numbers*, Fifth edition, Oxford University Press, 1979.
11. H. Lenstra, *Factoring integers with elliptic curves*, Annals of mathematics 126, 1987.
12. Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL), available at `ftp://ftp.compapp.dcu.ie/pub/crypto/miracl.zip`.
13. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, vol. 21-2, pp. 120-126, 1978.
14. RSA Data Security, PKCS #1: RSA *Encryption Standard*, Nov. 1993. Version 1.5.
15. RSA Laboratories, PKCS #1: RSA *Cryptography Specifications*, Sep. 1998, version 2.0.

A A Full-scale 1024-bit Attack

To confirm the validity of our attack, we experimented it on RSA Laboratories' official 1024-bit challenge RSA-309 for the public exponent $e = 3$. As a proof of proper generation r'_1 and r'_2 were chosen to be RSA-100 mod 2^{128} and RSA-110 mod 2^{128}. The parameters are $N = 1024$, $M = 280$, $R = 128$, $Z = 592$ and $\beta = 880$. Note that since $R > N/9$ and $R + M > N/3$, Coppersmith's attack on low-exponent RSA does not apply here.

$$
\begin{aligned}
n &= \text{RSA-309} \\
&= \texttt{bdd14965 645e9e42 e7f658c6 fc3e4c73 c69dc246 451c714e b182305b 0fd6ed47} \\
&\quad \texttt{d84bc9a6 10172fb5 6dae2f89 fa40e7c9 521ec3f9 7ea12ff7 c3248181 ceba33b5} \\
&\quad \texttt{5212378b 579ae662 7bcc0821 30955234 e5b26a3e 425bc125 4326173d 5f4e25a6} \\
&\quad \texttt{d2e172fe 62d81ced 2c9f362b 982f3065 0881ce46 b7d52f14 885eecf9 03076ca5} \\
r'_1 &= \text{RSA-100} \bmod 2^{128} \\
&= \texttt{f66489d1 55dc0b77 1c7a50ef 7c5e58fb} \\
r'_2 &= \text{RSA-110} \bmod 2^{128} \\
&= \texttt{e2a5a57d e621eec5 b14ff581 a6368e9b} \\
m &= \bar{m}\, 2^Z \\
&\quad \overset{\texttt{I}}{\texttt{0049276d}}\ \overset{'}{}\ \overset{\texttt{m}}{}\ \texttt{20612063 69706865 72746578 742c2070 6c656173 65206272 65616b20} \\
&\quad \overset{\texttt{m}}{\texttt{6d652021}}\ \overset{\texttt{e}}{}\ \overset{!}{} \\
\mu_1 &= \text{PKCS}(m, r'_1) \\
&= \texttt{0002f664 89d155dc 0b771c7a 50ef7c5e 58fb0049 276d2061 20636970 68657274} \\
&\quad \texttt{6578742c 20706c65 61736520 62726561 6b206d65 20210000 00000000 00000000} \\
&\quad \texttt{00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000} \\
&\quad \texttt{00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000} \\
\mu_2 &= \text{PKCS}(m, r'_2) \\
&\quad \texttt{0002e2a5 a57de621 eec5b14f f581a636 8e9b0049 276d2061 20636970 68657274} \\
&\quad \texttt{6578742c 20706c65 61736520 62726561 6b206d65 20210000 00000000 00000000} \\
&\quad \texttt{6578742c 20706c65 61736520 62726561 6b206d65 20210000 00000000 00000000} \\
&\quad \texttt{00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000} \\
&\quad \texttt{00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000} \\
c_1 &= \mu_1^3 \bmod n \\
&= \texttt{2c488b6f cf2e3d4c 01b82776 64790af0 d78f82fd 4605fda2 76b9356d 80e82cfb} \\
&\quad \texttt{8737340f 5a7091b0 38c4bb41 ae6462d9 f751766c c343c87b 54397ca2 647d6a81} \\
&\quad \texttt{3609d876 f29554e0 9efcbf2d b49d8300 5fce9ea8 80fd9cf2 476fbab0 257f1462} \\
&\quad \texttt{d295a4cb 5468bb86 b3151a49 14e51ed1 7cbc083c 9ae0b4da 9c2a7de0 079df4a0} \\
c_2 &= \mu_2^3 \bmod n \\
&= \texttt{829da9a7 af2c61ed 7bb16f94 7cb90aa7 df8b99df c06017d7 3afc80fd 64494abb} \\
&\quad \texttt{3c1cb8db 1167eccd d1b6d09e 8ca5a98c c5e19620 b6313eef 495169d7 9ed9a2b1} \\
&\quad \texttt{cb393e7d 45bea586 49e20986 9a2399f7 f70dd819 90183e1a 3c6a971a 33497e57} \\
&\quad \texttt{f0ad9fb9 0c7d331e 7108d661 4c487a85 36cf7750 060811d8 70b8a040 e0c39999}
\end{aligned}
$$

Using the ECM it took a few hours on a single workstation to find that:

$$\Delta = p_1^5 \times \prod_{i=2}^{10} p_i$$

where all the p_i are primes. Amongst the $3072 = 6 \times 2^9$ possible divisors only 663 corresponded to 128-bit candidates $\{\Delta_1, \Delta_2, \ldots, \Delta_{663}\}$ where the Δ_i are in decreasing order. Then we computed:

$$\mathcal{R}_j(z) = \gcd(z^e - c_1, (z - 2^\beta \Delta_j)^e - c_2) \quad \text{for } 1 \leq j \leq 663 \ .$$

For $j \neq 25$, $\mathcal{R}_j(z) = 1$ and for $j = 25$ we obtained:

$$\mathcal{R}_{25}(z) = z - m_1 \ .$$

One can check that:

$$\Delta_{25} = w = p_1^5 \, p_2 \, p_3 \, p_4 \, p_5 \, p_8 \, ,$$

and

$$m_1 = \mu_1 = \text{PKCS}(m, r_1') \ .$$

$$\begin{aligned}
\Delta &= \texttt{00000001 fa75bf4e 390bdf4b 7a0524e0 b9ebed20 5758be2e f1685067 1de199af} \\
&\quad \texttt{0f8714f7 077a6c47 6870ea6d 2de9e7fb 3c40b8d2 017c0197 f9533ed1 f4fe3eab} \\
&\quad \texttt{836b6242 aa03181a 56a78001 7c164f7a c54ecfa7 73583ad8 ffeb3a78 eb8bcbe2} \\
&\quad \texttt{8869da15 60be7922 699dc29a 52038f7b 83e73d4e 7082700d 85d3a720} \\
p_1 &= \texttt{00000002}, \ p_2 = \texttt{00000007}, \ p_3 = \texttt{00000035}, \ p_4 = \texttt{000000c5}, \ p_5 = \texttt{4330e379} \\
p_6 &= \texttt{548063d7}, p_7 = \texttt{001ebf96 ff071021}, \ p_8 = \texttt{0000021b ac4d83ae 7dedba55} \\
p_9 &= \texttt{0000128a ec52c6ec 096996bf} \\
p_{10} &= \texttt{00000022 e3b1a6b0 13829b67 f604074a 5a1135b3 45be0835 ea407ed7 8138a27a} \\
&\quad \texttt{112e78c8 131f3bc3 b6d17dc0 e8a905f1 ca4b6aff 680bc58c 4962309d c7aaccad} \\
&\quad \texttt{2116235c b0d6803e e0a58ca7 55cbea23 e936f189 a76dfbeb} \\
\Delta_{25} &= \texttt{13bee453 6fba1cb1 6b2a5b6d d627ca60} \\
\mathcal{R}_{25}(z) &= z - m_1 \\
m_1/2^Z \bmod 2^M &\quad \texttt{0049276d 20612063 69706865 72746578 742c2070 6c656173 65206272 65616b20} \\
&\quad \texttt{6d652021}
\end{aligned}$$

A NICE Cryptanalysis

Éliane Jaulmes[1] and Antoine Joux[2]

[1] SCSSI, 18 rue du Docteur Zamenhof
F-92131 Issy-les-Moulineaux cedex, France
eliane.jaulmes@wanadoo.fr
[2] SCSSI, 18 rue du Docteur Zamenhof
F-92131 Issy-les-Moulineaux cedex, France
Antoine.Joux@ens.fr

Abstract. We present a chosen-ciphertext attack against both NICE cryptosystems. These two cryptosystems are based on computations in the class group of non-maximal imaginary orders. More precisely, the systems make use of the canonical surjection between the class group of the quadratic order of discriminant $\sqrt{-pq^2}$ and the class group of the quadratic order of discriminant $\sqrt{-p}$. In this paper, we examine the properties of this canonical surjection and use them to build a chosen-ciphertext attack that recovers the secret key (p and q) from two ciphertexts/cleartexts pairs.

1 Overview

In [5], Hartmann, Paulus and Takagi have presented a new public-key cryptosystem based on ideal arithmetic in quadratic orders. This system was called NICE, which stands for New Ideal Coset Encryption.

In [7], Hünhlein, Jacobson, Paulus and Takagi have presented a cryptosystem analogous to ElGamal encryption [4] that uses the same properties of arithmetic in imaginary quadratic orders than NICE. They called it HJPT.

The security of the NICE and HJPT cryptosystems is closely related to factoring the discriminant of the quadratic order, which is a composite number of the special form pq^2. While there exists an algorithm that allows the factorization of numbers of the form pq^r, for large r (see [2]), no dedicated algorithm is currently known to factor numbers with a square factor. Furthermore, for appropriate sizes of the parameters, the currently known general factoring algorithms are not applicable to a direct attack. In [8], the authors also give several arguments to prove the security of their cryptosystem. Among these considerations, they argue that the chosen-ciphertext attack is not applicable to their cryptosystem.

Indeed, it seems that from a single chosen ciphertext, one cannot recover the secret key. However, we show that with two well chosen ciphertexts, it is possible to factor pq^2, thus breaking the system.

This paper is organized as follows : we first give a brief reminder of the properties of the class group of a quadratic order and recall the main ideas of

the two cryptosystems. Then we present our chosen-ciphertext attack and finally we give an example of this attack.

2 Theoretical Background

The NICE and HJPT cryptosystems rely on the canonical surjection between the class group of a non-maximal order and the class group of the maximal order in an imaginary quadratic field. We will first recall the properties of the class groups and the surjection before presenting the algorithms.

2.1 Class Group of a Quadratic Order

An introduction to quadratic orders and their class groups can be found in [3]. In this section, we briefly recall the definition and main properties of the class group of a quadratic order.

Definitions and Properties.

Quadratic Field. Let $K = \mathbb{Q}(\sqrt{d})$ be a quadratic field with $d \neq 1$ squarefree. Let Δ_1 be the discriminant of K. If $d \equiv 1 \bmod 4$, we can take $1, (1+\sqrt{d})/2$ as an integral basis for K and $\Delta_1 = d$, while if $d \equiv 2$ or $3 \bmod 4$, we can take $1, \sqrt{d}$ and we have $\Delta_1 = 4d$.

Fundamental Discriminant. An integer Δ_1 is called a fundamental discriminant if Δ_1 is the discriminant of a quadratic field K. In other words, $\Delta_1 \neq 1$ and either $\Delta_1 \equiv 1 \bmod 4$ and is squarefree, or $\Delta_1 \equiv 0 \bmod 4$, $\Delta_1/4$ is squarefree and $\Delta_1/4 \equiv 2$ or $3 \bmod 4$. In the NICE cryptosystem, we consider only $\Delta_1 < 0$ and such that $\Delta_1 \equiv 1 \bmod 4$.

Order of a Quadratic Field. An order R in K is a subring of K which as a $\mathbb{Z}$-module is finitely generated and of maximal rank $n = deg(K)$. Every element of an order is an algebraic integer. If K is a quadratic field of discriminant Δ_1, then every order R of K has discriminant $q^2\Delta_1$, where q is a positive integer called the conductor of the order. Conversely, if Δ_q is any non-square integer such that $\Delta_q \equiv 0$ or $1 \bmod 4$, then Δ_q is uniquely of the form $\Delta_q = q^2\Delta_1$ where Δ_1 is a fundamental discriminant, and there exists an unique order R of discriminant Δ_q.

Maximal Order. Let $\mathcal{O}_{\Delta_q}$ be the order of discriminant Δ_q. It can be written as $\mathcal{O}_{\Delta_q} = \mathbb{Z} + w\mathbb{Z}$ where $w = \frac{\Delta_q + i\sqrt{|\Delta_q|}}{2}$. $\mathcal{O}_{\Delta_q}$ is related to $\mathcal{O}_{\Delta_1}$ by the relation $\mathcal{O}_{\Delta_q} = \mathbb{Z} + q\mathcal{O}_{\Delta_1}$ and we have $\mathcal{O}_{\Delta_q} \subset \mathcal{O}_{\Delta_1}$. We call $\mathcal{O}_{\Delta_1}$ the maximal order. It is the ring of integers of the quadratic field $\mathbb{Q}(\sqrt{\Delta_1})$.

Ideals of a Quadratic Order. An ideal $\mathfrak{a}$ of $\mathcal{O}_{\Delta_q}$ can be written as

$$\mathfrak{a} = m\left(a\mathbb{Z} + \frac{b + i\sqrt{|\Delta_q|}}{2}\mathbb{Z}\right)$$

where $m \in \mathbb{Z}$, $m > 0$, $a \in \mathbb{Z}$, $a > 0$, and $b \in \mathbb{Z}$ such that $b^2 \equiv \Delta_q \bmod 4a$. The norm of the ideal is defined as $N(\mathfrak{a}) = ma$. When $m = 1$, we say that $\mathfrak{a}$ is primitive and we represent it by the pair (a, b).

Two ideals $\mathfrak{a}$, $\mathfrak{b} \in \mathcal{O}_{\Delta_q}$ are called equivalent if there exists $\alpha, \beta \in \mathcal{O}_{\Delta_q}^*$ such that $\alpha\mathfrak{a} = \beta\mathfrak{b}$. We denote this relation by $\mathfrak{a} \sim \mathfrak{b}$. For any element $\gamma \in \mathcal{O}_{\Delta_q}$ the ideal $\gamma\mathcal{O}_{\Delta_q}$ is called a principal ideal. If $\mathfrak{a}$ and $\mathfrak{b}$ are two principal ideals, they are equivalent.

For a primitive ideal, we say that $\mathfrak{a} = (a, b)$ is reduced if and if only $|b| \leq a \leq c = (b^2 - \Delta_q)/4a$ and moreover $b \geq 0$ when $a = c$ or $a = |b|$. There exists a unique reduced ideal in the equivalence class of an ideal $\mathfrak{a}$, denoted by $Red_{\Delta_q}(\mathfrak{a})$. An algorithm to compute $Red_{\Delta_q}(\mathfrak{a})$ from $\mathfrak{a}$ is described in [3, p238].

The reduction algorithm works as follows in the quadratic order of discriminant Δ : We start with an ideal (a, b) with $-a < b \leq a$ and proceed by successive steps. In each step, we replace (a, b) by (a', b') where $a' = \frac{b^2 - \Delta}{4a}$ and b' satisfies $-b = b' + 2a'k$ with $-a' < b' \leq a'$. When it reaches a reduced ideal, the algorithm stops.

For any reduced ideal $\mathfrak{a} = (a, b)$, $a < \sqrt{|\Delta_q|/3}$. Conversely, for a primitive ideal, if $a < \sqrt{|\Delta_q|/4}$, then $\mathfrak{a}$ is reduced.

Class Group. The ideals of $\mathcal{O}_{\Delta_q}$, respectively $\mathcal{O}_{\Delta_1}$, whose norm is prime to f form an Abelian group. They are called ideals prime to f. We denote this group by $I_{\Delta_q}(f)$, respectively $I_{\Delta_1}(f)$. If q is a prime and $\sqrt{|\Delta_1|/3} < q$, then all the reduced ideals in $\mathcal{O}_{\Delta_1}$ have a norm prime to q. From now on, we will suppose that this is the case.

In $\mathcal{O}_{\Delta_q}$, the principal ideals prime to q form a subgroup of $I_{\Delta_q}(q)$. We denote it by $P_{\Delta_q}(q)$. The quotient group $I_{\Delta_q}(q)/P_{\Delta_q}(q)$ is called the class group of $\mathcal{O}_{\Delta_q}$ and denoted by $Cl(\Delta_q)$.

We can consider the following map :

$$\begin{aligned} \varphi_q : Cl(\Delta_q) &\to Cl(\Delta_1) \\ \mathfrak{a} &\mapsto Red_{\Delta_1}(\mathfrak{a}\mathcal{O}_{\Delta_1}) \end{aligned}$$

φ_q is a surjective group morphism.

We can also defined a restricted inverse map, denoted φ_q^{-1}.

$$\varphi_q^{-1}(A, B) = (A, Bq \bmod 2A)$$

We have indeed $\varphi_q(\varphi_q^{-1}(\mathfrak{a})) = Red_{\Delta_1}(\mathfrak{a})$. Conversely, for an ideal $\mathfrak{a} = (a, b) \in \mathcal{O}_{\Delta_q}$ such that $a < \sqrt{|\Delta_1|/4}$, we have $\varphi_q^{-1}(\varphi_q(\mathfrak{a})) = \mathfrak{a}$. However, if $a \geq \sqrt{|\Delta_1|/4}$, we may have $\varphi_q^{-1}(\varphi_q(\mathfrak{a})) \neq \mathfrak{a}$. Our attack relies on this observation.

How to Compute φ_q. Let Φ_q be the map between the primitive ideals of $\mathcal{O}_{\Delta_q}$ and the primitive ideals of $\mathcal{O}_{\Delta_1}$ defined by $\Phi_q(\mathfrak{a}) = \mathfrak{a}\mathcal{O}_{\Delta_1}$. We clearly have $\varphi_q = Red_{\Delta_1}(\Phi_q)$. To compute $\Phi_q(\mathfrak{a})$ from $\mathfrak{a} = (a, b)$, proceed as follows : $\Phi_q(a, b) = (A, B)$ where $A = a$ and $b\mu + ab_{\mathcal{O}}\nu = 2ka + B$ with $-a < B \leq a$, $b_{\mathcal{O}} = \Delta_q \bmod 2$, $1 = \mu q + \nu a$ for $\mu,\nu \in \mathbb{Z}$. To compute $\varphi_q(\mathfrak{a})$, we must then apply to (A, B) the reduction algorithm described in section 2.1.

2.2 Description of the Cryptosystems

Description of NICE.

The Key Generation. The key generation consists in generating two random primes p,$q > 4$ with $p \equiv 3 \bmod 4$ and $\sqrt{p/3} < q$. We then let

$$\Delta_1 = -p$$
$$\Delta_q = -pq^2,$$

and choose an ideal $\mathfrak{p}$ in $Cl(\Delta_q)$, where $\varphi_q(\mathfrak{p}) = 1_{Cl(\Delta_1)}$. To generate such a $\mathfrak{p}$, proceed as follows : choose a number $\alpha \in \mathcal{O}_{\Delta_1}$ with norm less than $\sqrt{|\Delta_q|/4}$, compute the standard representation of the ideal $\alpha\mathcal{O}_{\Delta_1}$ and compute $\varphi_q^{-1}(\alpha\mathcal{O}_{\Delta_1})$.

Let k and l be the bit lengths of $\lfloor\sqrt{|\Delta_1|/4}\rfloor$ and $q - \left(\frac{\Delta_1}{q}\right)$ respectively, where $\left(\frac{\Delta_1}{q}\right)$ is the Kronecker symbol. The public key is $(\mathfrak{p}, \Delta_q, k, l)$ and the secret key is (Δ_1, q). None of the maps Φ_q, φ_q, φ_q^{-1} are public.

Encryption and Decryption Proceedings. A message is represented by an ideal $\mathfrak{m}$, where $\mathfrak{m}$ is reduced in $Cl(\Delta_q)$ and $\log_2 N(\mathfrak{m}) < k$, which means that $\Phi_q(\mathfrak{m})$ is also reduced in $Cl(\Delta_1)$. The embedding of a message into an ideal that represents it may be done as follows : let x be the message and t a random number of length $k - 2 - \lfloor\log_2 x\rfloor + 1$. We determine the smallest prime a larger than the concatenation of x and t as bit strings with $\left(\frac{\Delta_q}{a}\right) = 1$. Then we need to compute b such that $\Delta_q \equiv b^2 \bmod 4a$, $-a < b \leq a$. Our message is finally encoded as $\mathfrak{m} = (a, b)$.

We encrypt the message by computing $\mathfrak{c} = Red_{\Delta_q}(\mathfrak{m}\mathfrak{p}^r)$, where r is a random $l - 1$ bit integer.

To decrypt, we compute $\mathfrak{k} = \varphi_q(\mathfrak{c})$.
Since

$$\varphi_q(\mathfrak{m}\mathfrak{p}^r) = \varphi_q(\mathfrak{m})\varphi_q(\mathfrak{p}^r) = \varphi_q(\mathfrak{m}),$$

the plaintext is then $\mathfrak{m} = \varphi_q^{-1}(\mathfrak{k})$.

Note that this is a probabilistic encryption and the multiplication by $\mathfrak{p}^r$ allows to choose a random pre-image of $\varphi_q(\mathfrak{m})$ by φ_q.

Description of HJPT. In this cryptosystem, the encryption is done completely analogous to ElGamal encryption [4] in the non-maximal order $\mathcal{O}_{\Delta_q}$. All ideals are chosen prime to q. The public parameters are the discriminant Δ_q, an ideal $\mathfrak{g} \in \mathcal{O}_{\Delta_q}$, called the base ideal, and an ideal $\mathfrak{a} \in \mathcal{O}_{\Delta_q}$ such that $\mathfrak{a} = Red_{\Delta_q}(\mathfrak{g}^a)$, where a is a random integer $a \in [2, \lfloor\sqrt{\Delta_1}\rfloor]$. The secret key is a and q.

We embed the message in an ideal $\mathfrak{m} \in \mathcal{O}_{\Delta_q}$ as in NICE, select an integer k and compute $(\mathfrak{n}_1, \mathfrak{n}_2)$ where $\mathfrak{n}_1, \mathfrak{n}_2$ are reduced ideals in $\mathcal{O}_{\Delta_q}$ and

$$\mathfrak{n}_1 = Red_{\Delta_q}(\mathfrak{g}^k)$$

$$\mathfrak{n}_2 = Red_{\Delta_q}(\mathfrak{m}\mathfrak{a}^k)$$

We require $N(\mathfrak{m}) < \sqrt{|\Delta_1|/4}$ in order to uniquely decrypt the message $\mathfrak{m}$.

The decryption works in the maximal order $\mathcal{O}_{\Delta_1}$. We compute :

$$\mathfrak{N}_1 = \varphi_q(\mathfrak{n}_1)$$

$$\mathfrak{N}_2 = \varphi_q(\mathfrak{n}_2)$$

$$\mathfrak{M} = \mathfrak{N}_2(\mathfrak{N}_1^a)^{-1}$$

$$\mathfrak{m} = \varphi_q^{-1}(\mathfrak{M})$$

$\mathfrak{m}$ is the decoded message.

Security Considerations. The security of the cryptosystems depends on the difficulty of factoring the discriminant Δ_q. If it can be factored, the cryptosystems are clearly broken.

To prevent a direct factorization of Δ_q using general methods such as the number field sieve or the elliptic curve method, the authors suggest that we choose p and q larger than 256 bits. Although Δ_q is of the special form pq^2, there exists no dedicated algorithm better than the general ones. They conclude that their system is secure against attacks by factorization.

The authors also prove that nobody can compute $\Phi_q(a, b)$ without the knowledge of the conductor q. That means that it is not possible to recover the message from the coded ideal without the knowledge of the factors of Δ_q.

Concerning NICE, Paulus and Takagi then argue that the knowledge of $\mathfrak{p}$ does not substantially help to factor Δ_q. A possible attack would be to find an ideal $\mathfrak{f}$ power of $\mathfrak{p}$ in $\mathcal{O}_{\Delta_q}$ such that $\mathfrak{f}^2 \sim 1$ and $\mathfrak{f} \nsim 1$, however the only apparent way to do that is to compute the order of $\mathfrak{p}$ in the group $Cl(\Delta_q)$ which is much slower to do than factoring Δ_q with the available algorithms.

In [8], Paulus and Takagi also claim that the chosen-ciphertext attack is not applicable to their cryptosystem and give a few observations .

3 The Chosen-Ciphertext Attack

In this section, we study more precisely the question of the chosen-ciphertext attack. As claimed in section 2.2, the knowledge of one coded message and the corresponding decrypted message is indeed not sufficient for factoring Δ_q. However, we show that with two chosen ciphertexts, factoring Δ_q becomes easy.

Both cryptosystems use the following property of the canonical surjection to recover the message after encryption :

$$\varphi^{-1}(\varphi(\mathfrak{m})) = \mathfrak{m} \text{ if } N(\mathfrak{m}) < \sqrt{|\Delta_1|/4}.$$

Conversely, the attack uses the fact that

$$\varphi^{-1}(\varphi(\mathfrak{m})) \neq \mathfrak{m} \text{ if } N(\mathfrak{m}) > \sqrt{|\Delta_1|/3}.$$

3.1 Relation Involving a Single Chosen Ciphertext

The main idea behind our attack is to use a message $\mathfrak{m}$ slightly longer than proper messages and hope that in $\mathcal{O}_{\Delta_1}$ the corresponding ideal will be a single reduction step away from a reduced ideal. Note that after multiplication by a power of $\mathfrak{p}$ (in NICE) or $\mathfrak{a}$ (in HJPT), there is no way for the deciphering process to distinguish a correct ciphertext from an incorrect one, and thus to detect this attack. Of course, if one add some verification bits to the message, then it becomes feasible to make this distinction. This will be further discussed in section 3.3. In order to attack the system we need to make explicit the relation between the original message and the decoded message.

Let $\mathfrak{m} = (m, n) \in Cl(\Delta_q)$ be a message such that

$$\varphi_q^{-1}(\varphi_q(\mathfrak{m})) \neq \mathfrak{m}.$$

It means that $\Phi_q(\mathfrak{m})$ is not reduced in $\mathcal{O}_{\Delta_1}$. If we further suppose that a single reduction step is needed to reduce $\Phi_q(\mathfrak{m})$, we can make precise the relation between $\mathfrak{m}$ and $\varphi^{-1}(\varphi(\mathfrak{m}))$.

We apply the decryption algorithm and one reduction step as described in section 2.1 to $\mathfrak{m}$ and instead of finding $\mathfrak{m}$, we obtain $\mathfrak{m}' = (m', n')$ where (m', n') satisfies :

$$\begin{cases} m' = \frac{N^2 - \Delta_1}{4m} \\ n' \equiv -Nq \pmod{2m'} \end{cases}$$

and N is an integer that satisfies $-m < N < m$.

3.2 How to Find a Suitable Ciphertext

In order to be sure that a given message (m, n) will not be reduced in $Cl(\Delta_1)$ we need to take $m > \sqrt{|\Delta_1|/3}$. Moreover, from [3, p239], we know that if (m, n) is an ideal of $\mathcal{O}_{\Delta_1}$ such that $-m < n \leq m$ and $m < \sqrt{|\Delta_1|}$. Then either (m, n)

is already reduced, or the ideal (o, r) where $o = \frac{n^2 - \Delta_1}{2m}$ and $-n = 2ko + r$ with $-o < r \leq o$, obtained by one reduction step, will be reduced.

In order to be sure that our ciphertext will have the described properties, we need to choose m as follows :

$$\sqrt{|\Delta_1|/3} < m < \sqrt{|\Delta_1|}.$$

Since we can estimate Δ_1 to be approximately $\sqrt[3]{|\Delta_q|}$, m should be of the size of $\sqrt[6]{|\Delta_q|}$. Moreover, the maximum size of possible messages, that is the bit length of $\lfloor \sqrt{|\Delta_1|/4} \rfloor$, is public, thus giving us the bit length of $\sqrt{|\Delta_1|/3}$. With this information, we need only two ciphertexts in the correct range to break the system. However, if the given maximum size has been underestimated, it may be that our first try for m we will still be in the allowed range of correct decryption. We may thus have to decrypt a few more messages, multiplying steps of $\sqrt{3}$ before finding a suitable m.

3.3 Using Two Chosen Ciphertexts

With only one pair $(\mathfrak{m}, \mathfrak{m}')$ we cannot find Δ_1, but with two such pairs $(\mathfrak{m}_1, \mathfrak{m}'_1)$, $(\mathfrak{m}_2, \mathfrak{m}'_2)$, we have :

$$\begin{cases} m'_1 = \frac{N_1^2 - \Delta_1}{4m_1} \\ m'_2 = \frac{N_2^2 - \Delta_1}{4m_2} \end{cases}$$

and then :

$$p = -\Delta_1 = 4m_1 m'_1 - N_1^2 = 4m_2 m'_2 - N_2^2.$$

We need to find N_1,N_2. That means we have to find an integer solution of the equation :

$$4m_1 m'_1 - 4m_2 m'_2 = N_1^2 - N_2^2.$$

Let $X = N_1 + N_2$, $Y = N_1 - N_2$ and $k = 4m_1 m'_1 - 4m_2 m'_2$, the equation now is :

$$k = XY,$$

where k is known and X, Y unknown. Once X is found, we can easily compute N_1, N_2 and p. Since X is a factor of k, it suffices to factor k and try every divisor as a possible value for X. Since the number of factors of k is quite small, the possible X can be tested in a reasonable amount of time. When $4m_1 m'_1 - N_1^2 = 4m_2 m'_2 - N_2^2$, there is a high probability that we have found the correct X. We just need to check that this value of Δ_1 divides Δ_q.

The size of k is approximately the size of m^2. As we choose m approximately of size $\sqrt[6]{|\Delta_q|}$, the size of k is $\sqrt[3]{|\Delta_q|}$. With the parameters given in [8], p, q and k all have 256 bits, thus k is easy to factor and the attack succeeds.

If we want to prevent the attack from succeeding, we need a size for k that prevent its factorization. Since k is an ordinary number, it may well have many small factors. Moreover, using more ciphertexts, we may choose between many values of k one that factors easily. This means that, for the algorithms to be

secure, k should have at the very least 768 bits. We would then have keys of 2304 bits.

To repair the cryptosystem, one could add redundancy to the plaintext, before encryption. If after decryption, the obtained message does not have this redundancy, the output is discarded, thus preventing someone from feeding wrong messages to the decryption algorithm. However, this should preferably be done in a provably secure way. Ideas from the OAEP work of Bellare and Rogaway [1] may be of use. However, as usual with this approach, it will decrease the number of information bits in the message.

4 Example

The example described in this section is based on the NICE cryptosystem. In [8], it is suggested that security should be assured for the factorization attack if p and q are larger than 256 bits and if Δ_q is larger than 768 bits. In our example, we took Δ_q of 770 bits, a p of 256 bits and a q of 257 bits.

Public key :

$$\begin{aligned}\Delta_q = &-1001133619402846750073919037082619174565372425946674915149340539464219927955168182167600836407521987097\\&2619973270184386441185324964453536572880202249818566\\&5592983708546453282107912775914256762913490132215200\\&22224671621236001656120923\end{aligned}$$

$$\begin{aligned}\mathfrak{p} &= (a, b)\\ a &= 5702268770894258318168588438117558871300783180769995195092715895755173700399141486895731384747\\ b &= -3361236040582754784958586298017949110648731745605930164666819569606755029773074415823039847007\end{aligned}$$

Messages used for the attack :

$$\begin{aligned}\mathfrak{m}_1 &= (m_1, n_1)\\ m_1 &= 5809514784174292431747787277630205686535809514784174292431747787277630205689811\\ n_1 &= 2132637274650808372604967710816406514351\end{aligned}$$

$$\begin{aligned}\mathfrak{m}_2 &= (m_2, n_2)\\ m_2 &= 5809514784174292431747787277630205689815809514784174292431747787277630205689811\\ n_2 &= 5510635055886459952993916901849848021191\end{aligned}$$

Decoded messages :

$$\begin{aligned}\mathfrak{m}'_1 &= (m'_1, n'_1)\\ m'_1 &= 8345669710339337494972659453786147486911\\ n'_1 &= 7867165323191132309340559971888017205711\end{aligned}$$

$$\begin{aligned}\mathfrak{m}_2' &= (m_2', n_2')\\ m_2' &= 831363826969102043969673088753836977 67\\ n_2' &= -799132302773000590439366599288209128 89\end{aligned}$$

That gives us a value for k :

$$\begin{aligned}k = 744348512019197260111329217472677897277 06\\ 600792815510352758060887027806412 0\end{aligned}$$

k is factored into :

$$\begin{aligned}k = 2^3 * 3 * 5 * 11 * 211 * 557 * 4111 * 155153 * 24329881\\ *28114214269943 * 413746179653057\\ *26580133430529627286021\end{aligned}$$

For the following values of X and Y :

$$\begin{aligned}X &= 2^2 * 5 * 11 * 211 * 557 * 4111 * 155153 * 24329881\\ &\quad *413746179653057\\ &= 16601295001642548056622403660641267734 0\\ Y &= 2 * 3 * 28114214269943 * 26580133430529627286021\\ &= 44836773995375102009813566857862008 18\end{aligned}$$

We found :

$$\begin{aligned}p = 18666989127415343787417570818050325965428159 31\\ 03800953935381353078144162357587\end{aligned}$$

5 Conclusion

Since the discrete logarithm problem in the class group of imaginary quadratic order is a difficult problem (see [6]), it was tempting to build public key cryptosystems on it. However, for performance sake, it was necessary to add more structure, and make use of the canonical surjection from $\mathcal{O}_{\Delta_q}$ to $\mathcal{O}_{\Delta_1}$. Unfortunately, this additional structure opens a way to the chosen-ciphertext attack that was described here.

Nonetheless, the discrete logarithm in class groups is an interesting problem that might yet find other applications to public key cryptography.

References

1. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption – how to encrypt with RSA. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

2. Dan Boneh, Glenn Durfee, and Nick Howgrave-Graham. Factoring $n = pq^r$ for large r. In M. Wiener, editor, *Advances in Cryptology — CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 326–337. Springer, 1999.
3. Henri Cohen. *A Course in Computational Algebraic Number Theory.* Graduate Texts in Mathematics. Springer, 1995.
4. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.
5. M. Hartmann, S. Paulus, and T. Takagi. Nice - new ideal coset encryption. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES'99*, pages 341–352. Springer Verlag, 1999. Pre-proceedings, final proceedings to appear in LNCS.
6. Hühnlein and Takagi. Reducing logarithms in totally non-maximal orders to logarithms in finite fields. In *ASIACRYPT'99*, 1999.
7. D. Hühnlein, M. J. Jacobson, S. Paulus Jr., and T. Takagi. A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption. In *Avances in Cryptology — EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 294–307, 1998.
8. Sachar Paulus and Tsuyoshi Takagi. A new public-key cryptosystem over quadratic orders with quadratic decryption time. to appear in Journal of Cryptology.

Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*

Nicolas Courtois[1,3], Alexander Klimov[2], Jacques Patarin[3], and Adi Shamir[4]

[1] MS/LI, Toulon University, BP 132, F-83957 La Garde Cedex, France
courtois@univ-tln.fr
[2] Dept. of Appl. Math. & Cybernetics, Moscow State University, Moscow, Russia
ask@ispras.ru
[3] Bull CP8, 68, route de Versailles, BP45, 78431 Louveciennes Cedex, France
J.Patarin@frlv.bull.fr
[4] Dept. of Applied Math. The Weizmann Institute of Science, Rehovot 76100, Israel
shamir@wisdom.weizmann.ac.il

Abstract. The security of many recently proposed cryptosystems is based on the difficulty of solving large systems of quadratic multivariate polynomial equations. This problem is NP-hard over any field. When the number of equations m is the same as the number of unknowns n the best known algorithms are exhaustive search for small fields, and a Gröbner base algorithm for large fields. Gröbner base algorithms have large exponential complexity and cannot solve in practice systems with $n \geq 15$. Kipnis and Shamir [9] have recently introduced a new algorithm called "relinearization". The exact complexity of this algorithm is not known, but for sufficiently overdefined systems it was expected to run in polynomial time.

In this paper we analyze the theoretical and practical aspects of relinearization. We ran a large number of experiments for various values of n and m, and analysed which systems of equations were actually solvable. We show that many of the equations generated by relinearization are linearly dependent, and thus relinearization is less efficient that one could expect. We then develop an improved algorithm called XL which is both simpler and more powerful than relinearization. For all $0 < \epsilon \leq 1/2$, and $m \geq \epsilon n^2$, XL and relinearization are expected to run in polynomial time of approximately $n^{\mathcal{O}(1/\sqrt{\epsilon})}$. Moreover, we provide strong evidence that relinearization and XL can solve randomly generated systems of polynomial equations in subexponential time when m exceeds n by a number that increases slowly with n.

1 Introduction

In this paper we consider the problem of solving systems of multivariate polynomial equations. This problem is NP-complete even if all the equations are quadratic and the field is $GF(2)$. It has many applications in cryptography, since

* An extended version of this paper is available from the authors.

a large number of multivariate schemes had been proposed (and cryptanalysed) over the last few years. In addition, the problem arises naturally in other subareas of Mathematics and Computer Science, such as optimization, combinatorics, coding theory, and computer algebra.

The classical algorithm for solving such a system is Buchberger's algorithm for constructing Gröbner bases, and its many variants (see, e.g., [1]). The algorithm orders the monomials (typically in lexicographic order), and eliminates the top monomial by combining two equations with appropriate polynomial coefficients. This process is repeated until all but one of the variables are eliminated, and then solves the remaining univariate polynomial equation (e.g., by using Berlekamp's algorithm over the original or an extension field). Unfortunately, the degrees of the remaining monomials increase rapidly during the elimination process, and thus the time complexity of the algorithm makes it often impractical even for a modest number of variables. In the worst case Buchberger's algorithm is known to run in double exponential time, and on average its running time seems to be single exponential. The most efficient variant of this algorithm which we are aware of is due to Jean-Charles Faugere (private communication [5, 6]) whose complexity in the case of $m = n$ quadratic equations is:

- If K is big, the complexity is proved to be $\mathcal{O}(2^{3n})$ and is $\mathcal{O}(2^{2.7n})$ in practice.
- When K =GF(2), the complexity is about $\mathcal{O}(2^{2n})$ (which is worse than the $O(2^n)$ complexity of exhaustive search).

In practice, even this efficient variant cannot handle systems of quadratic equations with more than about $n = 15$ variables.

In this paper we are interested in the problem of solving overdefined systems of multivariate polynomial equations in which the number of equations m exceeds the number of variables n. Random systems of equations of this type are not expected to have any solutions, and if we choose them in such a way that one solution is known to exist, we do not expect other interference solutions to occur. We are interested in this type of systems since they often occur in multivariate cryptographic schemes: if the variables represent the cleartext then we want the decryption process to lead to a unique cleartext, and if the variables represent the secret key we can typically write a large number of polynomial equations which relate it to the known public key, to the cleartexts, and to the ciphertexts.

Gröbner base techniques do not usually benefit from the fact that the number of equations exceeds the number of variables, since they proceed by sequentially eliminating a single monomial from a particular pair of equations. Unfortunately, this cryptographically important case received very little attention in the vast literature on Gröbner base algorithms. To see that much better algorithms exist in this case, consider a system of $n(n+1)/2$ random homogeneous quadratic equations in n variables $x_1, ...x_n$. The well known linearization technique replaces each product $x_i x_j$ by a new independent variable y_{ij}. The quadratic equations give a system of $n(n+1)/2$ linear equations in $n(n+1)/2$ variables which can be solved efficiently by Gauss elimination. Once we find all the y_{ij} values, we can find two possible values for each x_i by extracting the square root of y_{ii} in the field, and use the values of y_{ij} to combine correctly the roots of y_{ii} and y_{jj}.

At Crypto 99, Kipnis and Shamir [9] introduced a new method for solving overdefined systems of polynomial equations, called *relinearization*. It was designed to handle systems of ϵn^2 quadratic equations in n variables where ϵ is smaller than $1/2$. The basic idea of relinearization is to add to the given system of linear equations in the y_{ij} additional nonlinear equations which express the fact that these variables are related rather than independent. In its simplest form, relinearization is based on the commutativity of multiplication of 4-tuples of variables: For any a, b, c, d, $(x_a x_b)(x_c x_d) = (x_a x_c)(x_b x_d) = (x_a x_d)(x_b x_c)$ and thus $y_{ab} y_{cd} = y_{ac} y_{bd} = y_{ad} y_{bc}$. There are several generalizations of relinearization, including higher degree variants and a recursive variant. The relinearization technique can solve many systems of equations which could not be solved by linearization, but its exact complexity and success rate are not well understood.

In the first part of this paper, we analyse the theoretical and practical aspects of the relinearization technique. We concentrate in particular on the issue of the linear independence of the generated equations, and show that many of the generated equations are provably dependent on other equations, and can thus be eliminated. This reduces the size of the linearized systems, but also limits the types of polynomial equations which can be successfully solved by the technique.

In the second part of the paper, we introduce the XL (eXtended Linearization) technique which can be viewed as a combination of bounded degree Gröbner bases and linearization. The basic idea of this technique is to generate from each polynomial equation a large number of higher degree variants by multiplying it with all the possible monomials of some bounded degree, and then to linearize the expanded system. This is a very simple technique, but we prove that it is at least as powerful as relinearization. We analyse the time complexity of the XL technique, and provide strong theoretical and practical evidence that the expected running time of this technique is:

- Polynomial when the number m of (random) equations is at least ϵn^2, and this for all $\epsilon > 0$.
- Subexponential if m exceeds n even by a small number.

If the size of the underlying field is not too large, we can sometimes apply this subexponential technique even to an underdefined (or exactly defined) systems of equations by guessing the values of some of the variables and simplifying the resulting equations.

2 Experimental Analysis of the Relinearization technique

In this part we concentrate on systems of randomly generated homogeneous quadratic equations of the form:

$$\sum_{1 \le i \le j \le n} a_{ijk} x_i x_j = b_k, \qquad k = 1 \ldots m \tag{1}$$

The general idea of the *relinearization* method is to first use linearization in order to solve the system of m linear equations in the $n(n+1)/2$ variables

$y_{ij} = x_i x_j$. The system is typically underdefined, and thus we express each y_{ij} as a linear combination of $l < n(n+1)/2$ new parameters $t_1, \ldots, t_l$. We then create additional equations which express the commutativity of the multiplication of x_i which can be paired in different orders. Let $(a, b, c, d, \ldots, e, f) \sim (a', b', c', d', \ldots, e', f')$ denote that the two tuples are permuted versions of each other. Then:

$$(x_a x_b)(x_c x_d)...(x_e x_f) = (x_{a'} x_{b'})(x_{c'} x_{d'})...(x_{e'} x_{f'}) \tag{2}$$

This can be viewed as an equation in the y_{ij} variables, and thus also as an equation in the (smaller number of) parameters t_s expressing them. The new system of equations derived from all the possible choices of tuples of indices and their permutations can be solved either by another linearization or by recursive relinearization.

2.1 Degree 4 relinearization

We have applied the degree 4 relinearization technique to a large number of systems of randomly generated homogeneous quadratic equations of various sizes. We always got linearly independent equations (except when the field was very small). For several small values of n, the critical number of equations which make the system (barely) solvable is summarized in the following table: Assuming the

Table 1. Fourth degree relinearization

n	m	l	n'	m'
6	8	13	104	105
8	12	24	324	336
10	16	39	819	825
15	30	90	4185	4200

n Number of variables in original quadratic system
m Number of equations in original quadratic system
l Number of parameters in the representation of the y_{ij}
n' Number of variables in the final linear system
m' Number of equations in the final linear system

linear independence of the derived equations (which was experimentally verified), we can easily derive the asymptotic performance of degree 4 relinearization for large n: The method is expected to find the solution (in polynomial time) whenever the number of equations exceeds ϵn^2 for $\epsilon > 1/2 - 1/\sqrt{6} \approx 0.1$. This case is thus well understood.

2.2 Higher degree relinearization

The problem becomes much more complicated when we consider degree 6 relinearizations, which are based on all the equations of the form:

$$y_{ab}y_{cd}y_{ef} = y_{gh}y_{ij}y_{kl}, \quad \text{where } (a,b,c,d,e,f) \sim (g,h,i,j,k,l) \tag{3}$$

Note that these equations are cubic in the free parameters t_s (even if the original equations are quadratic), so we need many more equations to relinearize it successfully.

Unlike the case of degree 4 relinearizations, many of these equations were experimentally found to be linearly dependent. We have identified several distinct causes of linear dependence, but its complete characterization is still an open research problem.

We first have to eliminate trivial sources of linear dependence. We only have to consider 6-tuples of indices (a,b,c,d,e,f) which are sorted into non-decreasing order within each successive pair (a,b), (c,d), (e,f), and then into non-decreasing lexicographic order on these pairs. For 6-tuples which contain 6 distinct indices such as (0,1,2,3,4,5), we get 15 (rather than $6! = 720$) legal permutations:

$$\begin{array}{lll}
(0,1,\ 2,3,\ 4,5) & (0,1,\ 2,4,\ 3,5) & (0,1,\ 2,5,\ 3,4) \\
(0,2,\ 1,3,\ 4,5) & (0,2,\ 1,4,\ 3,5) & (0,2,\ 1,5,\ 3,4) \\
(0,3,\ 1,2,\ 4,5) & (0,3,\ 1,4,\ 2,5) & (0,3,\ 1,5,\ 2,4) \\
(0,4,\ 1,2,\ 3,5) & (0,4,\ 1,3,\ 2,5) & (0,4,\ 1,5,\ 2,3) \\
(0,5,\ 1,2,\ 3,4) & (0,5,\ 1,3,\ 2,4) & (0,5,\ 1,4,\ 2,3)
\end{array}$$

so we can create 14 possible equations. But for the 6-tuple $(0,1,1,1,1,2)$, there are only 2 legal permutations $(0,1,\ 1,1,\ 1,2)$ and $(0,2,\ 1,1,\ 1,1)$ and thus we get only one equation. In general, there are 32 types of repetition of values in the given 6-tuple, and each one of them gives rise to a different number of equations. Table (2) summarizes the number of non-trivial equations which can actually be formed using 6-tuples for small values of n.

Table 2. Number of non trivial equations defined by 6-tuples

n	equations
4	136
5	470
6	1309
7	3136
8	6720
9	13212
10	24255
11	42108
12	69784
20	1388520

2.3 Eliminating redundant linear equations

In this section we show that most of the non-trivial equations defined so far are redundant, since they can be linearly derived from other equations. Consider a typical non-trivial equation generated by degree r relinearization:

$$\begin{gathered} y_{i_1 i_2} y_{i_3 i_4} \cdots y_{i_{r-1} i_r} = y_{j_1 j_2} y_{j_3 j_4} \cdots y_{j_{r-1} j_r} \\ \text{with} \quad (i_1, \ldots, i_r) \sim (j_1, \ldots, j_r) \end{gathered} \tag{4}$$

We call such an equation *special* if the lists of y's are the same on both sides of the equation, except for exactly two y's whose indices are permuted. For example, the non-trivial equation

$$y_{01} y_{23} y_{45} y_{67} y_{89} = y_{01} y_{27} y_{36} y_{45} y_{89} \tag{5}$$

is special since 3 out of the 5 terms are common in the two expressions. For large n only a small fraction of the equations are special, but we can prove:

Lemma: The set of special equations linearly span the set of all the non-trivial equations for the same relinearization degree.

Proof (sketch): Consider two particular permutations A and B of the same r-tuple of indices, which define one of the possible equations. A basic property of permutation groups is that any permutation can be derived by a sequence of transpositions which affect only adjacent elements. Consider the pairing of consecutive indices which defines the sequence of y's. Applying a single transposition of adjacent indices can permute the indices of at most two y's, and thus we can derive the equality of the product of y's for any two permuted versions of some subset of indices from the transitivity of the equality in special equations.

To further reduce the number of equations, recall that each y_{ij} variable is a linear combination of a smaller number of parameters t_s. Instead of having all the possible common products of y_{ij} variables on both sides of the equation, it suffices to consider only common products of t_s parameters, since each product of the first type is expressible as a linear combination of products of the second type. We can thus consider only the smaller number of equations of the form:

$$y_{ab} y_{cd} t_e t_f \cdots t_g = y_{ac} y_{bd} t_e t_f \cdots t_g = y_{ad} y_{bc} t_e t_f \cdots t_g \tag{6}$$

The common t's on both sides of the equation seem to be cancellable, and thus we are led to believe that degree r relinearization is just a wasteful representation of degree 4 relinearization, which can solve exactly the same instances. However, division by a variable is an algebraic rather than linear operation, and thus we cannot prove this claim. The surprising fact is that these seemingly unnecessary common variables are very powerful, and in fact, they form the basis for the XL technique described in the second part of this paper. As a concrete example, consider a slightly overdefined system of 10 quadratic equations in 8 variables. Experiments have shown that it can be solved by degree 6 relinearization, whereas degree 4 relinearizations need at least 12 quadratic equations in 8 variables. Other combinations of solvable cases are summarized in table 3.

As indicated in this table, even the equations derived from special equations are still somewhat dependent, since we need more equations than variables in the final linear system. We have found several other sources of linear dependence, but due to space limitations we cannot describe them in this extended abstract.

Table 3. Experimental data for degree 6 relinearization

n	m	l	n'	m"
4	8	2	9	9
4	7	3	19	19
4	6	4	34	40
4	5	5	55	86
5	9	6	83	83
5	8	7	119	129
5	7	8	164	215
5	6	9	219	443
6	10	11	363	394
6	9	12	454	548
6	8	13	559	806
6	7	14	679	1541
7	11	17	1139	1363
7	10	18	1329	1744
7	9	19	1539	2318
8	12	24	2924	3794
8	11	25	3275	4584
8	10	26	3653	5721
9	13	32	6544	9080
9	12	33	7139	10567
9	11	34	7769	12716

n Number of variables in the original quadratic system
m Number of equations in the original quadratic system
l Number of parameters in the representation of the y_{ij}
n' Number of variables in the final linear system
m" number of equations which were required to solve the final linear system

3 The XL Algorithm

We present another algorithm for solving systems of multivariate polynomial equations called XL (which stands for eXtended Linearizations, or for multiplication and linearization). As we will see, each independent equation obtained by relinearization exists (in a different form) in XL, and thus XL can be seen as a simplified and improved version of relinearization.

Let K be a field, and let $\mathcal{A}$ be a system of multivariate quadratic equations $l_k = 0 \quad (1 \leq k \leq m)$ where each l_k is the multivariate polynomial $f_k(x_1, \ldots, x_n) - b_k$.

The problem is to find at least one solution $x = (x_1, \ldots, x_n) \in K^n$, for a given $b = (b_1, \ldots, b_n) \in K^m$.

We say that the equations of the form $\prod_{j=1}^{k} x_{i_j} * l_i = 0$ are of type $x^k l$, and we call $x^k l$ the set of all these equations. For example the initial equations $\mathcal{I}_0 = \mathcal{A}$ are of type l.

We also denote by x^k the set of all terms of degree k, $\prod_{j=1}^{k} x_{i_j}$. It is a slightly modified extension of the usual convention $x = (x_1, \ldots, x_n)$.

Let $D \in \mathbb{N}$. We consider all the polynomials $\prod_j x_{i_j} * l_i$ of total degree $\leq D$.

Let $\mathcal{I}_D$ be the set of equations they span. $\mathcal{I}_D$ is the linear space generated by all the $x^k l,\ 0 \leq k \leq D-2$.

$\mathcal{I}_D \subset \mathcal{I}$, $\mathcal{I}$ being the ideal spanned by the l_i (could be called $\mathcal{I}_\infty$).

The idea of the XL algorithm is to find in some $\mathcal{I}_D$ a set of equations which is easier to solve than the initial set of equations $\mathcal{I}_0 = \mathcal{A}$. As we show later, the XL algorithm with maximal degree D completely contains the relinearization technique of degree D.

Definition 1 (The XL algorithm) *Execute the following steps:*

1. **Multiply:** *Generate all the products* $\prod_{j=1}^{k} x_{i_j} * l_i \in \mathcal{I}_D$ *with* $k \leq D-2$.
2. **Linearize:** *Consider each monomial in* x_i *of degree* $\leq D$ *as a new variable and perform Gaussian elimination on the equations obtained in 1. The ordering on the monomials must be such that all the terms containing one variable (say* x_1*) are eliminated last.*
3. **Solve:** *Assume that step 2 yields at least one univariate equation in the powers of* x_1*. Solve this equation over the finite fields (e.g., with Berlekamp's algorithm).*
4. **Repeat:** *Simplify the equations and repeat the process to find the values of the other variables.*

The XL algorithm is very simple, but it is not clear for which values of n and m it ends successfully, what is its asymptotic complexity, and what is its relationship to relinearization and Gröbner base techniques. As we will see, despite it's simplicity XL may be one of the best algorithms for randomly generated overdefined systems of multivariate equations.

Note 1: The equations generated in XL are in $x^k l$ and belong to $\mathcal{I}$, the ideal generated by the l_i. There is no need to consider more general equations such as l_1^2 since they are in $\mathcal{I}_4$ and are thus in the linear space generated by the equations of type $x^2 l \cup xl \cup l$.

Note 2: Sometimes it is more efficient to work only with a subset of all the possible monomials. For example, when all the equations are homogeneous quadratic equations, it suffices to use only monomials of odd (or even) degrees.

Note 3: A related technique was used by Don Coppersmith to find small roots of univariate modular equations [2]. However, in that application he used

LLL rather than Gauss elimination to handle the generated relations, and relied heavily on the fact that the solution is small (which plays no role in our application).

4 A toy example of XL

Let $\mu \neq 0$. Consider the problem of solving:

$$\begin{cases} x_1^2 + \mu x_1 x_2 = \alpha & (4.1) \\ x_2^2 + \nu x_1 x_2 = \beta & (4.2) \end{cases}$$

For $D = 4$ and even degree monomials, the equations we generate in step 1 of the XL algorithm are $l \cup x^2 l$. Those are the 2 initial equations and $6 = 2 * 3$ additional equations generated by multiplying the initial 2 equations l_i by the 3 possible terms of degree 2: $x_1^2, x_1 x_2, x_2^2 \in x^2$.

$$\begin{cases} x_1^4 + \mu x_1^3 x_2 = \alpha x_1^2 & (4.3) \\ x_1^2 x_2^2 + \nu x_1^3 x_2 = \beta x_1^2 & (4.4) \\ x_1^2 x_2^2 + \mu x_1 x_2^3 = \alpha x_2^2 & (4.5) \\ x_2^4 + \nu x_1 x_2^3 = \beta x_2^2 & (4.6) \\ x_1^3 x_2 + \mu x_1^2 x_2^2 = \alpha x_1 x_2 & (4.7) \\ x_1 x_2^3 + \nu x_1^2 x_2^2 = \beta x_1 x_2 & (4.8) \end{cases}$$

In step 2 we eliminate and compute:

From (4.1): $x_1 x_2 = \frac{\alpha}{\mu} - \frac{x_1^2}{\mu}$;

From (4.2): $x_2^2 = (\beta - \frac{\alpha\nu}{\mu}) + \frac{\nu}{\mu} x_1^2$;

From (4.3): $x_1^3 x_2 = \frac{\alpha}{\mu} x_1^2 - \frac{x_1^4}{\mu}$;

From (4.4): $x_1^2 x_2^2 = (\beta - \frac{\alpha\nu}{\mu}) x_1^2 + \frac{\nu}{\mu} x_1^4$;

From (4.8): $x_1 x_2^3 = \frac{\alpha\beta}{\mu} + (\frac{\alpha\nu^2}{\mu} - \beta\nu - \frac{\beta}{\mu}) x_1^2 - \frac{\nu^2}{\mu} x_1^4$;

From (4.6): $x_2^4 = (\beta^2 - \frac{2\alpha\beta\nu}{\mu}) + (\frac{2\nu\beta}{\mu} + \beta\nu^2 - \frac{\alpha\nu^2}{\mu}) x_1^2 + \frac{\nu^3}{\mu} x_1^4$;

Finally from (4.5) we get one equation with only one variable x_1:

$$\alpha^2 + x_1^2(\alpha\mu\nu - \beta\mu^2 - 2\alpha) + x_1^4(1 - \mu\nu) = 0.$$

5 Experimental results on XL

5.1 Experimental results with $m = n$ over GF(127)

When $m = n$ our simulation has shown that we need $D = 2^n$ in order to be able to solve the equations (so the algorithm works only for very small n).

An explanation of this is given in the Sect. 6.2.

3 variables and 3 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	3/3	-3	1	6	x^2
$xl \cup l$	12/12	-5	3	19	$x^3 \cup x^2 \cup x$
$x^3l \cup xl$	30/39	-2	3	34	$x^5 \cup x^3 \cup x$
$x^5l \cup x^3l \cup xl$	66/102	-1	4	70	$x^7 \cup x^5 \cup x^3 \cup x$
$x^6l \cup x^4l \cup x^2l \cup l$	91/150	0	4	94	$x^8 \cup x^6 \cup x^4 \cup x^2$
$x^7l \cup x^5l \cup x^3l \cup xl$	121/210	0	5	5	$125\ x^9 \cup x^7 \cup x^5 \cup x^3 \cup x$
$x^{17}l \cup x^{15}l \cup x^{13}l \cup \ldots$	821/1845	4	9	825	$x^{19} \cup x^{17} \cup x^{15} \cup \ldots$

4 variables and 4 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	4/4	-6	1	10	x^2
$x^4l \cup x^2l \cup l$	122/184	-5	3	129	$x^6 \cup x^4 \cup x^2$
$x^8l \cup x^6l \cup x^4l \cup x^2l \cup l$	573/1180	-3	5	580	$x^{10} \cup x^8 \cup x^6 \cup x^4 \cup x^2$
$x^{12}l \cup x^{11}l \cup x^{10}l \cup \ldots$	3044/7280	-2	14	3059	$x^{14} \cup \ldots$
$x^{14}l \cup x^{12}l \cup x^{10}l \cup \ldots$	2677/6864	0	8	2684	$x^{16} \cup x^{14} \cup x^{12} \cup \ldots$

T: number of monomials $\Delta \geq 0$ when XL solves the equations, (Δ = Free+B-T-1)

B: nb. of monomials in one variable e.g. x_1 **Free/All**: numbers of free/all equations of given **type**

5.2 Experimental results with $m = n + 1$ over GF(127)

When $m = n + 1$ our simulations show that we have to take $D = n$ in order to obtain $\Delta \geq 0$ and be able to solve the equations.

4 variables and 5 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	5/5	-4	1	10	x^2
$xl \cup l$	25/25	-8	3	34	$x^3 \cup x^2 \cup x$
$x^2l \cup l$	45/55	1	2	45	$x^4 \cup x^2$

8 variables and 9 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	9/9	-27	1	36	x^2
$x^2l \cup l$	297/333	-68	2	366	$x^4 \cup x^2$
$x^4l \cup x^2l \cup l$	2055/3303	-25	3	2082	$x^6 \cup x^4 \cup x^2$
$x^5l \cup x^3l \cup xl$	4344/8280	-5	4	4352	$x^7 \cup x^5 \cup x^3 \cup x$
$x^6l \cup x^4l \cup x^2l \cup l$	8517/18747	3	4	8517	$x^8 \cup x^6 \cup x^4 \cup x^2$

T: number of monomials $\Delta \geq 0$ when XL solves the equations, (Δ = Free+B-T-1)
B: nb. of monomials in one variable e.g. x_1 **Free/All**: numbers of free/all equations of given **type**

5.3 Experimental results with $m = n + 2$ over GF(127)

In case $m = n + 2$ it may be possible to take $D = \sqrt{n} + C$ but the data is still inconclusive. We are currently working on larger simulations, which will be reported in the final version of this paper.

8 variables and 10 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	10/10	-26	1	36	x^2
$x^2l \cup l$	325/370	-40	2	366	$x^4 \cup x^2$
$x^3l \cup xl$	919/1280	1	3	920	$x^5 \cup x^3 \cup x$

9 variables and 11 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	11/11	-34	1	45	x^2
$x^3l \cup xl$	1419/1914	-40	3	1461	$x^5 \cup x^3 \cup x$
$x^4l \cup x^2l \cup l$	3543/5951	2	3	3543	$x^6 \cup x^4 \cup x^2$

T: number of monomials $\Delta \geq 0$ when XL solves the equations, (Δ = Free+B-T-1)
B: nb. of monomials in one variable e.g. x_1 **Free/All**: numbers of free/all equations of given **type**

6 Complexity evaluation of XL

Given m quadratic equations with n variables, we multiply each equation by all the possible $x_{i_1} \cdot \ldots \cdot x_{i_{D-2}}$. The number of generated equations (of type $x^{D-2}l$) is about $\alpha = \frac{n^{D-2}}{(D-2)!} \cdot m$ while we have about $\beta = \frac{n^D}{D!}$ linear variables of type $x^D \cup x^{D-2}$.

If most of the equations are linearly independent in XL (we will comment on this critical hypothesis below), we expect to succeed when $\alpha \geq \beta$, i.e. when

$$m \geq \frac{n^2}{D(D-1)} \tag{7}$$

We get the following evaluation

$$D \geq \text{ about } \frac{n}{\sqrt{m}}. \tag{8}$$

6.1 Case $m \approx n$

If $m \approx n$, and if we expect most of the equations to be independent, we expect the attack to succeed when $D \approx \sqrt{n}$. The complexity of the algorithm is thus lower bounded by the complexity of a Gaussian reduction on about $\frac{n^D}{D!}$ variables, $D \approx \sqrt{n}$. Its working factor is thus at least

$$WF \geq \left(\frac{n^{\sqrt{n}}}{\sqrt{n}!} \right)^{\omega}$$

where $\omega = 3$ in the usual Gaussian reduction algorithm, and $\omega = 2.3766$ in improved algorithms. By simplifying this expression, we get the subexponential complexity bound of approximately:

$$WF \geq e^{\omega \sqrt{n} (\frac{\ln n}{2} + 1)} \tag{9}$$

Notes:

- **When n is fixed** the XL algorithm is expected to run in polynomial time (in the size of K).
- **When K is fixed and** $n \to \infty$, the formula indicates that XL may run in sub-exponential time. We will see however that this is likely to be true only when $m - n$ is "sufficiently" big while still $m \approx n$. This point is the object of the study below.

6.2 Case $m = n$

When $m = n$ our simulation showed that $D = 2^n$ (instead of $D \approx \sqrt{n}$).

It is possible to give a theoretical explanation of this fact: If we look at the algebraic closure $\overline{K}^n$ of K we have generally 2^n solutions for a system of n equations with n variables. So the final univariate equation we can derive should be generally of degree 2^n.

6.3 Case $m = n + 1$

For $m = n + 1$ our simulations show that $D = n$ (instead of $\sqrt{n}$). The reason for this is not clear at present.

6.4 Case $m = n + C$, $C \geq 2$

For $m = n + C$, $C \geq 2$, it seems from our simulations that even for small values of C we will have $D \approx \sqrt{n}$. This remark will lead to the FXL algorithm below.

In order to know for what value of C it is reasonable to assume that $D \approx \sqrt{n}$ we need more simulations. Many of them will be included in the extended version of this paper, however given the limitated computing power available, the results does not give a precise estimation of C.

6.5 Case $m = \epsilon n^2$, $\epsilon > 0$

Let $0 < \epsilon \leq 1/2$ and $m = \epsilon n^2$. We expect XL to succeed when

$$D \approx \lceil 1/\sqrt{\epsilon} \rceil. \tag{10}$$

The working factor is in this case $WF \approx \frac{n^{\omega \lceil 1/\sqrt{\epsilon} \rceil}}{(\lceil 1/\sqrt{\epsilon} \rceil)!}$. So the algorithm is expected to be polynomial (in n) with a degree of about $\omega/\sqrt{\epsilon}$.

Remark: The fact that solving a system of $\epsilon \cdot n^2$ equations in n variables was likely to be polynomial was first suggested in [9]. Despite the fact that the relinearization is less efficient than what could have been expected, the complexity of solving ϵn^2 equations in n variables is still expected to be polynomial.

7 The FXL algorithm

In our simulations it is clear that when $m \approx n$, the smallest working degree D decreases dramatically when $m - n$ increases. For example, if $m = n$ then $D = 2^n$, if $m = n + 1$ then $D = n$, and if m is larger we expect to have $D \approx \sqrt{n}$.

We are thus led to the following extension of XL called FXL (which stands for Fixing and XL):

Definition 2 (The FXL algorithm)

1. *Fix μ variables (see below for the choice of μ).*
2. *Solve with XL the resultant system of m equations in $n - \mu$ variables.*

We choose the smallest possible μ such that in step 2 we have $D \approx \sqrt{n}$, in order to have minimal complexity in step 2.

The complexity of the FXL algorithm is $q^\mu e^{c\sqrt{n}lnn}$, as we have q^μ choices for μ variables in step 1, and XL is $e^{c\sqrt{n}lnn}$ for $D \approx \sqrt{n}$.

How μ increases when n increases is an open question. We can notice that if $\mu = \mathcal{O}(\sqrt{n})$, then the complexity of the FXL algorithm would be about $q^{\mathcal{O}(\sqrt{n})} e^{C\sqrt{n}lnn}$, which is approximately $e^{C\sqrt{n}(lnn+lnq)}$. Thus the FXL algorithm might be sub-exponential, even when $m = n$, but we have no rigorous proof of this conjecture.

8 XL and relinearization

We have formally proved that the set of equations defined by a successful relinearization of degree D is equivalent to a subset of equations derived from the XL algorithm with the same D. The proof is not difficult, but due to its length it will appear only in the extended version of this paper (available from the authors). It is based on a series of effective syntactic transformations on the system of equations $\mathcal{C}$ derived from the degree D relinearization of a given system of m quadratic equations in n variables. By eliminating redundant equations we get another system of equations $\mathcal{D}$, and by replacing each monomial in $\mathcal{D}$ by a new variable, we get a final system of equations denoted by $\mathcal{E}$. We then perform the following steps:

1. We replace $\mathcal{C}$ by another system $\mathcal{C}'$ that contains the same equations written in a 'special form'. We define the 'special degree' of such equations, and show that $SpecialDeg(\mathcal{C}') \leq D$.
2. We transform $\mathcal{C}'$ to $\mathcal{D}'$. We show that $\mathcal{D}'$ are the equations of $\mathcal{D}$ written in the special form, with $SpecialDeg(\mathcal{D}') \leq D$.
3. We transform $\mathcal{D}'$ to $\mathcal{E}'$, and show that $\mathcal{E}' \subset \mathcal{I}_D$.

Theorem 1 (Relinearization as a subcase of the XL algorithm.) *Let $\mathcal{C}$ be the equations obtained in a successful relinearization of degree D of a system of m quadratic equations with n variables. Then we can effectively construct a set of equations $\mathcal{E}$, that preserves the solvability of the system by Gaussian reduction, along with it's explicit expression $\mathcal{E}'$ as a subcase of the XL algorithm: $\mathcal{E}' \subset \mathcal{I}_D$.*

In practice, XL is more efficient than relinearization. For example, to solve 11 equations with 9 variables, relinearization requires the solution of a linear system with 7769 variables (see Table 3), whereas XL requires the solution of a system with only 3543 variables (see 5.3). Moreover, XL can use any D while relinearization can only use composite values of D. For example, to solve 10 quadratic equations with 8 variables we had to use the relinearization algorithm with $D = 6$, but the XL algorithm could use the smaller value of $D = 5$. Consequently, the system of linear equations derived from linearization had 3653 variables, while the system of linear equations derived from XL had only 919 variables (see 5.3).

9 Gröbner bases algorithms

One way of implementing the XL algorithm is to combine the equations in an organised way, rather than to multiply them by all the possible monomials. This would naturally lead to the classical Gröbner-bases algorithms.

We define $\mathcal{I}_{x_{i_1},\ldots,x_{i_j}}$ as a subspace of all the equations of $\mathcal{I}$ that can be written with just the variables $x_{i_1}, \ldots, x_{i_j}$. The XL method checks if there are any (univariate) equations in some $(\mathcal{I}_D)_{x_1}$.

The Gröbner bases algorithms construct a basis of a space of (univariate) equations in $\mathcal{I}_{x_1} = \bigcup_k (\mathcal{I}_k)_{x_1}$. However in order to get there, they compute successively bases of the $\mathcal{I}_{x_1,\ldots,x_k}$ for $k = n \ldots 1$.

It is not clear what is the best way to use Gröbner bases to solve our problem of overdefined systems of equations. A large number of papers have been written on Gröbner base techniques, but most of them concentrate either on the case of fields of characteristic 0, or look for solution in an algebraic closure of $\overline{K}^n$, and the complexity analysis of these algorithms is in general very difficult.

10 Cryptanalysis of HFE with XL/relinearization attacks

The HFE (Hidden Field Equations) cryptosystem was proposed at Eurocrypt 1996 [11]. Two different attacks were recently developed against it [3, 9], but

they do not compromise the practical security of HFE instances with well chosen parameters. Moreover it does not seem that these attacks can be extended against variations of the HFE scheme such as HFEv or HFEv$^-$ described in [8].

The first type of attack (such as the affine multiple attack in [11]) tries to compute the cleartext from a given ciphertext. It is expected to be polynomial when the degree d of the hidden polynomial is fixed, and not polynomial when $d = \mathcal{O}(n)$. In [3] Nicolas Courtois presented several improved attacks in this category, with an expected complexity of $n^{\mathcal{O}(ln(d))}$ (which is still not polynomial) instead of the original complexity of $n^{\mathcal{O}(d)}$.

A second line of attack tries to recover the secret key from the public key. The Kipnis-Shamir attack described in [9] was the first attack of this type. It is also expected to be polynomial when d is fixed but not polynomial when $d = \mathcal{O}(n)$.

To test the practicality of these attacks, consider the HFE "challenge 1" described in the extended version of [11] and in [4]. It is a trapdoor function over $GF(2)$ with $n = 80$ variables and $d = 96$. A direct application of the FXL to these 80 quadratic equations requires Gaussian reductions on about $80^9/9! \approx 2^{38}$ variables, and thus its time complexity exceeds the 2^{80} complexity of exhaustive search, in spite of its conjectured subexponential asymptotic complexity. The best attack on the cleartext (from [3]) is expected to run on "challenge 1" in time 2^{62}. The best attack on the secret key (from [9]) is expected to run in time 2^{152} when XL is used, and to take even longer when relinearization is used. A possible improvement of this attack (from [3], using sub-matrices) runs in time 2^{82}, which is still worse than the 2^{80} complexity of exhaustive search.

11 Conclusion

In this paper we studied the relinearization technique of Kipnis and Shamir, along with several improvements. We saw that in high degree relinearizations the derived equations are mostly linearly dependent, and thus the algorithm is much less efficient than originally expected.

We have related and compared relinearization to more general techniques, such as XL and Gröbner bases. We have proved that XL "contains" relinearization and demonstrated that it is more efficient in practice. We also concluded that the complexity of solving systems of multivariate equations drops rapidly when the number of equations exceeds the number of variables (even by one or two). Consequently, over a small field the FXL algorithm may be asymptotically subexponential even when $m = n$, since it guesses the values of a small number of variables in order to make the system of equations slightly overdefined. However in many practical cases with fixed parameters $m \approx n$, the best known algorithms are still close to exhaustive search.

Finally, when the number of equations m and the number of variables n are related by $m \geq \epsilon n^2$ for any constant $0 < \epsilon \leq 1/2$, the asymptotic complexity seems to be polynomial with an exponent of $\mathcal{O}(1/\sqrt{\epsilon})$.

References

1. Iyad A. Ajwa, Zhuojun Liu, and Paul S. Wang: "Grobner Bases Algorithm", ICM Technical Reports, February 1995.
   See http://symbolicnet.mcs.kent.edu/icm/reports/index1995.html.
2. Don Coppersmith: "Finding a small root of a univariate modular equation"; Proceedings of Eurocrypt'96, Springer-Verlag, pp.155-165.
3. Nicolas Courtois "The security of HFE", to be published.
4. Nicolas Courtois: The HFE cryptosystem web page.
   See http://www.univ-tln.fr/~courtois/hfe.html
5. Jean-Charles Faugère: "A new efficient algorithm for computing Gröbner bases (F_4)." Journal of Pure and Applied Algebra 139 (1999) pp. 61-88.
   See www.elsevier.com/locate/jpaa
6. Jean-Charles Faugère: "Computing Gröbner basis without reduction to 0", technical report LIP6, in preparation, source: private communication.
7. Rudolf Lidl, Harald Niederreiter: "Finite Fields"; Encyclopedia of Mathematics and its applications, Volume 20, Cambridge University Press.
8. Aviad Kipnis, Jacques Patarin, Louis Goubin: "Unbalanced Oil and Vinegar Signature Schemes" ; Eurocrypt 1999, Springer-Verlag, pp. 216-222.
9. Aviad Kipnis, Adi Shamir: "Cryptanalysis of the HFE Public Key Cryptosystem"; Proceedings of Crypto'99, Springer-Verlag.
10. Neal Koblitz: "Algebraic aspects of cryptography"; Springer-Verlag, ACM3, 1998, Chapter 4 "Hidden Monomial Cryptosystems", pp. 80-102.
11. Jacques Patarin: "Hidden Field Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms"; Eurocrypt'96, Springer Verlag, pp. 33-48. An extended up-to-date version can be found at http://www.univ-tln.fr/~courtois/hfe.ps

Cryptanalysis of Patarin's 2-Round Public Key System with S Boxes (2R)

Eli Biham

Computer Science Department
Technion – Israel Institute of Technology
Haifa 32000, Israel
Email: biham@cs.technion.ac.il
WWW: http://www.cs.technion.ac.il/~biham/

Abstract. In a series of papers Patarin proposes new efficient public key systems. A very interesting proposal, called *2-Round Public Key System with S Boxes*, or *2R*, is based on the difficulty of decomposing the structure of several rounds of unknown linear transformations and S boxes. This difficulty is due to the difficulty of decomposing compositions of multivariate binary functions. In this paper we present a novel attack which breaks the 64-bit block variant with complexity about 2^{30} steps, and the more secure 128-bit blocks variant with complexity about 2^{60} steps. It is interesting to note that this cryptanalysis uses only the ciphertexts of selected plaintexts, and does not analyze the details of the supplied encryption code.

1 Introduction

The search for efficient public key cryptosystems is as old as the idea of public key cryptosystems itself [1]. Many of the most efficient proposed schemes were based on multivariate polynomials [3, 9, 2], but they were usually broken later [8, 10, 4]. In a series of papers Patarin proposes new secure and efficient public key systems [5, 6] based on hiding the structure of polynomials in a difficult-to-analyze encryption code, and analyzes other similar schemes [4]. One of his more promising scheme is the very efficient *2-Round Public Key System with S Boxes* (shortly called *2R*) [7]. The design of this scheme is unique as it uses techniques from symmetric ciphers in designing a public key cryptosystem, while still claiming security based on relation to the difficulty of decomposing compositions of multivariate binary functions.

Patarin's public key cryptosystem with S boxes encrypts by performing the following secret operations on 64-bit or 128-bit plaintexts:

1. Invertible linear transformation L_0
2. First layer of 8 (or 16) 8x8-bit quadratic S boxes $S_{1,0}, \ldots S_{1,7}$, collectively denoted by S_1
3. Another linear transformation L_1
4. Second layer of S boxes $S_{2,0}, \ldots S_{2,7}$, collectively denoted by S_2

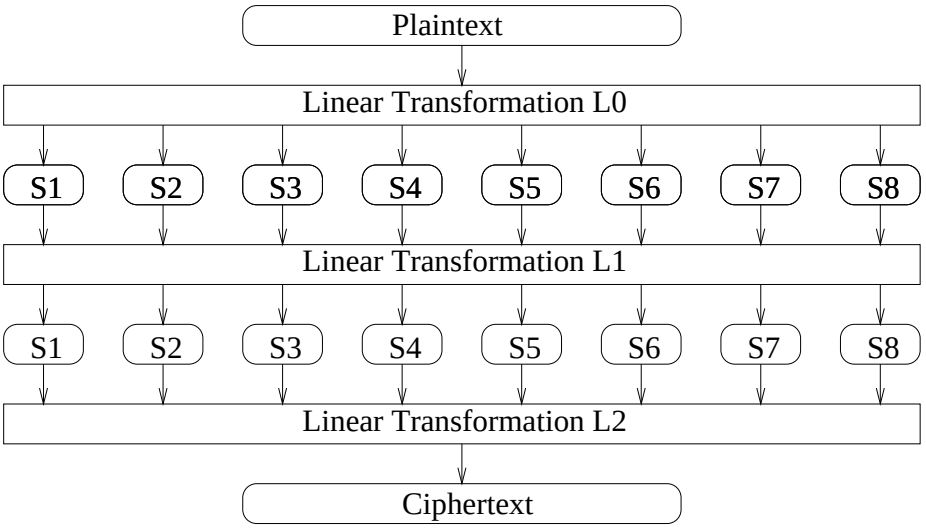


Fig. 1. Outline of Patarin's 2R scheme

5. Final linear transformation L_2
6. The ciphertext is $C = E(P) = L_2(S_2(L_1(S_1(L_0(P)))))$

Figure 1 outlines this scheme.

Only the owner of the system knows these transformations, and uses this knowledge for decryption. The publication of the system hides the structure by giving an equivalent description from which it is claimed to be very difficult to identify the original description, due to the difficulty of decomposing compositions of binary functions.

In the rest of this paper we assume that the encryption function is given as an oracle (black box). Our analysis does not study the supplied code of the system, and does not try to decompose the binary function. On the contrary, it observes the full details of the function given only the ciphertexts of (many) selected plaintexts, which are encrypted using the supplied encryption function. Moreover, it does not rely on the quadraticness of the S boxes.

A major observation is that the S boxes are not bijective. The designer claims that if the S boxes were bijective, the security of the system might had been compromised. Therefore, decryption is not unique, and some redundancy should be added to the plaintext to allow unique decryption[1].

In this paper we present the first cryptanalysis of this scheme, for which we received the prize promised by the designer for the first person to break this scheme. Later, Ye, Lam and Dai devised a different attack based on the algebraic structure of the scheme [11]. Their attack is less efficient than ours, although it

[1] Although an attacker may decide not to add this redundancy when he generates the ciphertexts used for the attack.

imposes more restrictions than we do (e.g., their attack would not work if the S boxes were not quadratic, while our attack is not very sensitive to the method used to construct the S boxes).

Our attack can break the 2R scheme with complexity about 2^{30} when the block size is 64 bits. Moreover, it can also break the more secure variant with 128-bit blocks with complexity about 2^{60}.

This paper is organized as follows: In Section 2 we describe our main observations and tools used later in the analysis, and in Section 3 we describe the attack. Section 4 discusses possible improvements of the scheme. The paper is summarized in Section 5.

2 The Main Observations and Tools

Our main observation is that the S boxes are not bijective, and thus outputs of different inputs to the same S box may collide. Given a pair of plaintexts, such S boxes which have different inputs in both encryptions but have the same outputs will be called *active* S boxes. Therefore, there exist many pairs P, P^* of plaintexts for which the outputs of all the S boxes in the first round collide, which cause the ciphertexts to collide as well. Such collisions can be constructed as follows: For any S box S_i there exist pairs of 8-bit values Y and Y^* such that $S_{1,i}(Y) = S_{1,i}(Y^*)$. Let X be a 64-bit value whose bits $8i, \ldots, 8i+7$ equal Y, and let X^* be equal to X, except for these bits, whose value is replaced by Y^*. Let $P = L_0^{-1}(X)$ and $P^* = L_0^{-1}(X^*)$. Then, the ciphertexts $E(P)$ and $E(P^*)$ are equal. Figure 2 outlines the differences in pairs of encryptions.

The attacker does not know the details of the linear transformation and the S boxes, and cannot construct collisions using this algorithm. However, if the attacker finds a collisions, he might be interested to know if the collision already occurs after the first level of S boxes, and if there is only one active S box. He can also be interested to know if there is a common active S box in two different colliding pairs. In the next subsections we present two algorithms which will be useful for this purpose.

2.1 Algorithm A

We observe that given a pair P, P^* such that $E(P) = E(P^*)$, we can identify whether there is only one active S box in the first round (and none in the second round), or there are two or more active S boxes. This identification can be performed by the following algorithm:

1. Given P and P^*
2. Repeat about 1000 times:
   (a) Select Q at random
   (b) Compute $Q^* = Q \oplus P \oplus P^*$
   (c) Check whether $E(Q) = E(Q^*)$
3. Let q be the fraction of times with equality $E(Q) = E(Q^*)$ in the previous step

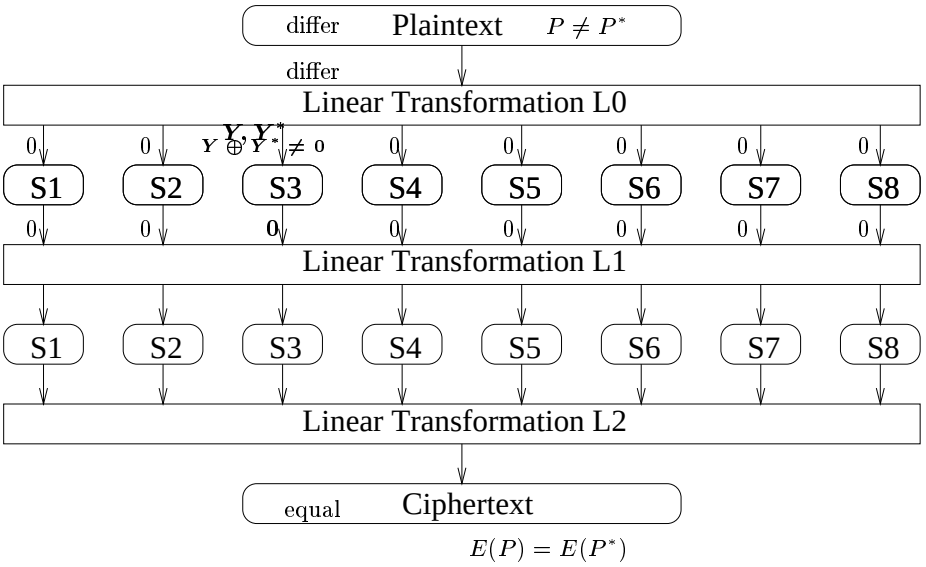


Fig. 2. Outline of the main observation

4. If $q > 1/256$, output *one-active-S-box*, and conclude that there is only one active S box in the first round.
5. Otherwise output *several-active-S-boxes*.

This algorithm works since the differences in the inputs of the S boxes in the first round are the same for P and P^* and for Q and Q^* due to linearity. Thus,

1. If there is a difference only in one S box for P and P^*, so is for Q and Q^*. Since Q is chosen randomly, and since there are 128 pairs of inputs to an S box with the particular difference, there is a probability of $1/128$ that the pair of inputs of the active S box are the same in both pairs, and thus a fraction of at least $1/128$ of the pairs of outputs collide in Q and Q^*. The repetition 1000 times ensures that the probability of a wrong output is very small.
2. If there are differences in the inputs of two or more S boxes in the first round, the expected q reduces to about 2^{-7m} or a small factor of it, where m is the number of S boxes with differing inputs.
3. If the outputs of the first layer of S boxes of P and P^* do not collide, but the ciphertexts do collide, the expected q is negligible (typically about 2^{-60} or less for 64-bit blocks).

2.2 Algorithm B

We can also identify if two pairs P_1, P_1^* and P_2, P_2^* which satisfy $E(P_1) = E(P_1^*)$ and $E(P_2) = E(P_2^*)$ (and collide already after the first round) have distinct active S boxes in the first round, or whether there is some common active S box:

1. Given P_1, P_1^*, P_2, P_2^*
2. Compute $Q_1 = P_1 \oplus P_2 \oplus P_2^*$ and $Q_1^* = P_1^* \oplus P_2 \oplus P_2^*$
3. Compute $Q_2 = P_2 \oplus P_1 \oplus P_1^*$ and $Q_2^* = P_2^* \oplus P_1 \oplus P_1^*$
4. If $E(Q_1) \neq E(Q_1^*)$ or $E(Q_2) \neq E(Q_2^*)$ output *common*, and conclude that there exists a common active S box.
5. If $E(Q_1) = E(Q_1^*)$ and $E(Q_2) = E(Q_2^*)$ output *distinct* and conclude that the pairs have distinct active S boxes with a high probability.

This algorithm should detect almost all cases of common active S boxes. If it does not detect as such, there is a high probability that there are no common active S boxes.

This algorithm works, as the pairs Q_1, Q_1^* (and Q_2, Q_2^*) differ exactly in the same S boxes as of P_1, P_1^* (P_2, P_2^*, respectively). If there are no common active S boxes, the active S boxes in Q_1, Q_1^* (Q_2, Q_2^*) have exactly the same inputs as in P_1, P_1^* (P_2, P_2^*), and thus the same collision occurs. If there is some common active S box, the inputs in Q_1, Q_1^* (Q_2, Q_2^*) are different than in P_1, P_1^* (P_2, P_2^*), and thus the probability of collision is small.

3 The Attack

3.1 Analyzing the First Linear Transformation

The first step of the attack computes the ciphertexts of many random plaintexts, and collects pairs P, P^* whose ciphertexts $E(P)$, $E(P^*)$ are equal, and for which Algorithm A outputs one-active-S-box.

Given such pairs we use algorithm B to divide them to eight sets, sorted by the active S box. We can use this result to find a basis for the differences $P \oplus P^*$ of each set. The combination of all these bases form a basis for the plaintext space, which relates to the inputs of S boxes of the first round. In total we get eight sets, each consists of eight basis vectors. Each set of basis vectors affect a different S box in the first round. We cannot identify the order of the S boxes nor any transformation on the inputs of individual S boxes. Therefore, this basis is all the information we can get on the linear transformation L_0. Without loss of generality, all the rest of the definition of L_0 can be viewed as part of the S boxes and their order.

This step of the attack is the most complex: 2^{37} random plaintexts are encrypted by the attacker in order to find about 1000 random collisions. However, a careful analysis shows that due to the structure of the cipher, the number of collisions will be about 256 times higher, and about 4000 of these collisions will have only one active S box. Then, the application of Algorithm A to identify the pairs with one active S box requires about $256 \cdot 1000 \cdot 2000 = 2^{29}$ encryptions.

The application of Algorithm B to recover the basis takes a few thousands additional encryptions. When applied efficiently, considering all the information we get during the analysis (such as a partial basis) and improvements to the attack (such as analysis of pairs with two or three active S boxes in the first round), the total complexity of this step can be reduced to below 2^{30}.

3.2 Analyzing the Last Linear Transformation

In the next step of the attack we find a basis for the ciphertexts, related to the outputs of the eight S boxes in the second round. We observe that inputs to the first layer of S boxes which differ only in the inputs to one S box cause a difference in the output of this S box only. In turn, the output of L_1 has difference in the inputs of most S boxes in the second round. In some cases however such differences do not affect about one or two S boxes in the second round. Although to overcome this weakness, designers might design linear transformations as multipermutations from the outputs of the S boxes in one round to the inputs of the S boxes in the next round, a similar property may occur when differences in pairs of S boxes in the first round lead to zero differences in the inputs of one (or a few) S boxes in the second round.

Using the basis we got for the plaintexts, we can now control the inputs of selected S boxes in the first round. In particular, we can generate structures of many plaintexts with exactly the same value in the inputs to one (or two) S boxes in the first round, but with random values in the inputs to the rest of the S boxes. We can even generate pairs of S boxes in which 1) in all pairs one member has some fixed (but unknown) input F_1 to one or two S boxes and the other member has some other fixed (but unknown) value F_2, 2) these fixed values F_1, F_2 are fixed in all the pairs, 3) in each pair random values are selected for the inputs of the rest of the S boxes, and both members of the pair have the same values in all the rest of the S boxes. As an example for such pairs, the inputs of the S boxes in a pair can be $(F_1, R_1), (F_2, R_1)$, where in another pair it is $(F_1, R_2), (F_2, R_2)$, and in a third pair it is $(F_1, R_3), (F_2, R_3)$, etc. Note that the input differences to the second layer of S boxes are fixed in all the pairs of a structure, and depend only on $F_1 \oplus F_2$.

In this step we generate many such structures, compute the ciphertext differences, and compute a basis for the space spanned by the ciphertext differences. If there are differences in the inputs of all the S boxes in the second round, it is expected that the space spanned by the ciphertext differences is of dimension 64. If there is one S box with zero difference the spanned space is expected to be with dimension 56 (and cannot be larger than 56). In general, it is expected that non-zero differences in the inputs to m S boxes in the second round lead to dimension $8m$ of the spanned space. It is also expected that in such case structures of about 100 pairs (200 encrypted plaintexts) suffice to span all the space, and additional pairs rarely enlarge the spanned space. It is also expected that about one of every $256/8 = 32$ structures lead to space of dimension less than 64. In order to divide the space to the partial spaces inherited by the outputs of the S boxes, we need about 7–10 such structures, and thus we need in total

about $10 \cdot 32 \cdot 200 = 64000$ encrypted plaintexts. At the end of this step we know all the possible information on L_2, except for the information that can be viewed as part of the S boxes.

3.3 Analyzing the Intermediate Layers

Up to this point we showed how to remove the first and last linear transformations. Using this information we can now find how many (effective) output bits of each S box in the first round affect each one of the inputs of the S boxes in the second round. This step is performed by encrypting 256 plaintexts for each S box in the first round, which contain all the possible inputs to the S box, and in which the inputs to all the other S boxes are fixed. We count the number of different outputs of each S box in the second round. If the count is 2^m (or slightly smaller) for some m, there is a high probability that the rank of the linear transformation from the output of the S box in the first round to the input of (only) the S box in the second round is m.

From the same data we can even find information on the values of the S boxes, as by looking at the outputs of the S boxes in the second round we can group inputs to the S box in the first round by the values of the m bits of its output that affect the S box in the second round. By correlating this information among several S boxes in the second round we can complete most information on the S boxes in the first round, including their values, and the actual bits that are transfered by L_1 to each S box. The only values that we cannot identify are the affine constants, which can be viewed as part of the second S box layer instead.[2] It is now easy to complete the second layer of S boxes, as all the rest of the cipher is already recovered.

4 Discussion

A possible improvement is using bijective S boxes in the first layer of S boxes. This modification ensures that the first step of the attack is not applicable, although other attacks might become practical. This modification can be combined with an increased number of rounds using non-bijective S boxes to protect against other kinds of attacks that require at least two rounds of non-bijective S boxes.

If the first layer of S boxes remain non-bijective, the first step of the attack can still find the first linear transformation regardless of the number of rounds and the design of the other rounds. Therefore, such a variant of this scheme may starts with the layer of S boxes without a preceding linear transformation without affecting its security. In such a case we would propose using at least three layers of S boxes, each followed by a linear transformation to ensure that

[2] They can actually be eliminated, as the scheme is equivalent to a scheme in which the zero inputs to S boxes always have zero outputs, and the plaintext zero (or any other selected plaintext) is encrypted using only zero inputs of S boxes.

the rest of the attack is not applicable. However, adding layers come with an unacceptable penalty in the size of the public key and encryption speed.

A promising improvement seems to discard the redundant input variables from the published equations, replacing them by their equivalent formulae in terms of other variables. In such a way Algorithm A might never output one-active-S-box in the first step of the attack. If in addition some of the 64 equations are also discarded, it seems that all the known attacks will not work.

Remark: The authors of [11] propose that the S boxes should not be kept secret. However, if the S boxes are not secret, it would simplify the recovery of the linear transformations in our attack, giving more information to the attacker. We highly recommend to keep the S boxes secret, just as they should be in the original 2R scheme.

5 Summary

In this paper we proposed a practical attack, which is the first attack against Patarin's 2-Round Public Key System with S Boxes. For a blocksize of 64 bits, the complexity of the attack is about 2^{30} encryptions (that the attacker can compute on his machine as this is a public key scheme). The more secure variant with 128-bit blocks can be analyzed with complexity about 2^{60}. Efficient implementations of the attack might even have marginally smaller complexities than these.

Acknowledgments

We are grateful to Jacques Patarin for his great help during our work on this attack, and for the invitation for an excellent restaurant in Paris (the promised prize for breaking this system).

References

1. W. Diffie, M. E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. 22, No. 6, Nov. 1976.
2. Tsutomu Matsumoto, Hideki Imai, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'88, pp. 419-453, 1988.
3. H. Ong, C. P. Schnorr, Adi Shamir, *Efficient Signature Schemes Based on Polynomial Equations*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'84, pp. 37–46, 1984.
4. Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'95, pp. 248–261, 1995.

5. Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'96, pp. 33–48, 1996.
6. Jacques Patarin, *Asymmetric Cryptography with a Hidden Monomial*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'96, pp. 45–60, 1996.
7. Jacques Patarin, Louis Goubin, *Asymmetric Cryptography with S Boxes*, proceedings of ICICS'97, Lecture Notes in Computer Science 1334, pp. 369–380, 1997.
8. John M. Pollard, Claus P. Schnorr, *An Efficient Solution to the Congruence* $x^2 + ky^2 = m (\mathrm{mod}\, n)$, IEEE Transactions on Information Theory, Vol. 33, No. 5, 1987.
9. Adi Shamir, *Efficient Signature Schemes Based on Birational Permutations*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'93, pp. 1–12, 1993.
10. Serge Vaudenay, Jacques Stern, Don Coppersmith, *Attacks on the Birational Permutation Signature Schemes*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'93, pp. 435–443, 1993.
11. Ding-Feng Ye, Kwok-Yan Lam, Zong-Duo Dai, *Cryptanalysis of "2R" Schemes*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'99, pp. 315–325, 1999.

Colossus and the German Lorenz Cipher – Code Breaking in WW II

Anthony E Sale, Hon. FBCS

Bletchley Park Trust
tsale@qufaro.demon.co.uk

The Lorenz cipher system was used by the German Army High Command in World War II. It used a binary additive method for enciphering teleprinter signals.

The Lorenz machine used 12 wheels each with a mutually prime number of small cams round its periphery, 501 in all. The wheels were geared together to ensure a very long repetition period. The task facing the code breaker was to find the patterns of cams round each wheel and the relative start positions to which the operator had turned the wheels before sending his message.

The cryptographic structure of the Lorenz machine was given away by a catastrophic mistake made by a German operator on 30th August 1941.

A special section was set up in Bletchley Park, the Allies code breaking establishment, to attack this cipher, codename "Fish". Laborious hand methods were worked out which showed that it was possible but only with 4 to 6 weeks delay for deciphering each message.

Professor Max Newman had ideas for automating and speeding up the breaking. In March 1943 he approached Dr Tommy Flowers who started designing and building Colossus to meet Max Newman's requirements for a machine to break Lorenz more quickly. Colossus was working by December 1943 and installed in Bletchley Park over Christmas 1943. It was working by January 1944 and successful in its first trial on a real cipher message. It reduced the time to break Lorenz from weeks to hours providing vital intelligence just in time for D Day, the invasion of Europe on 6th June 1944.

After D Day 10 machines were built and working in Bletchley Park. Then at the end of the War eight machines were totally dismantled, two went to GCHQ at Cheltenham. These were destroyed in 1960 together with all the drawings of Colossus and its very existence was kept secret until the mid 1970's.

In 1991 Tony Sale and two colleagues started the campaign to save Bletchley Park from property developers. At this time he was restoring some early computers at the Science Museum in London. He thought it might be possible to rebuild Colossus and started gathering information. Eight wartime photographs and some fragments of circuit diagrams were recovered. He decided to have a go and had the basic Colossus working by 6th June 1996.

Now four years further on Colossus is nearly completed and demonstrates the power of what is now recognised as the world's first electronic programmable digital computer.

Efficient Concurrent Zero-Knowledge in the Auxiliary String Model

Ivan Damgård

Aarhus University, BRICS*
ivan@daimi.au.dk
Dept. of Computer Sience
Ny Munkegade
DK 8000 Aarhus C, Denmark

Abstract. We show that if any one-way function exists, then 3-round concurrent zero-knowledge arguments for all NP problems can be built in a model where a short auxiliary string with a prescribed distribution is available to the players. We also show that a wide range of known efficient proofs of knowledge using specialized assumptions can be modified to work in this model with no essential loss of efficiency. We argue that the assumptions of the model will be satisfied in many practical scenarios where public key cryptography is used, in particular our construction works given any secure public key infrastructure. Finally, we point out that in a model with preprocessing (and no auxiliary string) proposed earlier, concurrent zero-knowledge for NP can be based on any one-way function.

1 Introduction

In a zero-knowledge protocol [23], a prover convinces a verifier that some statement is true, while the verifier learns nothing except the validity of the assertion. Apart from being interesting as theoretical objects, it is well-known that zero-knowledge protocols are extremely useful tools for practical problems. For instance as stand-alone for identification schemes[1], but probably a more important application is as subprotocols in schemes for more complex tasks such as voting, electronic cash and distributed key generation.

Hence the applicability of the theory of zero-knowledge in real life is of extreme importance. One important aspect of this is composition of protocols, and the extent to which such composition preserves zero-knowledge. While sequential composition does preserve zero-knowledge, this is not always the case for parallel composition [22].

* Basic Research in Computer Science, Center of the Danish National Research Foundation

[1] However, the identification problem can also be solved without using zero-knowledge [2]

In [12] Dwork, Naor and Sahai pointed out that the strict synchronization usually assumed when composing zero-knowledge protocols is unrealistic in scenarios such as Internet based communication. Here, many instances of the same or different protocols may start at different times and may run with no fixed timing of messages. What is needed here is a stronger property known as *concurrent zero-knowledge*, i.e., even an arbitrary interleaving of several instances of zero-knowledge protocols is again zero-knowledge, even when the verifiers are all controlled by a single adversary, who may use information obtained from one protocol to determine its behavior in another instance.

Unfortunately, standard constructions for zero-knowledge protocols fail to provide this property. This is because they are based on simulation by rewinding the verifier. In a concurrent setting, the simulator may be forced to rewind an exponential number of times. In fact, it seems that concurrent zero-knowledge cannot be provided at all in the usual model with as few rounds as ordinary zero-knowledge. Kilian, Petrank and Rackoff [19] show that only BPP languages have concurrent zero-knowledge proofs or arguments with 4 rounds or less, if black-box simulation is assumed[2].

Thus, a lot of research has gone into finding ways of getting around this problem. In [12], it was shown that given constraints on the timing of messages[3], concurrent zero-knowledge can be achieved for all of NP in a constant number of rounds. Subsequently it was shown that the need for timing constraints could be pushed into a preprocessing phase[13]. In [10] it was shown that the timing constraints in the at all preprocessings are finished before the main proofs start. This comes at the price that the work needed in the preprocessing depends on the size and number of statements to be proved later. Finally, Richardson and Kilian [26] show that it is possible to do without timing constraints, at the expense of a non-constant number of rounds.

We note that a completely different approach is possible: one could go for a weaker property than zero-knowledge, one that would be preserved in a concurrent setting. One such possibility is the Witness-Hiding (WH) protocols of Feige and Shamir [15]. Most WH protocols are based on the standard paradigm of the prover proving knowledge of one of two "computationally independent" witnesses without revealing which one he knows. Such protocols are also WH when used concurrently, and can be used to construct secure identification systems. In [8], very efficient methods for building such protocols are developed. However, for more general use, e.g., as subrutines in multiparty computation or verifiable secret sharing protocols, WH is not always sufficient, one needs simulatability to prove the overall protocol secure.

[2] Virtually all known zero-knowledge protocols are black-box simulatable

[3] These constraints are much milder than strict synchronization, please refer to [12] for details

2 Our Work

Our main objective is to show that concurrent zero-knowledge can sometimes be obtained in a simple way using standard tools. We do not claim any major new techniques, in fact our solution is quite straightforward. Nevertheless, we believe it is useful to realize that in many real life scenarios, resources are available that allow achieving concurrent zero-knowledge easily.

We do not mean to suggest that our solution is always more practical than previous methods for achieving concurrent zero-knowledge in constant round, such as the timing based one from [12]. In fact the solutions are based on assumptions of very different nature. Which solution is preferable will often depend on the actual scenario in which the you want the solution to work.

Also, nothing we say here makes the theoretical work on the subject less interesting or important – a major problem, namely whether concurrent zero-knowledge for NP can be achieved in constant round *without* extra assumptions, remains open.

Independently, Kilian and Petrank [18] and Canetti, Goldreich, Goldwasser and Micali [9] have made observations similar to ours, in the case of [9] as a result of introducing a new general concept called Resettable Zero-Knowledge.

2.1 The Model

Our work starts from the following assumption: an auxiliary string with a prescribed distribution is available to the prover and verifier. Given this assumption we will see that concurrent zero-knowledge can be achieved in constant round with no timing constraints or preprocessing. Informally, zero-knowledge in such a setting means as usual that the verifiers entire view can be simulated efficiently, which here means its view of the interaction with the prover, as well as the auxiliary string. Soundness means that no polynomial time prover can cheat the verifier with non-negligible probability where the probability is taken over the choice of the auxiliary string as well as the coin tosses of the players.

More formally, an interactive argument for a language L in this model consists of a probabilistic polynomial time algorithm G, and polynomial time interactive Turing Machines P, V. The algorithm G gets as as input 1^k and outputs an auxiliary string σ. P, V then get σ and a word x of length k as input, and P gets a private input w. At the end of the interaction, V halts and outputs accept or reject. When we talk about the probability of acceptance or rejection in the following, these probabilities are taken over the coin tosses of G, P and V. Note that even when we consider cheating provers, we still assume that σ is correctly generated (by G).

As usual, a negligible function from natural to real numbers $\delta()$ is a function such that $\delta(k) \leq 1/p(k)$ for all polynomials $p(\cdot)$ and all large enough k)

Definition 1. *We say that (G, P, V) is an interactive argument in the auxiliary string model for language L, if*

- *For every $x \in L$, there is a w such that if P gets w as private input, V will accept on input x.*
- *For words $x \notin L$, and for every probabilistic polynomial time prover, the probability that V accepts input x is negligible in $|x|$.*

We have defined here for simplicity only the case where an auxiliary string is only used to prove a single statement, and where the input parameter to G is set equal to the length of the common input x. None of these restrictions are essential, and they can be ignored in a practical application.

To define zero-knowledge, we consider as usual an arbitrary probabilistic polynomial time verifier V^* that gets private auxiliary input y of length polynomial in $|x|$. We then have:

Definition 2. *For any verifier V^*, there exists a simulator M_{V^*}, such that for words $x \in L$ and arbitrary auxiliary inputs y, such that M_{V^*} runs in expected polynomial time, and the distribution of $M_{V^*}(x, y)$ is polynomially indistinguishable from the view of V^* produced from the same input (namely σ, the random coins of V^*, and the conversation with P).*

Note that the standard non-interactive zero-knowledge model (where the auxiliary string is a uniformly chosen random string) [3] is a special case, and indeed by their very nature non-interactive zero-knowledge proofs do not require rewinding to simulate, and so are robust in a concurrent setting. It is even possible to do any polynomial number of non-interactive proofs based on the same globally shared random string [14].

However, there are still several reasons why non-interactive zero-knowledge proofs are not the answer to all our problems: they are in general much less efficient than interactive ones and - as far as we know - require stronger cryptographic assumptions (trapdoor one-way permutations as opposed to arbitrary one-way functions). We would like a solution allowing us to use standard efficient constructions of protocols securely in a concurrent setting, without significant loss of efficiency.

We also need to consider proofs of knowledge in our model. For this, we use a straightforward adaptation of the definition of Bellare and Goldreich, in the version modified for computationally convincing proofs of knowledge[4]. The scenario, consisting of G, P, V is the same as before. Now, however, the language L is replaced by a binary relation R, and the prover's claim for the given common input x is that he knows w such that $(x, w) \in R$.

Definition 3. *We say that (G, P, V) is a proof of knowledge for R in the auxiliary string model, with knowledge error $\kappa()$, if*

- *If P is given w, such that $(x, w) \in R$, then V accepts.*
- *There exists an* extractor*, a machine with oracle access to the prover which on input x outputs w with $(x, w) \in R$. This extractor must satisfy the following for any probabilistic polynomial time prover and any long enough x: if the provers probability $\epsilon(x)$ of making the verifier accept is larger than $\kappa(x)$, then the extractor runs in expected time $p(|x|)/(\epsilon(x) - \kappa(x))$.*

The model we use (with a general auxiliary string) was also used in [5] (for a different purpose). The rationale for allowing a general distribution of the reference string is of course that one may hope that this allows for more efficient protocols, for examplea much shorter auxiliary string. The problem, on the other hand, may be that requiring a more powerful resource makes the model less realistic.

However, as we shall see, our protocols do in fact apply to a realistic situation, namely a public-key cryptography setting where users have public/private key pairs. In fact our prover and verifier do not need to have key pairs themselves, nevertheless, they will be able to prove and verify general NP statements in concurrent zero-knowledge by using the public key P_A of a third party A as auxiliary string. This will work, provided that

- The verifier believes that A's secret key is not known to the prover.
- The prover believes that P_A was generated using the proper key generation algorithm for the public-key system in use.

We stress that A does not need to take part in the protocols at all, nor does he need to be aware that his public key is being used this way, in particular keys for standard public key systems like RSA, El Gamal or DSS can be used directly.

Note that if we have a secure public key infrastructure where public keys are being certified by a certification authority (CA), then all our demands are already automatically satisfied because the CA can serve as player A in the above: in order for the infrastructure to be secure in the first place, each user needs to have an authentic copy of the CA's public key available, and one must of course trust that the CA generated its public key in the proper way and does not reveal its private key to anyone else.

So although our model does make stronger assumptions on the environment than the standard one, we believe that this can be reasonable: The problem of concurrent zero-knowledge arises from the need to apply zero-knowledge protocols in real situations. But then solutions to this problem should be also allowed to take advantage of resources that may exist in such scenarios.

It is important to realize one way in which our model can behave differently from the standard one: suppose a verifier shows to a third party a transcript of his interaction with the prover as evidence that the protocol really took place. Then, in our model, there are scenarios where this will be convincing to the third party (contrary to what is case with the standard model). This may in some applications be a problem because it can harm the privacy of users. We stress, however, that in the case where a public-key infrastructure exists, there are ways around this problem. We discuss this issue in more detail in Section 4.

2.2 The Results

Our first result is a construction for protocols of a particular form. Assume we have a binary relation R, and a 3-move proof of knowledge for R, where the verifier sends a random challenge as the second message. Thus this protocol gets

a string x as common input for prover and verifier, whereas the prover gets as private input a witness for x, i.e. w such that $(x, w) \in R$. Conversations in the protocol are of form (a, e, z), where the prover chooses a, z. We will assume that the protocol is honest verifier zero-knowledge in the sense that given e, one can efficiently compute a correctly distributed conversation where e is the challenge. Finally we assume that a cheating prover can answer only one of the possible challenges, or more precisely, from the common input x and any pair of accepting conversations $(a, e, z), (a, e', z')$ where $e \neq e'$, one can compute a witness for x. We call this a Σ-protocol in the following. We have

Theorem 1. *Given any binary relation R and a Σ-protocol for R. If one-way functions exist, then there exists a computationally convincing and concurrent zero-knowledge 3-move proof of knowledge (with negligible knowledge error and no timing constraints) for R in the auxiliary string model.*

The construction behind this result can be applied in practice to the well known proofs of knowledge of Schnorr [21] and Guillou-Quisquater [16] to yield concurrent zero-knowledge proofs of knowledge in the auxiliary string model with negligible loss of efficiency compared to the original protocols (which were not even zero-knowledge in the usual sense!). The idea behind this result also immediately gives:

Theorem 2. *If one-way functions exist, there exist 3-move concurrent zero-knowledge interactive arguments in the auxiliary string model (with no timing constraints) for any NP problem.*

In both these results, the length of the auxiliary string is essentially the size of the computational problem the prover must solve in order to cheat. The length does not depend on the size or the number of statements proved.

Our final result is an observation concerning the preprocessing model of Dwork and Sahai [13] (where there is no auxiliary string). It was shown in [13] that prover and verifier can do a once-and-for-all preprocessing (where timing constraints are applied), and then do any number of interactive arguments for any NP problem in concurrent zero-knowledge (with no timing constraints) in 4 rounds. This was shown under the assumption that one-way trapdoor permutations exist. Below, we observe the following:

Theorem 3. *If any one-way functions exists, then any NP problem has a 3-round concurrent zero-knowledge argument in the preprocessing model of Dwork and Sahai.*

We note that our preprocessing is once-and-for-all, like the one in [13]: once the preprocessing is done, the prover and verifier can execute any polynomial number of proofs securely, and the complexity of the preprocessing does not depend on the number or size of the statements proved.

3 The Protocols

3.1 Trapdoor Commitments Schemes

In a commitment scheme, a committer C can commit himself to a secret s chosen from some finite set by sending a *commitment* to a reciever R. The receiver should be unable to find s from the commitment, yet C chould be able to later open the commitment and convince R about the original choice of s.

A *trapdoor commitment scheme* is a special case that can be loosely described as follows: first a public key pk is chosen based on a security parameter value k, usually by R by running a probabilistic polynomial time *generator* G. Then pk is sent to C. There is a fixed function *commit* that C can use to compute a commitment c to s by choosing some random input r, and setting $c = commit(s, r, pk)$. Opening takes place by revealing s, r to R, who can then check that $commit(r, s, pk)$ is the value he received originally.

We then require the following:

Hiding: For a pk correctly generated by G, uniform r, r' and any s, s', the distributions of $commit(s, r, pk)$ and $commit(s', r', pk)$ are polynomially indistinguishable (as defined in [23]).

Binding: There is a negligible function $\delta()$ such that for any C running in expected polynomial time (in k) the probability that C on input pk computes s, r, s', r' such that $commit(s, r, pk) = commit(s', r', pk)$ and $s \neq s'$ is at most $\delta(k)$.

Trapdoor Property: The algorithm for generating pk also outputs a string t, the trapdoor. There is an efficient algorithm which on input t, pk outputs a commitment c, and then on input any s produces r such that $c = commit(s, r, pk)$. The distribution of c is poynomially indistinguishable from that of commitments computed in the usual way.

In other words, the commitment scheme is binding if you know only pk, but given the trapdoor, you can cheat arbitrarily.

From the results in Shamir et al.[20], it follows that existence of any one-way function f implies the existence of a trapdoor commitment scheme, where the public key is simply $f(y)$, where y is chosen uniformly in the input domain of f, and y is the trapdoor. Based on standard intractability assumptions such as hardness of discrete log or RSA root extraction, very efficient trapdoor commitment schemes can be built, see e.g. [6].

3.2 A construction for Σ-protocols

In what follows, we will assume that we have a relation R and a Σ-protocol $\mathcal{P}$ for R. The prover and verifier get as common input x, while the prover gets as private input w, such that $(x, w) \in R$.

We will be in the auxiliary string model, where the auxiliary string will be the public key pk of a trapdoor commitment scheme, generated from security parameter value $k = |x|$. For simplicity, we assume that the commitment scheme

allows to commit in one commitment to any string a, that may occur as the first message in $\mathcal{P}$ (in case of a bit commitment scheme, we could just commit bit by bit). Finally, note that since the properties of a Σ-protocol are preserved under parallel composition, we may assume without loss of generality that the length of a challenge e in the protocol is at least k.

The protocol then proceeds as follows:

1. On input x, w, the prover computes a using the prover's algorithm from $\mathcal{P}$, chooses r at random and sends $c = commit(a, r, pk)$ to the verifier.
2. The verifier chooses e at random and sends it to the prover.
3. The prover computes z, the answer to challenge e in $\mathcal{P}$ and sends z, a, r to the verifier.
4. The verifier accepts iff it would have accepted on x, a, e, z in $\mathcal{P}$, and if $c = commit(a, r, pk)$.

It is straightforward to show that this protocol has the desired properties. First, a simulator for the protocol given an arbitrary verifier V^*:

1. Generate pk with known trapdoor t and give x, pk to V^*.
2. Send a commitment c computed according to the trapdoor property to V^* and get e back.
3. Run the honest verifier simulator on input e to get an accepting conversation (a, e, z) in the original protocol. Use the trapdoor to compute r such that $c = commit(a, r, pk)$. Send z, a, r to V^*.

This simulation works based on the hiding and trapdoor properties of the commitment scheme, and does not require rewinding of V^*, hence the protocol is also concurrent zero-knowledge.

To show it is a proof of knowledge with knowledge error $\kappa()$, we will show that the protocol satisfies the definition when we choose $\kappa(x) = 1/q(|x|)$ for *any* polynomial $q()$, thus the "true" knowledge error is smaller than any polynomial and so is negligible. This analysis is rather loose because we are dealing with a general type of intractability assumption. A much better analysis can be obtained from making a concrete assumption on a particular commitment scheme.

Our algorithm for extracting a witness will based on the following

Claim. From any prover convincing the verifier with probability $\epsilon(x) > 1/q(k)$, we can extract, using rewinding, convincing answers to two different challenges (on the same intial message) e, e', in time proportional to $1/\epsilon(x)$ for all large enough k (Recall that we have set $k = |x|$).

Intuitively, this is just because $1/q(k) > 2^{-k}$ for all large enough k, and a success probability larger than 2^{-k} must mean that you can answer more than one challenge, since the number of challenges is at least 2^k. However, the proof is a bit less obvious than it may seem: the prover may be probabilistic, but we still have to fix his random tape once we start rewinding. And there is no gurantee that the prover has success probability $\epsilon(x)$ for all choices of random tapes, indeed $\epsilon(x)$ is the average over all such choices. However, a strategy for probing the prover can be devised that circumvents this problem:

Using a line of reasoning devised by Shamir, let H a matrix with one row for each possible set of random choices by the prover, and 2^k columns index by the possible challenges (assume for simplicity that there are precisely 2^k). In the matrix we write 1 if the verifier accepts with this random choice and challenge, and 0 otherwise. Say a row is *heavy* if it contains more that $\epsilon(x)/2$ 1's. Since the total fraction of 1's in H is $\epsilon(x)$, at least $1/2$ of the 1's are located in heavy rows. By using the prover as a black-box, we can probe random entries in H, or random entries in a given row, and the goal is of course to find two 1's in the same row. Consider the following algorithm:

1. Probe random entries in H until a 1 is found.
2. Start running the following two processes in parallel. Stop when at least one of them stops.
   A. Probe random entries in the row where we already found a 1, stop when a new 1 is found.
   B. Repeatedly flip a random coin that comes out heads with probability $\epsilon(x)/c$ (where c is an appropriately chosen constant, see below), and stop as soon as heads comes out. This can be done, e.g., by probing a random entry in H, choosing a random number among $1, ..., c$, and outputting heads iff both the entry and the number was 1.
3. If process A finished first, output the position of the two 1-entries found.

This algorithm obviously runs in expected time a polynomial in k times $O(1/\epsilon(x))$.

We then look at the success probability: assume the row we find is heavy. Then the expected number of trials to find a new 1 is $T(\epsilon(x)) = 2^k/(\epsilon(x)2^{k-1}-1)$ which is $O(1/\epsilon(x))$ if $\epsilon(x) > 2^{-k+2}$; and this last condition certainly holds for all large enough k. The probability that A uses less than $2T(\epsilon(x))$ trials is at least $1/2$. By choosing c large enough, we can ensure that the probability that B uses more trials than $2T(\epsilon(x))$ is constant. Since the probability that we find a heavy row to begin with is constant $(1/2)$, the total success probability is also constant. Hence repeating this entire procedure until we have success takes expected time a polynomial in k times $O(1/\epsilon(x))$, as required. This finishes the proof of the above claim.

Once we are successful, we get commitment c, conversations (a, e, z), (a', e', z') that are accepting in the original protocol, and finally values r, r' such that $c = commit(a, r, pk) = commit(a', r', pk)$. If $a = a'$, a witness for the common input x can be computed by assumption on the original protocol. Our extractor simply repeats the whole extraction process until $a = a'$.

Since one repeat of the extraction process takes expected polynomial time, it follows from the binding condition of the commitment scheme that the case $a \neq a'$ occurs with negligible probability, at most $\delta(k)$. Hence the entire extractor takes expected time $1/(1-\delta(k))$ times the time need for one attempt. This is certainly at most $p(|x|)/(\epsilon(x) - q(|x|))$ for some polynomial $p()$.

This and the result from [20] above on existence of trapdoor commitments now implies Theorem 1. As for Theorem 2, we just need to observe that the

standard zero-knowledge interactive protocols for NP complete problems [24, 1] can in fact be based on any commitment scheme. They are usually described as sequential iterations of a basic 3-move protocol. However, in our model we will use a trapdoor commitment scheme, and do the iterations in parallel: it is then trivial that the protocols can be straight line simulated if the simulator knows the trapdoor. And soundness for a poly-time bounded prover follows by a standard rewinding argument. A more careful analysis of the error probability and the way it depends on the intractability assumption we make can be obtained using the definitions from [11].

This same idea applies easily to the preprocessing model (with no auxiliary string) of Dwork and Sahai [13]: here, the prover and verifier are allowed to do a preprocessing, where timing constraints are used in order to ensure concurrent zero-knowledge. After this, the goal is to be able to do any number of interactive arguments in concurrent zero-knowledge, without timing constraints. In [13], it is shown how to achieve this based on existence of one-way trapdoor permutations. However, an idea similar to the above will allow us to do it based on any one-way function (and a smaller number of rounds): In the preprocessing, the verifier chooses an instance of the trapdoor commitment scheme from [20] and sends the public key to the prover. The verifier then proves knowledge of the trapdoor. After this, any number of interactive arguments for NP problems can be carried out in constant round and concurrent zero-knowledge. We will use the parallel version of [24] or [1] based on the commitment scheme we established in the preprocessing. Simulation can be done by extracting the trapdoor from the verifier's proof of knowledge (here, rewinding is allowed because of the timing constraints) and then simulate the main proofs straight-line.

4 Implementation in Practice

In our arguments for practicality of our model, we claimed that the public key of a third party can be used as auxiliary string. Given the construction above, this amounts to claiming that the public key of any public-key crypto-system or signature scheme can also be used without modification as the public key of a trapdoor commitment scheme.

We can assume that the public key was generated using some known key generation algorithm (recall that we originally assumed about the third party that he generates his keys properly and does not give away the private key). Clearly, the function mapping the random bits consumed by this algorithm to the resulting public key must be one-way. Otherwise, the system could be broken by reconstructing the random input and running the algorithm to obtain the private key. Thus, from a theoretical point of view, we can always think of the public key as the image of a random input under a one-way function and apply the commitment scheme from [20].

This will not be a practical solution. But fortunately, standard public key systems used in real life allow much more efficient implementations. Any system based on discrete logarithms in a prime order group, such as DSS, many El

Gamal variants, and Cramer-Shoup has as part of the public key some group element of form g^x where x is private and g is public, and where g has prime order q . This is precisely the public key needed for the trapdoor commitment scheme of Pedersen [25], which allows commitment to a string of length $\log q$ in one commitment.

If we have an RSA public key with modulus n, we can always construct from this a public key for the RSA based trapdoor commitment scheme described in [6]. We define q to be the least prime such that $q > n$ (this can easily be computed by both prover and verifier). We then fix some number b in Z_n^*, this could be for instance be a string representing the name of the verifier. The intractability assumption for the commitment scheme then is that the prover will not be able to extract a q'th root mod n of b (such a root always exists by choice of q). Also this scheme allows comitment to $\log q$ bits in one commitment.

Note that when executing a proof of the kind we constructed, it is always enough in practice for the prover to make only one commitment: he can always hash the string the wants to commit to using a standard collision intractable hash function and commit to the hash value. This means that well known efficient protocols can be executed in this model with no significant loss of efficiency.

Finally, we discuss the issue of whether a verifier can prove to a third party that he interacted with the prover. We give an example where this is possible in our model:

Suppose a public key pk is used as auxiliary string as we have described, to do proofs of knowledge for a hard relation. And suppose the verifier V interacts with a prover and then shows a transcript of the interaction to a third party C as evidence that the protocol actually took place.

Note that V, if he wanted to create the transcript completely on his own, would have to simulate the protocol *given the fixed key* pk. Now, if V computes his challenge string for instance by applying a one-way function to the first message sent by the prover in the protocol, this simulation appears to be a hard problem, unless one knows either the private key corresponding to pk or the prover's secret. Of course, this is different from simulating the verifier's *entire view*, which includes the random choice of pk - this can indeed be done efficiently since the protocol is zero-knowledge in our model.

So in this scenario, C would have to conclude that V could only have obtained the transcript by either interacting with the prover or cooperating with the party who generated pk in the first place. And if for instance this party is a CA that C trusts then he can exclude the latter possibility.

The implications of this property depend entirely on the scenario we are in. In some cases it can be an advantage to be able to prove that a protocol really took place, in other cases such tracing would harm the privacy of users.

However, in a case where a public-key infrastructure is available, in particular when V has a public key pk_V known to P, one can change the protocol slightly making it harder for V to convince C. The idea is to redefine the way in which P commits to bits, such that a commitment to bit b will have the form $commit(pk, b_1), commit(pk_V, b_2)$, where P chooses b_1, b_2 randomly such

that $b = b_1 \oplus b_2$. This preserves binding and hence soundness because P does not know an honest V's private key. Also hiding and hence zero-knowledge is preserved because we can still assume that pk is correctly generated and so no information on b_1 is leaked. However, assuming that V actually knows his own private key, he can clearly use it as trapdoor for the commitment in order to simulate the protocol without interacting with P, and so seeing a transcript will not convince C in this case. This idea is closely related to the concept of verifier designated proofs (see e.g. [7, 17]).

References

1. Brassard, Chaum and Crépeau: *Minimum disclosure proofs of knowledge*, JCSS vol. 37, pp.156-189, 1988.
2. M. Bellare, R. Canetti, and H. Krawczyk: *A modular approach to the design and analysis of authentication and key exchange protocols.* STOC 98.
3. Blum, De Santis, Micali and Persiano: *Non-Interactive Zero-Knowledge*, SIAM J.Computing, vol.20, 1991.
4. Bellare and Goldreich: *Defining proofs of knowledge*, Proc. of Crypto 92, Springer Verlag LNCS series nr. 740.
5. Brandt, Landrock, Damgaård and Pedersen: *Zero-knowledge authentication scheme with secret key exchange*, J.Cryptology, vol.11, pp.147-160, 1998.
6. Cramer and Damgård: *Zero-Knowledge proofs for finite field arithmetic*, Proc. of Crypto 98, Springer Verlag LNCS series nr.1462.
7. Cramer and Damgård: *Fast and Secure Immunization against Man-in-the-Middle Impersonation*, Proc. of EuroCrypt 97.
8. Cramer, Damgaård and Schoenmakers: *Proofs of partial knowlegde*, Proc. of Crypto 94, Springer Verlag LNCS series nr. 839.
9. Canetti, Goldreich, Goldwasser and Micali: *Resettable Zero-Knowledge*, proceedings of STOC 2000.
10. Di Crescenzo and Ostrovsky: *Concurrent Zero-Knowledge: Avoiding Impossibility with Pre-Processing*, Proc. of Crypto 99, to appear.
11. Damgård and Pfitzmann: *Squential Iteration of Interactive Arguments*, proc. of ICALP 98, Springer Verlag LNCS series.
12. Dwork, Naor and Sahai: *Concurrent Zero-Knowledge*, Proc. of STOC 98.
13. Dwork and Sahai: *Concurrent Zero-Knowledge. Reducing the need for timing constraints*, Proc. of Crypto 98, Springer Verlag LNCS series, nr.1462.
14. Feige, Lapidot and Shamir: *Multiple non-interactive zero knowledge proofs based on a single random string*, Proc. of FOCS 90.
15. Feige and Shamir: *Witness indistinguishability and witness hiding protocols*, Proc. of STOC 90.
16. L. Guillou and J.-J. Quisquater: *A Practical Zero-Knowledge Protocol fitted to Security Microprocessor Minimizing both Transmission and Memory*, Proc. of EuroCrypt 88, Springer Verlag LNCS series.
17. Jacobsson, Impagliazzo and Sako: *Designated Verifier Proofs and their Applications*, Proc. of EuroCrypt 96.
18. Joe Kilian: Private communication.
19. Kilian, Petrank and Rackoff: *Lower Bounds for Zero-Knowledge on the Internet*, Proc. of FOCS 98.

20. Feige and Shamir: *Zero-knowledge proofs of knowledge in two rounds*, Proc. of Crypto 89, Springer Verlag LNCS series nr. 435.
21. C.P. Schnorr: *Efficient Signature Generation by Smart Cards*, Journal of Cryptology 4 (1991) 161–174.
22. Goldrecish and Krawczyk: *On the composition of zero-knowledge proof systems*, SIAM J.Computing, vol.25, pp.169-192, 1996.
23. Goldwasser, Micali and Rackoff: *The knowledge complexity of interactive proof systems*, SIAM J.Computing, vol. 18, pp.186-208, 1989.
24. Goldreich, Micali and Wigderson: *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge interactive proofs systems*, JACM, vol 38, pp.691-729, 1991.
25. Pedersen: *Non-interactive and information theoretic secure verifiable secret sharing*, proc. of Crypto 91, Springer Verlag LNCS series, nr. 576.
26. Richardson and Kilian: *On the Concurrent Composition of Zero-Knowledge Proofs*, to appear in Proceedings of EuroCrypt 99.

Efficient Proofs that a Committed Number Lies in an Interval

Fabrice Boudot

France Télécom - CNET
42 rue des Coutures, B.P. 6243
14066 CAEN CEDEX 4, FRANCE
fabrice.boudot@cnet.francetelecom.fr

Abstract. Alice wants to prove that she is young enough to borrow money from her bank, without revealing her age. She therefore needs a tool for proving that a committed number lies in a specific interval. Up to now, such tools were either inefficient (too many bits to compute and to transmit) or inexact (i.e. proved membership to a much larger interval). This paper presents a new proof, which is both *efficient* and *exact*. Here, "efficient" means that there are less than 20 exponentiations to perform and less than 2 Kbytes to transmit. The potential areas of application of this proof are numerous (electronic cash, group signatures, publicly verifiable secret encryption, etc ...).

1 Introduction

The idea of checking whether a committed integer lies in a specific interval was first developed in [2]. Such kind of proofs are intensively used in several schemes: electronic cash systems [7], group signatures [11], publicly verifiable secret sharing schemes [17, 4], and other zero-knowledge protocols (e.g. [13, 10]). Nowadays, there exist two methods to prove that a committed integer is in a specific interval:

- the first one (see e.g. [17]) allows to prove that the bit-length of the committed number is less or equal to a fixed value k, and hence belongs to $[0, 2^k - 1]$. Unfortunately, this method is very inefficient.
- the second one (see e.g. [2, 8]) is much more efficient, but the price to pay is that only membership to a much larger interval can be proven.

In this paper, we give a new method to prove that a committed number belongs to an interval that is much more efficient than the first method and that effectively proves, unlike the second method, that a committed number $x \in I$ belongs to I (and not a larger interval).

Throughout this paper, $\mathbb{Z}_n$ denotes the residue class ring modulo n, and $\mathbb{Z}_n^*$ denotes the multiplicative group of invertible elements in $\mathbb{Z}_n$. $|\cdot|$ denotes binary

length, $a \parallel b$ is the concatenation of the strings a and b. We denote by $\sharp I$ the cardinal of the set I. For $g \in \mathbb{Z}_n^*$ and a in the group generated by g, we denote by $\log_g(a)$ the discrete logarithm of a in base g modulo n, i.e. the number x such that $a = g^x \bmod n$ which belongs to $\{-\mathrm{ord}(g)/2, \ldots, \mathrm{ord}(g)/2 - 1\}$, where $\mathrm{ord}(g)$ is the order of g in $\mathbb{Z}_n^*$. We denote by $PK(x : \mathcal{R}(x))$ a zero-knowledge proof of knowledge of x such that $\mathcal{R}(x)$ is true.

1.1 Definitions

Definition 1 *Let $E = BC(x)$ be a commitment to a value $x \in [b_1, b_2]$. A* proof of membership to an interval $[b_1, b_2]$ *is a proof of knowledge that ensures the verifier that the prover knows x such that $E = BC(x)$ and that x belongs to $[B_1, B_2]$, an interval which contains $[b_1, b_2]$.*

Definition 2 *Following the notations of definition 1, the* expansion rate *of a proof of membership to an interval is the quantity $\delta = (B_2 - B_1)/(b_2 - b_1)$. This quantity may or not be dependent on $(b_2 - b_1)$.*

We evaluate the quality of a proof of membership to an interval by the length of the proof (which must be as short as possible) and by its expansion rate (which must be as low as possible).

1.2 Known Results

In this subsection, we present three existing proofs of membership to an interval. They are based on zero-knowledge proofs of knowledge of a discrete logarithm either modulo a prime (Schnorr [19]) or a composite number (Girault [16]).

1.2.1 Classical Proof [17]

This protocol proves that a committed number $x \in I = [0, b]$ belongs to $I = [0, 2^k - 1]$, where the binary length of b is k.

Let p be a large prime number, let q such that $q|p-1$, and g and h be elements of order q in $\mathbb{Z}_p^*$ such that the discrete logarithm of h in base g is unknown by Alice. We denote by $E(x, r) = g^x h^r \bmod p$ a commitment to x, where r is randomly selected over $\mathbb{Z}_p^*$. Let $x = x_0 2^0 + x_1 2^1 + \cdots + x_{k-1} 2^{k-1}$ for $x_i \in \{0, 1\}$ and $i = 0, 1, \ldots, k-1$ be the binary representation of x. Alice sets $E(x_i, r_i)$ for $i = 0, 1, \ldots, k-1$, where the r_i are such that $\sum_{i=0,\ldots,k-1} r_i = r$, and proves for all i that the number hidden by $E(x_i, r_i)$ is either 0 or 1 by proving that she knows either a discrete logarithm of $E(x_i, r_i)$ in base h or a discrete logarithm of $E(x_i, r_i)/g$ in base h. This can be done using proofs of knowledge of a discrete logarithm [19] and a proof of knowledge of one out of two secrets [5]. Bob also checks that $\prod_{i=0,\ldots,k-1} E(x_i, r_i) = E(x, r)$.

Characteristics of this proof: For $|p| = 1024$ bits, $|q| = 1023$ bits, $|b| = 512$ bits, and the Schnorr's proof security parameter $t = 90$.

- *completeness*: The proof always succeeds.

- *soundness*: A cheating prover can succeed with probability less than $1-(1-2^{-89})^{512} < 2^{-80}$.
- *zero-knowledge*: Perfectly zero-knowledge in the random-oracle model defined in [3].
- *what is proven*: $x \in [0, 2^k - 1]$.
- *expansion rate*: $1 \leq \delta < 2$ (can be decreased to 1 by proving that both x and $b - x$ are k-bit numbers).
- *length of the proof*: 1,612,800 bits = 196.9 kB.

1.2.2 BCDG Proof [2]

This protocol proves that a committed number $x \in I$ belongs to J, where the expansion rate $\sharp J/\sharp I$ is equal to 3. We give here a slightly different presentation from the one of the original paper.

Let t be a security parameter. Let p be a large prime number, let q such that $q|p-1$, and g and h be elements of order q in $\mathbb{Z}_p^*$ such that the discrete logarithm of h in base g is unknown by Alice. We denote by $E = E(x, r) = g^x h^r \bmod p$ a commitment to $x \in [0, b]$, where r is randomly selected over $\mathbb{Z}_p^*$.

For simplicity, we present an interactive version of the protocol which can be easily turned into a non-interactive one using the Fiat-Shamir heuristic [15].

Protocol: $PK_{[BCDG]}(x, r : E = E(x, r) \wedge x \in [-b, 2b])$.
Run t times in parallel:

1. Alice picks random $\omega_1 \in_R [0, b]$ and sets $\omega_2 = \omega_1 - b$. She also randomly selects $\eta_1 \in_R [0, q-1]$ and $\eta_2 \in_R [0, q-1]$, and sends to Bob the unordered pair of commitments $W_1 = g^{\omega_1} h^{\eta_1} \bmod p$ and $W_2 = g^{\omega_2} h^{\eta_2} \bmod p$.
2. Bob challenges Alice by $c \in_R \{0, 1\}$.
3. If $c = 0$, Alice sends to Bob the values of ω_1, ω_2, η_1 and η_2.
   If $c = 1$, Alice sends to Bob the value of $x + \omega_j$, $r + \eta_j$ for the value $j \in \{1, 2\}$ such that $x + \omega_j \in [0, b]$.
4. Bob checks that $W_1 = g^{\omega_1} h^{\eta_1} \bmod p$ and $W_2 = g^{\omega_2} h^{\eta_2} \bmod p$ in the former case and $W_j = g^{\omega_j} h^{\eta_j} \bmod p$, $x + \omega_j \in [0, b]$ in the latter case.

Characteristics of this proof: For $|p| = 1024$ bits, $|q| = 1023$ bits, $|b| = 512$ bits, $t = 80$ and $l = 40$.

- *completeness*: The proof always succeeds if $x \in [0, b]$
- *soundness*: A cheating prover can succeed with probability less than $2 \times 2^{-t} = 2^{-79}$.
- *zero-knowledge*: Perfectly zero-knowledge in the random-oracle model.
- *what is proven*: $x \in [-b, 2b]$.
- *expansion rate*: $\delta = 3$.
- *length of the proof (on average)*: 225,320 bits = 27.5 kB.

1.2.3 CFT Proof [8]

The main idea of this proof is roughly the same as the one of [2]. Let t, l and s be three security parameters. This protocol (due to Chan, Frankel and Tsiounis [7], and corrected in [8], and also due to [14] in another form) proves that a committed number $x \in I$ belongs to J, where the expansion rate $\sharp J/\sharp I$ is equal to 2^{t+l+1}. Let n be a large composite number whose factorization is unknown by Alice and Bob, g be an element of large order in $\mathbb{Z}_n^*$ and h be an element of the group generated by g such that both the discrete logarithm of g in base h and the discrete logarithm of h in base g are unknown by Alice. Let H be a hash-function which outputs $2t$-bit strings. We denote by $E = E(x,r) = g^x h^r \bmod n$ a commitment to $x \in [0,b]$, where r is randomly selected over $[-2^s n+1, 2^s n-1]$. This commitment, from [13], statistically reveals no information about x to Bob.

Protocol: $PK_{[CFT]}(x, r : E = E(x,r) \wedge x \in [-2^{t+l}b, 2^{t+l}b])$.

1. Alice picks random $\omega \in_R [0, 2^{t+l}b - 1]$ and $\eta \in_R [-2^{t+l+s}n + 1, 2^{t+l+s}n - 1]$, and then computes $W = g^\omega h^\eta \bmod n$.
2. Then, she computes $C = H(W)$ and $c = C \bmod 2^t$.
3. Finally, she computes $D_1 = \omega + xc$ and $D_2 = \eta + rc$ (in $\mathbb{Z}$). If $D_1 \in [cb, 2^{t+l}b - 1]$, she sends (C, D_1, D_2) to Bob, otherwise she starts again the protocol.
4. Bob checks that $D_1 \in [cb, 2^{t+l}b - 1]$ and that $C = H(g^{D_1} h^{D_2} E^{-c})$. This convinces Bob that $x \in [-2^{t+l}b, 2^{t+l}b]$.

Characteristics of this proof: For $|n| = 1024$ bits, $|b| = 512$ bits, $t = 80$, $l = 40$ and $s = 40$.

- *completeness*: The proof succeeds with probability greater than $1 - 2^l = 1 - 2^{-40}$ if $x \in [0,b]$.
- *soundness*: A cheating prover can succeed with probability less than 2^{-79}.
- *zero-knowledge*: Statistically zero-knowledge in the random-oracle model.
- *what is proven*: $x \in [-2^{t+l}b, 2^{t+l}b] = [-2^{120}b, 2^{120}b]$.
- *expansion rate*: $\delta = 2^{t+l+1} = 2^{121}$.
- *length of the proof*: 1,976 bits = 0.241 kB.

1.3 Our Results

The schemes we propose in this paper are much more efficient than the classical proof and the BCDG proof, and their expansion rates are $\delta = 1 + \varepsilon$ for the first one, and $\delta = 1$ for the other one, where ε is a negligible quantity with respect to 1 if the considered interval is large enough ($\varepsilon = 2^{-134}$ if the committed number lies in $[0, 2^{512} - 1]$).

We briefly describe our algorithms: first note that it is sufficient to know how to prove that a number is positive to prove that a number belongs to an interval. Indeed, to prove that x belongs to $[a,b]$, it is sufficient to prove that $x - a \geq 0$ and $b - x \geq 0$.

Consider the following commitment scheme: to hide an integer x, Alice computes $E(x,r) = g^x h^r \bmod n$, where n is a composite number whose factorization is unknown by both Alice and Bob, g is an element of large order in $\mathbb{Z}_n^*$, h is an element of large order of the group generated by g such that both the discrete logarithm of g in base h and the discrete logarithm of h in base g are unknown by Alice, r is randomly selected over $[-2^s n + 1, 2^s n - 1]$ and s is a security parameter. This commitment has been introduced in [13], and statistically reveals no information of x to Bob (see section 2.1). Note that this commitment is homomorphic, i.e. $E(x+y, r+s) = E(x,r) \times E(y,s) \bmod n$.

Assume that Alice commits herself to a positive integer x by $E = E(x,r)$ and wants to prove that $x \in [a,b]$.

In our first scheme, Alice writes the positive integer $x-a$ as the sum of x_1^2, the greatest square less than x and of ρ, a positive number less than $2\sqrt{x-a}$ (and therefore less than $2\sqrt{b-a}$). Then, she randomly selects r_1, r_2 in $[0, 2^s n - 1]$ such that $r_1 + r_2 = r$ and computes $E_1 = E(x_1^2, r_1)$ and $E_2 = E(\rho, r_2)$. Then, she proves to Bob that E_1 hides a square in $\mathbb{Z}$ and that E_2 hides a number whose absolute value is less than $2^{t+l+1}\sqrt{b-a}$ by a CFT proof. Finally, she applies the same method to $b-x$. This leads to a proof that $x \in [a - 2^{t+l+1}\sqrt{b-a}, b + 2^{t+l+1}\sqrt{b-a}]$. The expansion rate of this proof is equal to $1 + (2^{t+l+2}/\sqrt{b-a})$, which becomes close to 1 when $b-a$ is large.

In our second scheme, we artificially enlarge the size of x by setting $x' = 2^T x$. By using the first scheme, we prove that $x' \in [2^T a - 2^{t+l+T/2+1}\sqrt{b-a}, 2^T b + 2^{t+l+T/2+1}\sqrt{b-a}]$, and if T is large enough (i.e. T is such that $2^{t+l+T/2+1}\sqrt{b-a} < 2^T$), Bob is convinced that $x' \in [2^T a - 2^T + 1, 2^T b + 2^T - 1]$, so that $x \in [a - \varepsilon, b + \varepsilon]$ where $0 \leq \varepsilon < 1$. So, as x is an integer, Bob is convinced that $x \in [a,b]$.

1.4 Organization of the Paper

In Section 2, we describe some building blocks used in our protocols: a proof that two commitments hide the same secret, and a proof that a committed number is a square. In Section 3, we describe our two schemes: a proof of membership to an interval with tolerance and a proof of membership without tolerance. Then, we extend our results to various commitments. In Section 4, we give an application of our schemes. Finally, we conclude in Section 5.

2 Building Blocks

The schemes we present in this section are based on the following assumption, introduced e.g. in [13]:

Strong RSA Assumption: There exist an efficient algorithm that on input $|n|$ outputs an RSA-modulus n and an element $z \in \mathbb{Z}_n^*$ such that it is infeasible to find integers $e \notin \{-1, 1\}$ and u such that $z = u^e \bmod n$.

2.1 The Fujisaki-Okamoto Commitment Scheme

In this subsection, we briefly describe the commitment scheme we use throughout this paper.

Let s be a security parameter. Let n be a large composite number whose factorization is unknown by Alice and Bob, g be an element of large order in $\mathbb{Z}_n^*$ and h be an element of large order of the group generated by g such that both the discrete logarithm of g in base h and the discrete logarithm of h in base g are unknown by Alice.

We denote by $E = E(x, r) = g^x h^r \bmod n$ a commitment to x in base (g, h), where r is randomly selected over $\{-2^s n + 1, ..., 2^s n - 1\}$.

This commitment has first appeared in [13].

Proposition 1 *$E(x, r)$ is a statistically secure commitment scheme, i.e.:*

- *Alice is unable to commit herself to two values x_1 and x_2 such that $x_1 \neq x_2$ (in $\mathbb{Z}$) by the same commitment unless she can factor n or solve the discrete logarithm of g in base h or the discrete logarithm of h in base g. In other words, under the factoring assumption, it is computationally infeasible to compute x_1, x_2, r_1, r_2 where $x_1 \neq x_2$ such that $E(x_1, r_1) = E(x_2, r_2)$.*
- *$E(x, r)$ statistically reveals no information to Bob. More formally, there exists a simulator which outputs simulated commitments to x which are statistically indistinguishable from true ones.*

As Alice only knows one couple of numbers (x, r) such that $E = g^x h^r \bmod n$, we say that x is the value committed by (or hidden by) E, and that E hides the secret x.

2.2 Proof that Two Commitments Hide the Same Secret

Let t, l, s_1 and s_2 be four security parameters. Let n be a large composite number whose factorization is unknown by Alice and Bob, g_1 be an element of large order in $\mathbb{Z}_n^*$ and g_2, h_1, h_2 be elements of the group generated by g_1 such that the discrete logarithm of g_1 in base h_1, the discrete logarithm of h_1 in base g_1, the discrete logarithm of g_2 in base h_2 and the discrete logarithm of h_2 in base g_2 are unknown by Alice. Let H be a hash-function which outputs $2t$-bit strings. We denote by $E_1(x, r_1) = g_1^x h_1^{r_1} \bmod n$ a commitment to x in base (g_1, h_1) where r_1 is randomly selected over $[2^{s_1} n + 1, 2^{s_1} n - 1]$, and $E_2(x, r_2) = g_2^x h_2^{r_2} \bmod n$ a commitment to x in base (g_2, h_2) where r_2 is randomly selected over $[-2^{s_2} n + 1, 2^{s_2} n - 1]$.

Alice secretly holds $x \in [0, b]$. Let $E = E_1(x, r_1)$ and $F = E_2(x, r_2)$ be two commitments to x. She wants to prove to Bob that she knows x, r_1, r_2 such that $E = E_1(x, r_1)$ and $F = E_2(x, r_2)$, i.e. that E and F hide the same secret x.

This protocol is derived from proofs of equality of two discrete logarithms from [6, 12, 1], combined with a proof of knowledge of a discrete logarithm modulo n [16].

Protocol: $PK(x, r_1, r_2 : E = E_1(x, r_1) \wedge F = E_2(x, r_2))$.

1. Alice picks random $\omega \in [1, 2^{l+t}b-1]$, $\eta_1 \in [1, 2^{l+t+s_1}n-1]$, $\eta_2 \in [1, 2^{l+t+s_2}n-1]$. Then, she computes $W_1 = g_1^{\omega} h_1^{\eta_1} \bmod n$ and $W_2 = g_2^{\omega} h_2^{\eta_2} \bmod n$.
2. Alice computes $c = H(W_1 \parallel W_2)$.
3. She computes $D = \omega + cx$, $D_1 = \eta_1 + cr_1$, $D_2 = \eta_2 + cr_2$ (in $\mathbb{Z}$) and sends (c, D, D_1, D_2) to Bob.
4. Bob checks whether $c = H(g_1^D h_1^{D_1} E^{-c} \bmod n \parallel g_2^D h_2^{D_2} F^{-c} \bmod n)$.

It is shown in [9] that a successful execution of this protocol convinces Bob that the numbers hidden in E and F are equal provided the Strong RSA problem is infeasible.

Characteristics of this proof: For $|n| = 1024$ bits, $|b| = 512$ bits, $t = 80$, $l = 40$, $s_1 = 40$ and $s_2 = 552$.

- *completeness*: The proof always succeeds.
- *soundness*: Under the strong RSA assumption, a cheating prover can succeed with probability less than $2 \times 2^{-t} = 2^{-79}$.
- *zero-knowledge*: Statistically zero-knowledge in the random-oracle model if $1/l$ is negligible.
- *length of the proof*: $2,648 + 2|x|$ bits $= 3672$ bits $= 0.448$ kB.

2.3 Proof that a Committed Number is a Square

Let t, l, and s be three security parameters. Let n be a large composite number whose factorization is unknown by Alice and Bob, g be an element of large order in $\mathbb{Z}_n^*$ and h be an element of the group generated by g such that both the discrete logarithm of g in base h and the discrete logarithm of h in base g are unknown by Alice. Let H be a hash-function which outputs $2t$-bit strings. We denote by $E(x, r) = g^x h^r \bmod n$ a commitment to x in base (g, h) where r is randomly selected over $[-2^s n + 1, 2^s n - 1]$.

Alice secretly holds $x \in [0, b]$. Let $E = E(x^2, r_1)$ be a commitment to the square of x (in $\mathbb{Z}$). She wants to prove to Bob that she knows x and r_1 such that $E = E(x^2, r_1)$, i.e. that E hides the square x^2.

The first proof that a committed number is a square has appeared in [13].

Protocol: $PK(x, r_1 : E = E(x^2, r_1))$.

1. Alice picks random $r_2 \in [-2^s n + 1, 2^s n - 1]$ and computes $F = E(x, r_2)$.
2. Then, Alice computes $r_3 = r_1 - r_2 x$ (in $\mathbb{Z}$). Note that $r_3 \in [-2^s bn + 1, 2^s bn - 1]$. Then, $E = F^x h^{r_3} \bmod n$.
3. As E is a commitment to x in base (F, h) and F is a commitment to x in base (g, h), Alice can run $PK(x, r_2, r_3 : F = g^x h^{r_2} \bmod n \wedge E = F^x h^{r_3} \bmod n)$, the proof that two commitments hide the same secret described in section 2.2. She gets (c, D, D_1, D_2).
4. She sends (F, c, D, D_1, D_2) to Bob.
5. Bob checks that $PK(x, r_2, r_3 : F = g^x h^{r_2} \bmod n \wedge E = F^x h^{r_3} \bmod n)$ is valid.

The soundness of this protocol is clear: if Alice is able to compute F and a proof that E and F are commitments to the same number $\tilde{x}$ resp. in base (F, h) and (g, h), then Alice knows $\tilde{x}$, $\tilde{r}_2$ and $\tilde{r}_3$ such that $E = F^{\tilde{x}} h^{\tilde{r}_3} = g^{\tilde{x}^2} h^{\tilde{x}\tilde{r}_2 + \tilde{r}_3} = g^{\tilde{x}^2} h^{\tilde{r}_1} \bmod n$. Then, this proof shows that Alice knows $\tilde{x}^2$, a square which is hidden in the commitment E. In other words, a successful execution of this protocol convinces Bob that the value hidden in the commitment E is a square in $\mathbb{Z}$.

Technical proofs of the soundness and the zero-knowledgeness of this protocol are easily obtained from the properties of the previous protocol.

Characteristics of this proof: For $|n| = 1024$ bits, $|b| = 512$ bits, $t = 80$, $l = 40$ and $s = 40$.

- *completeness*: The proof always succeeds.
- *soundness*: Under the strong RSA assumption, a cheating prover can succeed with probability less than $2 \times 2^{-t} = 2^{-79}$.
- *zero-knowledge*: Statistically zero-knowledge in the random-oracle model if $1/l$ is negligible.
- *length of the proof*: $3,672 + 2|x|$ bits $= 4696$ bits $= 0.573$ kB.

3 Our Schemes

3.1 Proof that a committed number belongs to an interval

Let t, l and s be three security parameters. Let n be a large composite number whose factorization is unknown by Alice and Bob, g be an element of large order in $\mathbb{Z}_n^*$ and h be an element of the group generated by g such that both the discrete logarithm of g in base h and the discrete logarithm of h in base g are unknown by Alice. We denote by $E(x, r) = g^x h^r \bmod n$ a commitment to x in base (g, h) where r is randomly selected over $[-2^s n + 1, 2^s n - 1]$.

3.1.1 Proof with Tolerance: $\delta = 1 + \varepsilon$

The above protocol allows Alice to prove to Bob that the committed number $x \in [a, b]$ belongs to $[a - \theta, b + \theta]$, where $\theta = 2^{t+l+1}\sqrt{b-a}$.

Protocol: $PK_{[WithTol.]}(x, r : E = E(x, r) \wedge x \in [a - \theta, b + \theta])$.

1. [Knowledge of x]
   Alice executes with Bob:
   $PK(x, r : E = E(x, r))$
2. [Setting]
   Both Alice and Bob compute $\tilde{E} = E/g^a \bmod n$ and $\bar{E} = g^b/E \bmod n$. Alice sets $\tilde{x} = x - a$ and $\bar{x} = b - x$. Now, Alice must prove to Bob that both $\tilde{E}$ and $\bar{E}$ hide secrets which are greater than $-\theta$.

3. [Decomposition of $\tilde{x}$ and $\bar{x}$]
   Alice computes:
   $\tilde{x}_1 = \lfloor\sqrt{x-a}\rfloor$, $\tilde{x}_2 = \tilde{x} - \tilde{x}_1^2$,
   $\bar{x}_1 = \lfloor\sqrt{b-x}\rfloor$, $\bar{x}_2 = \bar{x} - \bar{x}_1^2$.
   Then, $\tilde{x} = \tilde{x}_1^2 + \tilde{x}_2$ and $\bar{x} = \bar{x}_1^2 + \bar{x}_2$, where $0 \leq \tilde{x}_2 \leq 2\sqrt{b-a}$ and $0 \leq \bar{x}_2 \leq 2\sqrt{b-a}$.
4. [Choice of random values for new commitments]
   Alice randomly selects $\tilde{r}_1$ and $\tilde{r}_2$ in $[-2^s n+1, ..., 2^s n-1]$ such that $\tilde{r}_1 + \tilde{r}_2 = r$, and $\bar{r}_1$ and $\bar{r}_2$ such that $\bar{r}_1 + \bar{r}_2 = -r$.
5. [Computation of new commitments]
   Alice computes:
   $\tilde{E}_1 = E(\tilde{x}_1^2, \tilde{r}_1)$, $\tilde{E}_2 = E(\tilde{x}_2, \tilde{r}_2)$,
   $\bar{E}_1 = E(\bar{x}_1^2, \bar{r}_1)$, $\bar{E}_2 = E(\bar{x}_2, \bar{r}_2)$.
6. [Sending of the new commitments]
   Alice sends $\tilde{E}_1$ and $\bar{E}_1$ to Bob. Bob computes $\tilde{E}_2 = \tilde{E}/\tilde{E}_1$ and $\bar{E}_2 = \bar{E}/\bar{E}_1$
7. [Validity of the commitments to a square]
   Alice executes with Bob
   $PK(\tilde{x}_1, \tilde{r}_1 : \tilde{E}_1 = E(\tilde{x}_1^2, \tilde{r}_1))$,
   $PK(\bar{x}_1, \bar{r}_1 : \bar{E}_1 = E(\bar{x}_1^2, \bar{r}_1))$.
   which prove that both $\tilde{E}_1$ and $\bar{E}_1$ hide a square.
8. [Validity of the commitments to a small value]
   Let $\theta = 2^{t+l+1}\sqrt{b-a}$. Alice executes with Bob the two following CFT proofs:
   $PK_{[CFT]}(\tilde{x}_2, \tilde{r}_2 : \tilde{E}_2 = E(\tilde{x}_2, \tilde{r}_2) \wedge \tilde{x}_2 \in [-\theta, \theta])$,
   $PK_{[CFT]}(\bar{x}_2, \bar{r}_2 : \bar{E}_2 = E(\bar{x}_2, \bar{r}_2) \wedge \bar{x}_2 \in [-\theta, \theta])$.
   which prove that both $\tilde{E}_2$ and $\bar{E}_2$ hide numbers which belong to $[-\theta, \theta]$, where $\theta = 2^{t+l+1}\sqrt{b-a}$, instead of proving that they belong to $[0, 2\sqrt{b-a}]$.

Sketch of analysis:

After a successful execution of this protocol, Bob is convinced that :

- $\tilde{E}_1$ and $\bar{E}_1$ hide numbers which are positive integers, as they are squares (Step 7).
- $\tilde{E}_2$ and $\bar{E}_2$ hide numbers which are greater than $-\theta$ (Step 8).
- Alice knows the values hidden by $\tilde{E}$ and $\bar{E}$ (Step 1 and 2).
- The number hidden in $\tilde{E}$ is the sum of the number hidden in $\tilde{E}_1$ and of the number hidden in $\tilde{E}_2$, and so are $\bar{E}$, $\bar{E}_1$ and $\bar{E}_2$ (Step 6).

So, Bob is convinced that $\tilde{E}$ and $\bar{E}$ hide numbers which are greater than $-\theta$, as they are the sum of a positive number and a number greater than $-\theta$.

Let x be the number known by Alice (from step 1) and hidden by E. Bob is convinced that $x - a$ is the value hidden by $\tilde{E}$ and $b - x$ is the value hidden by $\bar{E}$. So, Bob is convinced that $x - a \geq -\theta$ and $b - x \geq -\theta$, i.e. that x belongs to $[a - \theta, b + \theta]$, where $\theta = 2^{t+l+1}\sqrt{b-a}$.

Expansion rate: Following Definition 2, the expansion rate is equal to :

$$\delta = \frac{(b+\theta) - (a-\theta)}{b-a} = 1 + \frac{2\theta}{b-a} = 1 + \varepsilon$$

where:

$$\varepsilon = \frac{2\theta}{b-a} = \frac{2^{t+l+2}}{\sqrt{b-a}} \leq 2^{t+l+2-\lfloor \frac{|b-a|}{2} \rfloor}$$

ε is negligible if and only if $|b-a| \geq 2t+2l+2z+4$, where z is a security parameter. If it is the case, the expansion rate is equal to $\delta = 1+2^{-z}$.

Characteristics of this proof: for $|n| = 1024$ bits, $|b-a| = 512$ bits $t = 80$, $l = 40$ and $s = 40$.

- *length of the proof*: 13860 bits = 1.692 kB.
- *expansion rate*: $\delta = 1+\varepsilon$, where $\varepsilon \leq 2^{t+l+2-\lfloor \frac{|b-a|}{2} \rfloor} = 2^{-134}$.

3.1.2 Proof without Tolerance: $\delta = 1$

The above protocol allows Alice to prove to Bob that the committed number $x \in [a,b]$ belongs to the desired interval $[a,b]$.

To achieve a proof of membership without tolerance, we artificially enlarge the size of x by setting $x' = 2^T x$, where $T = 2(t+l+1)+|b-a|$. Let $E' = E^{2^T}$. E' is a Fujisaki-Okamoto commitment to $x' = 2^T x$ that Alice can open.

By using the first scheme, Alice proves to Bob that she knows the value x' hidden by E' is such that $x' \in [2^T a - 2^{t+l+T/2+1}\sqrt{b-a}, 2^T b + 2^{t+l+T/2+1}\sqrt{b-a}]$ by a CFT proof (instead of proving that $x' \in [2^T a, 2^T b]$).

As $T = 2(t+l+1)+|b-a|$, we have:

$$\begin{aligned}
\theta' = 2^{t+l+T/2+1}\sqrt{b-a} &< 2^{t+l+T/2+1} \times 2^{\lceil (|b-a|-1)/2 \rceil} \\
&< 2^{T/2} \times 2^{t+l+1} \times 2^{\lceil (|b-a|-1)/2 \rceil} \\
&< 2^{T/2} \times 2^{T/2} \\
&< 2^T
\end{aligned}$$

Then, if Bob is convinced that $x' \in [2^T a - \theta', 2^T b + \theta']$, he is also convinced that $x' \in]2^T a - 2^T, 2^T b + 2^T[$.

Provided Alice does not know the factorization of n, she is unable to know two different values in $\mathbb{Z}$ hidden by E'. So, necessarily, $x' = 2^T x$. The proof convinces Bob that $2^T x \in]2^T a - 2^T, 2^T b + 2^T[$, and so that $x \in]a-1, b+1[$. Finally, as x is an integer, Bob is convinced that $x \in [a,b]$.

Protocol: $PK(x, r : E = E(x,r) \wedge x \in [a,b])$.

1. [Setting]
   Both Alice and Bob compute $E' = E^{2^T}$, where $T = 2(t+l+1)+|b-a|$.
2. [Proof]
   Alice executes with Bob:
   $PK_{[WithTol.]}(x', r' : E' = E(x', r') \wedge x' \in [2^T a - 2^{t+l+T/2+1}\sqrt{b-a}, 2^T b + 2^{t+l+T/2+1}\sqrt{b-a}]$.

Characteristics of this proof: for $|n| = 1024$ bits, $|b-a| = 512$ bits, $t = 80$, $l = 40$ and $s = 40$.

- *length of the proof*: 16176 bits = 1.975 kB.
- *expansion rate*: $\delta = 1$.

3.2 Extensions

The above protocols can be used to prove that:

- a discrete logarithm modulo a composite number n whose factorization is unknown to Alice belongs to an interval. Let g be an element of large order in $\mathbb{Z}_n^*$ and h be an element of the group generated by g such that both the discrete logarithm of g in base h and the discrete logarithm of h in base g are unknown by Alice. Let x be such that $y = g^x \bmod n$. Alice randomly selects r and computes $y' = h^r \bmod n$. She proves to Bob that she knows a discrete logarithm of y' in base h, and then that $yy' = g^x h^r \bmod n$ is a commitment to a value which belongs to the given interval.
- a discrete logarithm modulo p (a prime number or a composite number whose factorization is known to Alice) belongs to an interval. Let x be such that $Y = G^x \bmod p$. Alice randomly selects r and computes $E = E(x,r) = g^x h^r \bmod n$, a commitment to x. Then, she executes with Bob $PK(x, r : Y = G^x \bmod p \wedge E = g^x h^r \bmod n)$ (see Appendix A) and $PK(x, r : E = g^x h^r \bmod n \wedge x \in [a, b])$.
- a third root (or, more generally, a e-th root) modulo N belongs to an interval. Let x be such that $Y = x^3 \bmod N$. Alice randomly selects r and computes $E = E(x,r) = g^x h^r \bmod n$, a commitment to x. Then, she executes with Bob $PK(x, r : Y = x^3 \bmod N \wedge E = g^x h^r \bmod n)$ (see Appendix B) and $PK(x, r : E = g^x h^r \bmod n \wedge x \in [a, b])$.

Note: to prove that a committed number x lies in $I \cup J$, Alice proves that x lies in I or x lies in J by using a proof of "or" by [5].

4 Application to Verifiable Encryption

As one of the several applications of proofs of membership to an interval, we present in this section an efficient (publicly) verifiable encryption scheme.

Alice has sent two encrypted messages to Charlie and Deborah, and wants to prove to Bob that the two ciphertexts encrypt the same message.

Charlie and Deborah use the Okamoto-Uchiyama [18] cryptosystem, i.e. Charlie holds a composite number $n_C = p_C^2 q_C$ ($|p_C| = |q_C| = k$), an element $g_C \in \mathbb{Z}_{n_C}^*$ such that the order of $g_C^{p_C - 1} \bmod p_C^2$ is p_C, and Deborah holds a composite number $n_D = p_D^2 q_D$ ($|p_D| = |q_D| = k$), an element $g_D \in \mathbb{Z}_{n_D}^*$ such that the order of $g_D^{p_D - 1} \bmod p_D^2$ is p_D.

We denote by $h_C = g_C^{n_C} \bmod n_C$ and $h_D = g_D^{n_D} \bmod n_D$.

To encrypt a message m such that $0 \leq m \leq 2^{k-1}$ intended to Charlie, Alice computes $E_C = g_C^m h_C^{r_C} \bmod n_C$, where r_C is randomly selected over $\mathbb{Z}_{n_C}^*$. In the same way, she encrypts the same message m intended to Deborah by computing $E_D = g_D^m h_D^{r_D} \bmod n_D$.

Now, Alice wants to prove to Bob that the two ciphertexts E_C and E_D encrypt the same message.

First, she executes with Bob $PK(m, r_C, r_D : E_C = g_C^m h_C^{r_C} \bmod n_C \wedge E_D = g_D^m h_D^{r_D} \bmod n_D)$, a proof of equality of two committed numbers with respect to different moduli (see Appendix A). This only proves that she knows an integer m such that $m \bmod p_C$ and $m \bmod p_D$ are respectively the messages decrypted by Charlie and Deborah. Note that if m is greater than p_C and p_D, then $m \bmod p_C \neq m \bmod p_D$. So it is necessary that Alice also proves to Bob that m is less than p_C and p_D. Alice uses the proof of membership to an interval without tolerance presented in section 3.1.2: $PK(m, r_C : E_C = g_C^m h_C^{r_C} \bmod n_C \wedge m \in [0; 2^{k-1}])$. Then, necessarily, $m \bmod p_C = m \bmod p_D$: Bob is convinced that Alice has secretly sent the same messages to Charlie and to Deborah.

5 Conclusion

We have presented in this paper efficient proofs that a committed number belongs to an interval and give examples of applications, more particularly an efficient verifiable encryption scheme. By their efficiency, they are well suited to be used in various cryptographic protocols.

Acknowledgements

We would like to thank Marc Girault for helpful discussions and comments.

References

1. Bao, F.: An Efficient Verifiable Encryption Scheme for Encryption of Discrete Logarithms. Proceedings of CARDIS'98 (1998)
2. Brickell, E., Chaum, D., Damgård, I., Van de Graaf, J.: Gradual and Verifiable Release of a Secret. Proceedings of CRYPTO'87, LNCS **293** (1988) 156–166
3. Bellare, M., Rogaway, P.: Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. Proceedings of the First Annual Conference and Communications Security (1993) 62–73
4. Boudot, F., Traoré, J.: Efficient Publicly Verifiable Secret Sharing Schemes with Fast or Delayed Recovery. Proceedings of the Second International Conference on Information and Communication Security, LNCS **1726** (1999) 87–102
5. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. Proceedings of CRYPTO'94, LNCS **839** (1997) 174–187
6. Chaum, D., Evertse, J.-H., Van de Graaf, J.: An Improved Protocol for Demonstrating Possession of Discrete Logarithm and Some Generalizations. Proceedings of EUROCRYPT'87, LNCS **304** (1998) 127–141
7. Chan, A., Frankel, Y., Tsiounis, Y,: Easy Come - Easy Go Divisible Cash. Proceedings of EUROCRYPT'98, LNCS **1403** (1998) 561–575
8. Chan, A., Frankel, Y., Tsiounis, Y,: Easy Come - Easy Go Divisible Cash. Updated version with corrections, GTE Tech. Rep. (1998), available at `http://www.ccs.neu.edu/home/yiannis/`

9. Camenisch, J., Michels, M.: A Group Signature Scheme Based on an RSA-Variant. Tech. Rep. **RS-98-27**, BRICS, Dept. of Comp. Sci., University of Aarhus, available at `http://www.zurich.ibm.com/~jca/` (1998)
10. Camenisch, J., Michels, M.: Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. Proceedings of EUROCRYPT'99, LNCS **1592** (1999) 106–121
11. Camenisch, J., Michels, M.: Separability and Efficiency for Generic Group Signature Schemes. Proceedings of CRYPTO'99, LNCS **1666** (1999) 413–430
12. Chaum, D., Pedersen, T.-P.: Wallet Databases with Observers. Proceedings of CRYPTO'92, LNCS **740** (1992) 89–105
13. Fujisaki, E., Okamoto, T.: Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. Proceedings of CRYPTO'97, LNCS **1294** (1997) 16–30
14. Fujisaki, E., Okamoto, T.: A Practical and Provably Secure Scheme for Publicly Verifiable Secret Sharing and Its Applications, Proceedings of EUROCRYPT'98, LNCS **1403** (1998) 32–46
15. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. Proceedings of CRYPTO'86, LNCS **263** (1986) 186–194
16. Girault, M.: Self-Certified Public Keys. Proceedings of EUROCRYPT'91, LNCS **547** (1991) 490–497
17. Mao, W.: Guaranteed Correct Sharing of Integer Factorization with Off-line Shareholders. Proceedings of Public Key Cryptography 98, (1998) 27–42
18. Okamoto, T., Uchiyama, S.: A New Public-Key Cryptosystem as Secure as Factoring. Proceedings of EUROCRYPT'98, LNCS **1403** (1998) 308–318
19. Schnorr, C.-P.: Efficient Signature Generation for Smart Cards Journal of Cryptology, (**4:3**) (1991) 239–252

A Proof of equality of two committed numbers in different moduli

This proof originally appeared in[4] and independently in [10] in a more general form.

Let t, l and s be three security parameters. Let n_1 be a large composite number whose factorization is unknown by Alice and Bob, and n_2 be another large number, prime or composite whose factorization is known or unknown by Alice. Let g_1 be an element of large order in $\mathbb{Z}^*_{n_1}$ and h_1 be an element of the group generated by g_1 such that both the discrete logarithm of g_1 in base h_1 and the discrete logarithm of h_1 in base g_1 are unknown by Alice. Let g_2 be an element of large order in $\mathbb{Z}^*_{n_2}$ and h_2 be an element of the group generated by g_2 such that both the discrete logarithm of g_2 in base h_2 and the discrete logarithm of h_2 in base g_2 are unknown by Alice. Let H be a hash-function which outputs $2t$-bit strings. We denote by $E_1(x, r_1) = g_1^x h_1^{r_1} \bmod n_1$ a commitment to x in base (g_1, h_1) where r_1 is randomly selected over $\{-2^s n + 1, ..., 2^s n - 1\}$, and $E_2(x, r_2) = g_2^x h_2^{r_2} \bmod n_2$ a commitment to x in base (g_2, h_2) where r_2 is randomly selected over $\{-2^s n + 1, ..., 2^s n - 1\}$.

Alice secretly holds $x \in \{0, \ldots, b\}$. Let $E = E_1(x, r_1)$ and $F = E_2(x, r_2)$ be two commitments to x. She wants to prove to Bob that she knows x, r_1, r_2 such that $E = E_1(x, r_1)$ and $F = E_2(x, r_2)$, i.e. that E and F hide the same secret x.

Protocol: $PK(x, r_1, r_2 : E = E_1(x, r_1) \bmod n_1 \wedge F = E_2(x, r_2) \bmod n_2)$.

1. Alice picks random $\omega \in \{1, \ldots, 2^{l+t}b - 1\}$, $\eta_1 \in \{1, \ldots, 2^{l+t+s}n - 1\}$, $\eta_2 \in \{1, \ldots, 2^{l+t+s}n - 1\}$. Then, she computes $W_1 = g_1^{\omega} h_1^{\eta_1} \bmod n_1$ and $W_2 = g_2^{\omega} h_2^{\eta_2} \bmod n_2$.
2. Alice computes $c = H(W_1 \parallel W_2)$.
3. She computes $D = \omega + cx$, $D_1 = \eta_1 + cr_1$, $D_2 = \eta_2 + cr_2$ (in $\mathbb{Z}$) and sends (c, D, D_1, D_2) to Bob.
4. Bob checks whether $c = H(g_1^D h_1^{D_1} E^{-c} \bmod n_1 \parallel g_2^D h_2^{D_2} F^{-c} \bmod n_2)$.

Note that this protocol can be used to prove the equality of more than two committed numbers, or to prove the equality of a committed number modulo n_1 and a discrete logarithm modulo n_2 by setting r_2, η_2 and D_2 to zero.

B Proof of equality of a third root and a committed number

This proof is derived from [14].

Let n_1 be a large composite number whose factorization is unknown by Alice and Bob, and n_2 be another large composite number whose factorization is known or unknown by Alice. Let g_1 be an element of large order in $\mathbb{Z}_{n_1}^*$ and h_1 be an element of the group generated by g_1 such that both the discrete logarithm of g_1 in base h_1 and the discrete logarithm of h_1 in base g_1 are unknown by Alice. We denote by $E_1(x, r_1) = g_1^x h_1^{r_1} \bmod n_1$ a commitment to x in base (g_1, h_1) where r_1 is randomly selected over $\{-2^s n + 1, ..., 2^s n - 1\}$. We also denote by $E_2(x) = x^3 \bmod n_2$ a RSA$(n_2, 3)$ encryption of x.

Alice secretly holds $x \in \{0, \ldots, b\}$. Let $E = E_1(x, r_1)$ and $F = E_2(x) = x^3 \bmod n_2$ be a commitment to x and a RSA encryption to x. She wants to prove to Bob that she knows x and r_1 such that $E = E_1(x, r_1)$ and $F = E_2(x)$, i.e. that E and F hide the same secret x.

Protocol: $PK(x, r_1, r_2 : E = E_1(x, r_1) \bmod n_1 \wedge F = E_2(x) \bmod n_2)$.

1. Alice computes $\alpha = \frac{F - x^3}{n_2}$ (in $\mathbb{Z}$), $G_2 = E_1(x^2, r_2)$, $G_3 = E_1(x^3, r_3)$ and $Z = E_1(\alpha n_2, -r_3)$.
2. Alice proves to Bob that E, G_2 and G_3 are commitments to the same value respectively in bases (g_1, h_1), (E, h_1) and (G_1, h_1), and that she knows which value is committed by Z in base $(g_1^{n_2}, h_1)$.
3. Bob checks these proofs, computes $T = g_1^F \bmod n_1$ and checks that $T = G_3 Z \bmod n_1$.

A Composition Theorem for Universal One-Way Hash Functions

Victor Shoup

IBM Zurich Research Lab, Säumerstr. 4, 8803 Rüschlikon, Switzerland
sho@zurich.ibm.com

Abstract. In this paper we present a new scheme for constructing universal one-way hash functions that hash arbitrarily long messages out of universal one-way hash functions that hash fixed-length messages. The new construction is extremely simple and is also very efficient, yielding shorter keys than previously proposed composition constructions.

1 Introduction

In this paper we consider the problem of constructing *universal one-way hash functions* (UOWHFs).

The notion of a UOWHF was introduced by Naor and Yung [NY89]. A UOWHF is a keyed hash function with the following property: if an adversary chooses a message x, and then a key K is chosen at random and given to the adversary, it is hard for he adversary to find a different message $x' \neq x$ such that $H_K(x) = H_K(x')$.

As a cryptographic primitive, a UOWHF is an attractive alternative to the more traditional notion of a *collision-resistant hash function* (CRHF), which is characterized by the following property: given a random key K, it is hard to find two different messages x, x' such that $H_K(x) = H_K(x')$.

A reasonable approach to designing a UOWHF that hashes messages of arbitrary and variable length is to first design a *compression function*, that is, UOWHF that hashes fixed-length messages, and then design a method for composing these compression functions so as to hash arbitrary messages. In this paper, we address the second problem, that of composing compression functions. The main technical problem in designing such composition schemes is to keep the key length of the composite scheme from getting too large.

This composition problem was studied in some detail by Bellare and Rogaway [BR97]. They proposed and analyzed several composition schemes.

In this paper, we propose and analyze a new composition scheme. This scheme is extremely simple, and yields shorter keys than previously proposed schemes.

We also suggest an efficient and concrete implementation based on our composition technique, using a standard "off the shelf" compression function, like SHA-1, under the weak assumption of second preimage collision resistance.

2 UOWHFs versus CRHFs

A UOWHF is an attractive alternative to a CRHF because

(1) it seems easier to build an efficient and secure UOWHF than to build an efficient and secure CRHF, and
(2) in many applications, most importantly for building digital signature schemes, a UOWHF is sufficient.

As evidence for claim (1), we point out the recent attacks on MD5 [dBB93,Dob96]. We also point out the complexity theoretic result of Simon [Sim98] that shows that there exists an oracle relative to which UOWHFs exist but CRHFs do not. CRHFs can be constructed based on the hardness of specific number-theoretic problems, like the discrete logarithm problem [Dam87]. Simon's result is strong evidence that CRHFs cannot be constructed based on an arbitrary one-way permutation, whereas Naor and Yung [NY89] show that a UOWHF can be so constructed.

As for claim (2), one of the main applications of collision resistant hashing is digital signatures. The idea is to create a short "message digest" that can then be signed using a signature algorithm that needs to work only on short messages. As pointed out by Bellare and Rogaway [BR97], a UOWHF suffices for this. To sign a message x, the signer chooses a key K for a UOWHF H, and produces the signature $(K, \sigma(K, H_K(x)))$, where σ is the underlying signing function for short messages. For some UOWHFs, the key K can grow with the message length—indeed, the scheme we propose here has a key that grows logarithmically with the message length. This can lead to technical difficulties, since then the message we need to sign with σ can get too large. One solution to this problem is to instead make the signature $(K, \sigma(H_{K'}(K), H_K(x)))$, where K' is a UOWHF key that is part of the signer's public key. This is a somewhat simpler solution to this problem than the one presented in [BR97], and we leave it to the reader to verify the security of this composite signature scheme.

Naor and Yung [NY89] in fact show how to build a secure digital signature scheme based solely on a UOWHF; however, the resulting scheme is not particularly practical.

3 Previous Composition Constructions

We briefly summarize here previous constructions for composing UOWHFs.

We assume we have UOWHF H that maps strings of length a to strings of length b, where $a > b$, and that H is keyed by a key K. The goal is to build from this a composite UOWHF that hashes messages of arbitrary and variable length. To simplify the discussion, we restrict our attention in this section to the problem of hashing long, but fixed-length messages. There are general techniques to deal with variable length messages (see [BR97]).

The simplest construction is the *linear hash*. Let $m = a - b$. Suppose the message x consists of l blocks $x_1, \ldots, x_l$, where each block is an m-bit string.

Then using l keys $K_1, \ldots, K_l$ for H, and an arbitrary b-bit "initial vector" h_0, we compute h_i for $1 \leq i \leq l$ as $h_i = H_{K_i}(h_{i-1} \circ x_i)$, where "$\circ$" denotes concatenation. The output of the composite hash function is h_l.

The security of this scheme is analyzed in detail in [BR97]. Note that we need to use l independent keys $K_1, \ldots, K_l$. If we use instead just a single key, the resulting scheme does not necessarily preserve the UOW property of the compression function. This situation is quite different from the situation where we are constructing a composite hash function out of a CR compression function; in that situation, the composite hash function does indeed inherit the CR property from the compression function [Dam89,Mer89].

Although the linear hash is quite simple, it is not very attractive from a practical point of view, as the key length for the composite scheme grows linearly with the message length.

If the keys for the compression function are longer than the output length b of the compression function, then a variant of the linear hash, the *XOR linear hash* [BR97], yields somewhat shorter, though still linear sized keys. In this scheme, we use a single key K for the compression function H, and in addition, the key of the composite scheme has l "masks" $M_1, \ldots, M_l$, each of which is a random b-bit string. The scheme is then the same as the linear hash, except that we compute h_i for $1 \leq i \leq l$ as $h_i = H_K((h_{i-1} \oplus M_i) \circ x_i)$.

As pointed out by Naor and Yung [NY89], we can get composite schemes with logarithmic key size by using a *tree hash*, which is the same as a construction proposed by Wegman and Carter [WC81] for composing universal hash functions. For simplicity, assume that $a = bd$ for an integer d, and that we want to hash messages of length bd^t for some $t > 0$. Then we hash using a tree evaluation scheme, where at each level i of the tree, for $1 \leq i \leq t$, we hash bd^i bits to bd^{i-1} bits. At a given level i, we apply the compression function H d^{i-1} times, using the same key K_i. So in the composite scheme we need t keys $K_1, \ldots, K_t$.

If the keys of the compression function are long, a more efficient scheme is the *XOR tree hash* [BR97]. This is the same as the tree hash scheme, except as follows. We used a single compression function key K, and in addition, we use t "masks" $M_1, \ldots, M_t$, each of which is a random a-bit string. Whenever we evaluate the compression function at level i in the tree, we "mask" its input with M_i; that is, we compute its input as the bit-wise exclusive-or of M_i and the input used in the normal tree hash.

The new scheme we present in the next section most closely resembles the XOR linear hash, except that we re-use the masks as much as possible to minimize the key length. The key length of the new scheme is smaller (asymptotically) than the key length of the XOR tree hash by a factor of $d/\log_2 d$, while at the same requiring essentially the same amount of computation. This, combined with the fact that the new scheme is extremely simple, makes it an attractive alternative to the XOR tree hash.

4 The New Scheme

We now describe our new scheme, which is the same as the XOR linear hash, except that we get by with a smaller number of masks. Since it is not difficult to do, we describe how our scheme works for variable length messages.

Again, our starting point is a UOW compression function H that is keyed by a key K, and compresses a bits to b bits. Let $m = a - b$. We assume that a message x is formatted as a sequence of l blocks $x_1, \ldots, x_l$, each of which is an m-bit string, and we assume that the last block x_l encodes the bit length of x in some canonical way. The number of blocks l may vary, but we assume that $l \leq L$ for some given L.

The key for the composite scheme consists of a single key K for H, together with a number of "masks," each of which is a random b-bit string. We need $t+1$ masks $M_0, \ldots, M_t$, where $t = \lceil \log_2 L \rceil$.

To define the scheme, we use the function $\nu_2(i)$ which counts the number of times 2 divides i, i.e., for $i \geq 1$, $\nu_2(i)$ is the largest integer ν such that 2^ν divides i.

The hash function is defined as follows. Let h_0 be an arbitrary b-bit string. For $1 \leq i \leq l$, we define $h_i = H_K((M_{\nu_2(i)} \oplus h_{i-1}) \circ x_i)$. The output of the composite hash is h_l.

Theorem 1. *If H is a UOWHF, then the above composite scheme is also a UOWHF.*

The remainder of this section is devoted to a proof of this theorem. We show how an adversary A that finds collisions in the composite scheme can be turned into an adversary A' that finds collisions in the compression function H. This reduction is quite efficient: the running time of A' is essentially the same as that of A, and if A finds a collision with probability ϵ, then A' finds a collision with probability about ϵ/L.

We begin with an auxiliary definition. Let x be an input to the composite hash function; for $1 \leq i \leq l$, define $S_i(x)$ be the first b bits of the input to the ith application of the compression function H. The definition of $S_i(x)$ depends, of course, on the value of the composite hash function's key, which will be clear from context.

Consider the behavior of adversary A. Suppose its first message x—the "target" message—is formatted as $x_1, \ldots, x_l$, and its second message x' that yields the collision is formatted as $x'_1, \ldots, x'_{l'}$.

For this collision, we let δ be the smallest nonnegative integer such that $S_{l-\delta}(x) \circ x_i \neq S_{l'-\delta}(x') \circ x'_i$. Since we are encoding the bit length of a message in the last message block, if the bit lengths of x and x' differ, then clearly $\delta = 0$. Otherwise, $l = l'$ and it is easy to see that δ is well defined.

The pair $S_{l-\delta}(x) \circ x_i$, $S_{l'-\delta}(x') \circ x'_i$ will be the collision on H_K that A' finds.

The adversary A' runs as follows. We let A choose its first message x. Then A' guesses the value of δ at random. This guess will be right with probability $1/L$. A' now constructs its target message as $S \circ x_{l-\delta}$, where S is a random

b-bit string. Now a random key K for the compression function H is chosen. The task of A' is to generate masks $M_0, \ldots, M_t$ such that the composite key $(K, M_0, \ldots, M_t)$ has the correct distribution, and also that $S_{l-\delta}(x) = S$. Once this is accomplished, the adversary A attempts to find a collision with x. If A succeeds, and if the guess at δ was correct, this will yield a collision for A'.

We now present a "key construction" algorithm that on input x, δ, K, S, t as above, generates masks $M_0, \ldots, M_t$ as required. The algorithm to do this is described in Figure 1.

```
    for (j ← 0; j ≤ t; j ← j + 1) status_j ← "undefined"
    S_{l−δ} ← S
    status_{ν2(l−δ)} ← "being defined"
    for (i ← l − δ − 1; i ≥ 1; i ← i − 1) {
      j ← ν2(i)
(1)   if (status_j = "undefined") {
        choose S_i as a random b-bit string
        status_j ← "being defined"
        h_i ← H_K(S_i ∘ x_i)
        i' ← i + 1; j' ← ν2(i')
        while (status_{j'} ≠ "being defined") {
          h_{i'} ← H_K((h_{i'−1} ⊕ M_{ν2(i')}) ∘ x_{i'})
          i' ← i' + 1; j' ← ν2(i')
        }
        M_{j'} ← h_{i'−1} ⊕ S_{i'}
        status_{j'} ← "defined"
      }
    }
    i' ← 1; j' ← 0
    while (status_{j'} ≠ "being defined") {
      h_{i'} ← H_K((h_{i'−1} ⊕ M_{ν2(i')}) ∘ x_{i'})
      i' ← i' + 1; j' ← ν2(i')
    }
    M_{j'} ← h_{i'−1} ⊕ S_{i'}
    status_{j'} ← "defined"

    for (j ← 0; j ≤ t; j ← j + 1)
      if (status_j = "undefined") choose M_j as a random b-bit string
```

Fig. 1. Key Construction Algorithm

We can describe the algorithm at a high level as follows. During the course of execution, each mask M_j, for $0 \leq j \leq t$, has a status, $status_j$, where the status is one of the values "undefined," "being defined," or "defined." Initially, each status value is "undefined." As the algorithm progresses, the status of a

mask changes first to "being defined," and finally to "defined," at which point the algorithm actually assigns a value the mask.

The algorithm starts at block $l - \delta$, and assigns the value S to $S_{l-\delta}$, where in general, S_i represents the value of $S_i(x)$ for $1 \leq i \leq l$. The algorithm sets the status of mask $\nu_2(l - \delta)$ to "being defined." Now the algorithm considers blocks $l - \delta - 1, l - \delta - 2, \ldots, 1$ in turn. When it reaches block i in this "right to left sweep," it looks at the status of mask $j = \nu_2(i)$. As we shall prove below, the status of this mask j is never "being defined" at this moment in time. If the status is "defined," it skips to the next value of i. If the status is "undefined," then it chooses S_i at random, and changes the status of mask j to "being defined." The algorithm also runs the hash algorithm from "left to right," computing $h_i, h_{i+1}, \ldots, h_{i'-1}$, until it finds a block i' whose mask $j' = \nu_2(i')$ has the status "being defined." At this point, the mask j' is computed as $M_{j'} = h_{i'-1} \oplus S_{i'}$, and the status of mask j' is changed to "defined." Thus, at any point in time, there is exactly one mask whose status is "being defined," except briefly during the "left to right hash evaluation."

When the algorithm finishes the "right to left sweep," there will still be one mask whose status is "being defined," and the "left to right hash evaluation" as described above is used to define this mask, thereby converting its status to "defined." There may still be other masks whose status is "undefined," and these are simply assigned random values.

The key to analyzing this algorithm is to show that when we visit block i in the "right to left sweep," we do not encounter a mask $j = \nu_2(i)$ such that the status of mask j is "being defined." Let us make this more precise. As i runs from $l - \delta - 1$ down to 1 in the main loop, let V_i be the value of $status_j$ when the line marked (1) in Figure 1 is executed. We prove below in Lemma 1 that $V_i \neq$ "being defined" for all i. So long as this is the case, we avoid circular definitions, and it is easy to see that the algorithm constructs masks $M_0, \ldots, M_t$ with just the right distribution. Indeed, the key construction algorithm implicitly defines a one-to-one map between tuples $(K, M_0, \ldots, M_t)$ and $(K, S, S^{(1)}, \ldots, S^{(t)})$, where $S^{(1)}, \ldots, S^{(t)}$ are randomly chosen b-bit strings, and $S = S_{l-\delta}(x)$.

So the proof of Theorem 1 now depends on the following lemma.

Lemma 1. *For $1 \leq i \leq l - \delta - 1$, $V_i \neq$ "being defined."*

To prove this lemma, we need two simple facts, which we leave to the reader to verify.

Fact 1. *For any positive integers $A < B$ with $\nu_2(A) = \nu_2(B)$, there exists an integer C with $A < C < B$ and $\nu_2(C) > \nu_2(A)$.*

Fact 2. *For any positive integers $A < B$, and for any nonnegative integer $\nu < \min\{\nu_2(A), \nu_2(B)\}$, there exists an integer C with $A < C < B$ and $\nu_2(C) = \nu$.*

Now to the proof of the lemma. Suppose $V_i =$ "being defined" for some i, and let A be the largest such value of i. Then there must be a *unique* integer B

with $A < B \leq l - \delta$ such that $\nu_2(B) = \nu_2(A)$. This is the point where we set the status of mask $\nu_2(A)$ to "being defined." The uniqueness of B follows from the maximality of the choice of A.

By Fact 1, there must be an index C with $A < C < B$ and $\nu_2(C) > \nu_2(A)$. There may be several such C; among these, choose from among those with maximal $\nu_2(C)$, and from among these, choose the largest one.

We claim that $V_C =$ "defined." To see this, note that we cannot have $V_C =$ "being defined," since we chose A to be the maximal index with this property. Also, we could not have since $V_C =$ "undefined," since then we would have defined mask $\nu_2(A)$ at this point, and we would have $V_A =$ "defined."

Since $V_C =$ "defined," we must have set the status of mask $\nu_2(C)$ to "being defined" in a loop iteration prior to C. Thus, there must exist D with $C < D \leq l - \delta$ and $\nu_2(D) = \nu_2(C)$. By the way we have chosen C, we must have $D > B$.

Again by Fact 1, there exists integer E with $C < E < D$, and $\nu_2(E) > \nu_2(C)$. Again, by the choice of C, we must have $E > B$.

Finally, by Fact 2, there exists an integer F with $E < F < D$ and $\nu_2(F) = \nu_2(A)$. So we have $B < F < l - \delta$ with $\nu_2(F) = \nu_2(A)$, which is a contradiction. That completes the proof of the lemma. See Figure 2 for a visual aid.

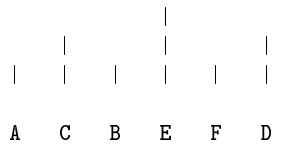


Fig. 2. From the proof of Lemma 1. The vertical lines represent the relative magnitudes of the corresponding values of ν_2.

5 A Concrete Implementation

In this section, we suggest a concrete implementation for a practical UOWHF.

Given a method for building a composite UOW hash function out of a UOW compression function, one still has to construct a UOW compression function. A pragmatic approach is to use an "off the shelf" compression function such as the SHA-1 compression function $C : \{0,1\}^{160} \times \{0,1\}^{512} \to \{0,1\}^{160}$. The assumption we make about C is that it is *second preimage collision resistant*, i.e., if a random input (S, B) is chosen, then it is hard to find different input $(S', B') \neq (S, B)$ such that $C(S, B) = C(S', B')$. This assumption seems to be much weaker than assumption that no collisions in C can be found at all (which as an intractability assumption is not even well defined). Indeed, the techniques used to find collisions in MD5 [dBB93,Dob96] do not appear to help in finding second preimages.

Note that from a complexity theoretic point of view, second preimage collision resistance is no stronger than the UOW property. Indeed, if $H_K(x)$ is a UOWHF, then the function sending (K, x) to $(K, H_K(x))$ is second preimage collision resistant.

The second preimage resistance assumption on C allows us to build a UOW compression function as follows. The key is a random element $(\hat{S}, \hat{B})$ in the domain of C, and the value of the compression function on (S, B) is $C(\hat{S} \oplus S, \hat{B} \oplus B)$.

We could apply our composition construction directly to this. However, there is one small optimization possible; namely, we can eliminate $\hat{S}$ from the key.

We can now put this all together. Assume that a message x is formatted as a sequence $x_1, \ldots, x_l$ of 512-bit blocks, where the last block encodes the bit length of x. Let L be an upper bound on l, and let $t = \lceil \log_2 L \rceil$. The key for our hash function consists of a random 512-bit string $\hat{B}$, along with $t+1$ 160-bit strings $M_0, \ldots, M_t$. Then the hash of x is defined to be h_l, where h_0 is an arbitrary 160-bit string, and $h_i = C(h_{i-1} \oplus M_{\nu_2(i)}, x_i \oplus \hat{B})$ for $1 \leq i \leq l$.

Our analysis shows that this hash function is UOW, assuming C is second preimage collision resistant.

References

[BR97] M. Bellare and P. Rogaway. Collision-resistant hashing: towards making UOWHFs practical. In *Advances in Cryptology–Crypto '97*, 1997.

[Dam87] I. Damgard. Collision free hash functions and public key signature schemes. In *Advances in Cryptology–Eurocrypt '87*, 1987.

[Dam89] I. Damgard. A design principle for hash functions. In *Advances in Cryptology–Crypto '89*, 1989.

[dBB93] D. den Boer and A. Bosselaers. Collisions for the compression function of MD5. In *Advances in Cryptology–Eurocrypt '93*, pages 293–304, 1993.

[Dob96] H. Dobbertin. The status of MD5 after a recent attack. *RSA Laboratories' CryptoBytes*, 2(2), 1996. The main result of this paper was announced at the Eurocrypt '96 rump session.

[Mer89] R. Merkle. One way hash functions and DES. In *Advances in Cryptology–Crypto '89*, 1989.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, 1989.

[Sim98] D. Simon. Finding collisions on a one-way street: can secure hash functions be based on general assumptions? In *Advances in Cryptology–Eurocrypt '98*, pages 334–345, 1998.

[WC81] M. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *J. of Computer and System Sciences*, 22:265–279, 1981.

Exposure-Resilient Functions and All-Or-Nothing Transforms

Ran Canetti[1], Yevgeniy Dodis[2], Shai Halevi[1], Eyal Kushilevitz[3], and Amit Sahai[2]

[1] IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598, USA. {canetti,shaih}@watson.ibm.com.

[2] Lab. for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02149, USA. {yevgen,amits}@theory.lcs.mit.edu.

[3] IBM T.J. Watson Research Center and Department of Computer Science, Technion, Haifa 32000, Israel. eyalk@cs.technion.ac.il.

Abstract. We study the problem of *partial key exposure*. Standard cryptographic definitions and constructions do not guarantee any security even if a tiny fraction of the secret key is compromised. We show how to build cryptographic primitives that remain secure even when an adversary is able to learn *almost all* of the secret key.

The key to our approach is a new primitive of independent interest, which we call an *Exposure-Resilient Function* (**ERF**) – a deterministic function whose output appears random (in a perfect, statistical or computational sense) even if *almost all* the bits of the input are known. **ERF**'s by themselves efficiently solve the partial key exposure problem in the setting where the secret is simply a random value, like in private-key cryptography. They can also be viewed as very secure pseudorandom generators, and have many other applications.

To solve the general partial key exposure problem, we use the (generalized) notion of an *All-Or-Nothing Transform* (**AONT**), an *invertible* (randomized) transformation T which, nevertheless, reveals "no information" about x even if *almost all* the bits of $T(x)$ are known. By applying an **AONT** to the secret key of any cryptographic system, we obtain security against partial key exposure. To date, the only known security analyses of **AONT** candidates were made in the random oracle model.

We show how to construct **ERF**'s and **AONT**'s with nearly optimal parameters. Our computational constructions are based on any one-way function. We also provide several applications and additional properties concerning these notions.

1 Introduction

A great deal of cryptography can be seen as finding ways to leverage the possession of a small but totally secret piece of knowledge (a key) into the ability to perform many useful and complex actions: from encryption and decryption to identification and message authentication. But what happens if our most *basic*

assumption breaks down — that is, if the secrecy of our key becomes partially compromised?

It has been noted that key exposure is one of the greatest threats to security in practice [1]. For example, at the Rump session of CRYPTO '98, van Someren [23] illustrated a breathtakingly simple attack by which keys stored in the memory of a computer could be identified and extracted, by looking for regions of memory with high entropy. Within weeks of the appearance of the followup paper [21], a new generation of computer viruses emerged that tried to use these ideas to steal secret keys [8]. Shamir and van Someren [21] gave some heuristic suggestions on preventing these kinds of attacks, but their methods still do not solve the problem of *partial exposure*.

Unfortunately, standard cryptographic definitions and constructions do not guarantee security *even if a tiny fraction of the secret key is exposed*. Indeed, many constructions become provably insecure (the simplest example would be "one-time pad" encryption), while the security of others becomes unclear. In this work, we show how to build cryptographic primitives, in the standard model (i.e., without random oracles) and using general computational assumptions, that remain provably secure even when the adversary is able to learn *almost all* of the secret key. Our techniques also have several applications in other settings.

Previous approaches and our goals. The most widely considered solutions to the problem of key exposure are distribution of keys across multiple servers via secret sharing [20, 3] and protection using specialized hardware. Distribution across many systems, however, is quite costly. Such an option may be available to large organizations, but is not realistic for the average user. Similarly, the use of specially protected hardware (such as smartcards) can also be costly, inconvenient, or inapplicable in many contexts.

Instead, we seek to enable a single user to protect itself against partial key exposure on a single machine. A natural idea would be to use a secret sharing scheme to split the key into shares, and then attempt to provide protection by storing these shares instead of storing the secret key directly. However, secret sharing schemes only guarantee security if the adversary misses at least one share *in its entirety*. Unfortunately, each share must be fairly large (about as long as the security parameter). Thus, in essence *we return to our original problem*: even if an adversary only learns a small fraction of all the bits, it could be that it learns a few bits from *each* of the shares, and hence the safety of the secret can no longer be guaranteed. We would like to do better. (Indeed, our techniques provide, for certain parameters, highly efficient computational secret sharing schemes [15], where the size of secret shares can be as small as *one bit*! See Remark 9 in Section 5.1.)

The All-Or-Nothing Transform. Recently Rivest [19], motivated by different security concerns arising in the context of block ciphers, introduced an intriguing primitive called the *All-Or-Nothing Transform (***AONT***)*. An **AONT**[1] is an efficiently computable transformation T on strings such that:

[1] Here we informally present a refinement of the definition due to Boyko [4].

- For any string x, given (*all* the bits of) $T(x)$, one can efficiently recover x.
- There exists some threshold ℓ such that any polynomial-time adversary that (adaptively) learns all but ℓ bits of $T(x)$ obtains "no information" about x.

The **AONT** solves the problem of partial key exposure: rather than storing a secret key directly, we store the **AONT** applied to the secret key. If we can build an **AONT** where the threshold value ℓ is very small compared to the size of the output of the **AONT**, we obtain security against almost total exposure. Notice that this methodology applies to secret keys with arbitrary structure, and thus protects all kinds of cryptographic systems. One can also consider more general **AONT**'s that have a two-part output: a public output that doesn't need to be protected (but is used for inversion), and a secret output that has the exposure-resilience property stated above. Such a notion would also provide the kind of protection we seek to achieve. As mentioned above, **AONT** has many other applications, such as enhancing the security of block-ciphers, hash functions and making fixed-blocksize encryption schemes more efficient (e.g., [14, 22]). For an excellent exposition on these and other applications of the **AONT**, see [4].

Our Results. Until now, the only known analysis of an **AONT** candidate was carried out by [4], who showed that Bellare and Rogaway's Optimal Asymmetric Encryption Padding (OAEP) [2] yields an **AONT** *in the Random Oracle model.* However, analysis in the Random Oracle model provides only a limited security guarantee for real-life schemes where the random oracle is replaced with an actual hash function [5].[2] In this work, we give the first constructions for **AONT**'s with essentially optimal resilience in the standard model, based only on computational assumptions.

The key to our approach and our main conceptual contribution is the notion of an *Exposure-Resilient Function* (**ERF**) — a deterministic function whose output appears random even if *almost all* the bits of the input are revealed. We believe this notion is useful and interesting in its own right. Consider for example an **ERF** with an output that is longer than its input — this can be seen a particularly strong kind of pseudorandom generator, where the generator's output remains pseudorandom *even if most of the seed is known.* **ERF**'s provide an alternative solution to **AONT** for the partial key exposure problem, since (at least, in principle) we can assume that our secret key is a truly random string R (say, the randomness used to generate the actual secret key). In such a case, we choose and store a random value r and use $f(r)$ (where f is an **ERF**) in place of R. In many settings (such as in private-key cryptography) this alternative is much more efficient than **AONT**. Another application of **ERF**'s is for protecting against gradual key exposure, where no bound on the amount of information the adversary obtains is assumed; instead, we assume only a bound on the *rate* at which that the adversary gains information.
Our main results regarding **ERF**'s and **AONT**'s are summarized as follows.

[2] Though for a much weaker definition, Stinson [25] has given an elegant construction for **AONT** with security analysis in the standard setting. As observed by [4], however, this construction does not achieve the kind of security considered here.

- We show how to construct, from any one-way function, for any $\epsilon > 0$, an **ERF** mapping an input of n bits to an output of *any size* polynomial in n, such that as long as *any* n^ϵ bits of the input remain unknown, the output will be pseudorandom.
- We build an *unconditionally* secure **ERF** whose output of size k is statistically close to uniform provided one misses only $\ell = k + o(k)$ bits of the input. This is optimal up to the lower order term, as no unconditionally secure **ERF**'s exist when $k < \ell$.
- Furthermore, we show that any computationally secure **ERF** with $k < \ell$ implies the existence of one-way functions.
- We give a simple construction of an **AONT** based on any **ERF**. For any $\epsilon > 0$, we show how to achieve a resilience threshold of $\ell = N^\epsilon$, where N is the size of the output of the **AONT**. If viewed as an **AONT** with separate public and secret outputs, then the size of the output of the **AONT** can be made optimal as well.
- We show that the existence of an **AONT** with $\ell < k - 1$, where k is the size of the input, implies the existence of one-way functions. We show that this result is tight up to a constant factor by constructing an unconditionally secure **AONT** with $\ell = \Theta(k)$ using no assumptions.
- We give another construction of an **AONT** based on any length-preserving function f such that both $[x \mapsto f(x)]$ and $[x \mapsto f(x) \oplus x]$ are **ERF**'s. This construction is similar to the OAEP, and so our analysis makes a step towards abstracting the properties of the random oracle needed to make the OAEP work as an **AONT**. It also has the advantage of meeting the standard definition of an **AONT** (without separate public and secret outputs) while retaining a relatively short output length.
- Finally, we show that a seemingly weaker "average-case" definition of **AONT** is equivalent to the standard "worst-case" definition of **AONT**, by giving an efficient transformation that achieves this goal.

Previous Work. Chor et al. [6] considered a notion called a t-resilient function, which are related to our notion of an Exposure-Resilient Function (**ERF**). A t-resilient function is a function whose output is *truly* random even if an adversary can fix any t of the inputs to the function. This turns out to be equivalent to the strongest formulation of unconditional security for an **ERF**. We give constructions for statistical unconditionally secure **ERF**'s that beat the impossibility results given in [6], by achieving an output distribution that is not truly random, but rather exponentially close in statistical deviation from truly random.

The concern of forward-security (or, protection from the *complete* exposure of past keys) was considered by Diffie et. al. [7] in the context of key exchange, and by Bellare and Miner [1] in the context of signature schemes. These works prevent an adversary that gains current secret keys from being able to decrypt past messages or forge signatures on messages "dated" in the past. In contrast, our work deals with providing security for both the future as well as the past, but assuming that not *all* of the secret key is compromised.

Organization. Section 2 briefly defines some preliminaries. Section 2.1 defines Exposure-Resilient Functions and All-Or-Nothing Transforms. Section 4 talks in detail about constructions and application of **ERF**'s, while Section 5 is concerned with constructing and examining the properties of **AONT**'s.

2 Preliminaries

For a randomized algorithm F and an input x, we denote by $F(x)$ the output distribution of F on x, and by $F(x;r)$ we denote the output string when using the randomness r. We write $m = poly(k)$ to indicate that m is polynomially bounded in k. In this paper we will not optimize certain constant factors which are not of conceptual importance. Unless otherwise specified, we will consider security against nonuniform adversaries.

Let $\{\binom{n}{\ell}\}$ denote the set of size-ℓ subsets of $[n] = \{1 \ldots n\}$. For $L \in \{\binom{n}{\ell}\}$, $y \in \{0,1\}^n$, let $[y]_{\bar{L}}$ denote y restricted to its $(n-\ell)$ bits *not* in L. We denote by $\oplus$ the bit-wise exclusive OR operator.

We recall that the *statistical difference* (also called *statistical distance*) between two random variables X and Y on a finite set D is defined to be

$$\max_{S \subseteq D} \left| \Pr[X \in S] - \Pr[Y \in S] \right| = \frac{1}{2} \cdot \sum_{\alpha} \left| \Pr[X = \alpha] - \Pr[Y = \alpha] \right|$$

Given two distributions A and B, we denote by $A \cong_c B$ ($A \cong_\epsilon B$, $A \equiv B$) the fact that they are computationally (statistically within ϵ, perfectly) indistinguishable (see, for instance, [9]). For the case of statistical closeness, we will always have ϵ negligible in the appropriate security parameter. When the statement can hold for any of the above choices (or the choice is clear from the context), we simply write $A \approx B$.

3 Definitions

In this section, we define the central concepts in our paper: Exposure-Resilient Functions (**ERF**'s) and All-Or-Nothing Transforms (**AONT**'s). An **ERF** is a function such that if its input is chosen at random, and an adversary learns *all* but ℓ bits of the input, for some threshold value ℓ, then the output of the function will still appear (pseudo) random to the adversary. Formally,

Definition 1. *A polynomial time computable function $f : \{0,1\}^n \to \{0,1\}^k$ is ℓ-***ERF** (exposure-resilient function) *if for any $L \in \{\binom{n}{\ell}\}$ and for a randomly chosen $r \in \{0,1\}^n$, $R \in \{0,1\}^k$, the following distributions are indistinguishable:*

$$\langle [r]_{\bar{L}}, f(r) \rangle \approx \langle [r]_{\bar{L}}, R \rangle \tag{1}$$

Here $\approx$ can refer to perfect, statistical or computational indistinguishability.

Remark 1. Note that this is a "non-adaptive" version of the definition. One may also consider an adaptive version of the definition, where the adversary may adaptively choose one-bit-at-a-time which $n - \ell$ positions of the input to examine. Owing only to the messiness of such a definition, we do not give a formal definition here, but we stress that all our constructions *satisfy this adaptive definition*, as well.

The definition states that an **ERF** transforms n random bits into k (pseudo) random bits, such that even learning all but ℓ bits of the input, leaves the output indistinguishable from a random value. There are several parameters of interest here: ℓ, n, and k. We see that the smaller ℓ is, the harder is to satisfy the condition above, since fewer bits are left unknown to the adversary. In general, there are two measures of interest: the fraction of ℓ with respect to n, which we would like to be as small as possible (this shows the "resilience"); and the size of k with respect to ℓ, which we want to be as large as possible (this shows how many pseudorandom bits we obtain compared to the number of random bits the adversary cannot see). We now define the notion of an **AONT**:

Definition 2. *A* randomized *polynomial time computable function* $T : \{0,1\}^k \to \{0,1\}^s \times \{0,1\}^p$ *is* ℓ-**AONT** (all-or-nothing transform) *if*

1. *T is efficiently invertible, i.e. there is a polynomial time machine I such that for any $x \in \{0,1\}^k$ and any $y = (y_1, y_2) \in T(x)$, we have $I(y) = x$.*
2. *For any $L \in \binom{s}{\ell}$, any $x_0, x_1 \in \{0,1\}^k$ we have*

$$\langle x_0, x_1,\ [T(x_0)]_{\bar{L}} \rangle \approx \langle x_0, x_1,\ [T(x_1)]_{\bar{L}} \rangle \tag{2}$$

In other words, the random variables in $\{[T(x)]_{\bar{L}} \mid x \in \{0,1\}^k\}$ are all indistinguishable from each other. Here $\approx$ can refer to perfect, statistical or computational indistinguishability.

If $T(x) = (y_1, y_2)$, we call y_1 the secret output *and y_2 the* public output *of T. If $p = 0$ (there is no public output), we call T a* secret-only ℓ-**AONT**.

Remark 2. Note again, as in Remark 1, that the definition given here is a "non-adaptive" definition. We stress that all our constructions satisfy the corresponding adaptive definition, as well.

Remark 3. The above definition is "indistinguishability" based. As usual, one can make the equivalent "semantic security" based definition, where the adversary, given $z = [T(x)]_{\bar{L}}$ (where x is picked according to some distribution M), cannot compute β satisfying some relation $\mathcal{R}(x, \beta)$ "significantly better" than without z at all. The proof of equivalence is standard and is omitted. Thus, the all-or-nothing transforms allow one to "encode" any x in such a form that the encoding is easily invertible, and yet, an adversary learning all but ℓ bits of the (secret part of the) encoding "cannot extract any useful information" about x.

Remark 4. The definition given above generalizes and simplifies (because there are no random oracles) the formal definition for secret-only **AONT** given by Boyko [4] (refining an earlier definition of Rivest [19]) in a setting with a random oracle. In particular, while previous definitions were restricted to *secret-only* **AONT**, our definition allows one to split the output y into two sections: a secret part y_1 and a public part y_2. The public part of the output requires *no protection* — that is, it is used only for inversion and can be revealed to the adversary in full. The security guarantee states that as long as ℓ bits of the *secret output* y_1 remain hidden (while all the bits of y_2 can be revealed), the adversary should have "no information" about the input. We note that our generalized notion of **AONT** solves the problem of partial key exposure and also remains equally applicable to all the other known uses of the secret-only **AONT**. However, we will see that it gives us more flexibility and also allows us to characterize the security of our constructions more precisely.

Boyko [4] showed that, *in the random oracle model*, the following so called "optimal asymmetric encryption padding" (OAEP) construction of [2] is a (secret-only) ℓ-**AONT** (where ℓ can be chosen to be logarithmic in the security parameter). Let $G : \{0,1\}^n \to \{0,1\}^k$ and $H : \{0,1\}^k \to \{0,1\}^n$ be random oracles (where n is any number greater than ℓ). The randomness of T is $r \leftarrow \{0,1\}^n$. Define $T(x;\ r) = \langle u, t\rangle$, where $u = G(r) \oplus x$, $t = H(u) \oplus r$. We note that the inverse $I(u,t) = G(H(u) \oplus t) \oplus u$. No constructions of **AONT** based on standard assumptions were previously known.

Remark 5. The notions of **ERF** and **AONT** are closely related with the following crucial difference. In an **ERF**, the "secret" is a (pseudo) random value $f(r)$. **ERF** allows one to represent this *random* secret in an "exposure-resilient" way by storing r instead. In **AONT**, the secret is an *arbitrary* x, which can be represented in an "exposure-resilient" way by storing $T(x)$ instead. Thus, **ERF** allows one to represent a *random* secret in an exposure-resilient way, while **AONT** allows this for *any* secret. We remark that **ERF**'s can be much more efficient that **AONT**'s for the case of (pseudo) random secrets; for example, in the computational setting we can store the value r that is *shorter* than the length of the actual secret $f(r)$, which is impossible to achieve with **AONT**'s due to their invertibility.

4 Exposure-Resilient Functions (ERF)

In this section we give constructions and some applications of exposure-resilient functions (**ERF**'s). First, we describe perfect **ERF**'s and their limitations. Then, on our way to building computational **ERF**'s with very strong parameters, we build statistical **ERF**'s, achieving essentially the best possible parameters and surpassing the impossibility results for perfect **ERF**'s. Finally, we show how to combine this construction with standard pseudorandom generators to construct computational **ERF**'s (from n to k bits) based on any one-way function that achieve any $\ell = \Omega(n^\epsilon)$ and any $k = poly(n)$ (in fact, we show that such

ERF's are equivalent to the existence of one-way functions). Our main results are summarized in the following theorem:

Theorem 1. *Assume $\ell \geq n^\epsilon$ (for some $\epsilon > 0$). Then*

1. *There exist statistical ℓ-**ERF**'s $f : \{0,1\}^n \to \{0,1\}^k$ with $k = \ell - o(\ell)$.*
2. *If $\ell < k \leq poly(n)$, computational ℓ-**ERF**'s $f : \{0,1\}^n \to \{0,1\}^k$ exist iff one-way functions exist.*

4.1 Perfect ERF

Here we require that $\langle [r]_{\bar{L}}, f(r)\rangle \equiv \langle [r]_{\bar{L}}, R\rangle$. Since the distributions are identical, this is equivalent to saying that no matter how one sets any $(n-\ell)$ bits of r (i.e. sets $[r]_{\bar{L}}$), as long as the remaining r bits are set at random, the output $f(r)$ is still perfectly uniform over $\{0,1\}^k$. This turns out to be exactly the notion of so called $(n-\ell)$-*resilient* functions considered in [6]. As an example, if $k = 1$, exclusive OR of n input bits is a trivial perfect 1-**ERF** (or a $(n-1)$-resilient function).

We observe that perfect ℓ-**ERF** can potentially exist only for $\ell \geq k$. Optimistically, we might expect to indeed achieve $\ell = O(k)$. However, already for $k = 2$ Chor et al [6] show that we must have $\ell \geq n/3$, i.e. at least third of the input should remain secret in order to get just 2 random bits! On the positive side, using *binary linear error correcting codes* (see [16]), one can construct the following perfect ℓ-**ERF**.

Theorem 2 ([6]). *Let M be a $k \times n$ matrix. Define $f(r) = M \cdot r$, where $r \in \{0,1\}^n$. Then f is perfect ℓ-**ERF** if and only if M is the generator matrix for a code of distance $d \geq n - \ell + 1$.*

Applying it to any *asymptotically good* (i.e. $n = O(k)$ and $d = \Omega(n)$) linear code (e.g. the Justesen code), we can get $\ell = (1-\epsilon)n$, $k = \delta n$, where ϵ and δ are (very small) constants.

Note that for any code, $k \leq n - d + 1$ (this is called the *Singleton bound*). Thus, we have $k \leq n - (n - \ell + 1) + 1 = \ell$, as expected. Also, it is known that $d \leq n/2$ for $k \geq 2 \log n$. This implies that we are limited to have $\ell \geq n/2$. However, at the expense of making $n = poly(k)$, using a Reed-Solomon code concatenated with a Hadamard code, we can achieve $\ell = n - d + 1$ to be arbitrarily close to $n/2$, but can never cross it.

4.2 Statistical ERF

We saw that perfect **ERF** cannot achieve $\ell < n/3$. Breaking this barrier will be crucial in achieving the level of security we ultimately desire from (computational) **ERF**'s. In this section, we show that by relaxing the requirement only slightly to allow negligible (in fact, *exponentially small*) statistical deviation, we are able to obtain **ERF**'s for essentially any value of ℓ (with respect to n) such that we obtain an output size $k = \Omega(\ell)$ (in fact, even $\ell - o(\ell)$). Note that this is

the best we can hope for (up to constant factors or even the lower order term), since it is not possible to have $k > \ell$ for any **ERF** with statistical deviation $\epsilon < \frac{1}{2}$ (proof is obvious, and omitted).

The key ingredient in our construction will be a combinatorial object called a *strong extractor*. An *extractor* is a family of hash functions $\mathcal{H}$ such that when a function h is chosen at random from $\mathcal{H}$, and is applied to a random variable X that has "enough randomness" in it, the resulting random variable $Y = h(X)$ is statistically close to the uniform distribution. In other words, by investing enough true randomness (namely, the amount needed to select a random member of $\mathcal{H}$), one can "extract" from X a distribution statistically close to the uniform distribution. A *strong* extractor has an extra property that Y is close to the uniform distribution *even when the random function h is revealed*. (Perhaps the best known example of a strong extractor is given in the Leftover Hash Lemma of [13], where standard 2-universal hash families are shown to be strong extractors.) Much work has been done in developing this area (e.g. [24, 26, 18]). In particular, it turns out that one can extract almost all the randomness in X by investing very few truly random bits (i.e. having small $\mathcal{H}$).

The intuition behind our construction is as follows. Notice that after the adversary observes $(n - \ell)$ bits of the input (no matter how it chose those bits), the input can still be any of the 2^ℓ completions of the input with equal probability. In other words, conditioned on any observation made by the adversary, the probability of any particular string being the input is at most $2^{-\ell}$. Thus, if we apply a sufficiently good extractor to the input, we have a chance to extract $\Omega(\ell)$ bits statistically close to uniform — exactly what we need. The problem is that we need some small amount of true randomness to select the hash function in the extractor family. However, if this randomness is small enough (say, at most $\ell/2$ bits), *we can take it from the input itself!* Hence, we view the first $\ell/2$ bits of r (which we will call u) as the randomness used to select the hash function h, and the rest of r we call v. The output of our function will be $h(v)$. Then observing $(n - \ell)$ bits of r leaves at least $2^{\ell/2}$ equally likely possible values of v (since $|u| = \ell/2$). Now, provided our extractor is good enough, we indeed obtain $\Omega(\ell)$ bits statistically close to uniform.

A few important remarks are in place before we give precise parameters. First, the adversary may choose to learn the entire u (i.e. it knows h). This is not a problem since we are using a *strong* extractor, *i.e.* the output is random even if one knows the true randomness used. Secondly, unlike the perfect **ERF** setting, where it was equivalent to let the adversary set $(n - \ell)$ input bits in any manner it wants, here the entire input (including u) *must* be chosen uniformly at random (and then possibly observed by the adversary).

Our most important requirement is that the hash function in the strong extractor family be describable by a very short random string. This requirement is met by the strong extractor of Srinivasan and Zuckerman [24] using the hash families of Naor and Naor [17]. Their results can be summarized as follows:

Lemma 1 ([24]). *For any ℓ and $t < \ell/2$, there exists a family $\mathcal{H}$ of hash functions mapping $\{0,1\}^n$ to a range $\{0,1\}^k$, where $k = \ell - 2t$, such that the following*

holds: A random member of $\mathcal{H}$ can be described by and efficiently computed using $4(\ell - t) + O(\log n)$ truly random bits (we will identify the hash function h with these random bits). Furthermore, for any distribution X on $\{0,1\}^n$ such that $\Pr[X = x] \leq 2^{-\ell}$ for all $x \in \{0,1\}^n$, we have that the statistical difference between the following two distributions is at most $\epsilon = 2 \cdot 2^{-t}$:

(A) Choose h uniformly from $\mathcal{H}$ and x according to X. Output $\langle h, h(x)\rangle$.

(B) Choose h uniformly from $\mathcal{H}$ and y uniformly from $\{0,1\}^k$. Output $\langle h, y\rangle$.

We are now ready to describe our statistical construction.

Theorem 3. *There exist statistical ℓ-**ERF** $f : \{0,1\}^n \to \{0,1\}^k$ with $k = \Omega(\ell)$ and statistical deviation $2^{-\Omega(\ell)}$, for any ℓ and n satisfying $\omega(\log n) \leq \ell \leq n$.*

Proof: Note that we will not optimize constant factors in this proof. Let $\ell' = \ell/5$ and $t = \ell/20$. We let the output size of our **ERF** be $k = \ell' - 2t = \ell/10$ and the statistical deviation be $\epsilon = 2 \cdot 2^{-t} = 2^{-\Omega(\ell)}$. Suppose the (random) input to our function is r. Now, we will consider the first $d = 4(\ell' - t) + O(\log n) < 4\ell/5$ bits of r to be h (here we use $\ell = \omega(\log n)$), which describes some hash function in $\mathcal{H}$ mapping $\{0,1\}^n$ to $\{0,1\}^k$ as given in Lemma 1. Let r' be r with the first d bits replaced by 0's. Note that r' is independent of h, and the length of r' is n. Define $f(r) = h(r')$.

We now analyze this function. Observe that for any $L \in \binom{n}{\ell}$, conditioned on the values of both $[r]_{\bar{L}}$ and h, there are still at least $\ell/5$ bit positions (among the last $n - d$ bit positions) of r that are unspecified. Hence, for all $L \in \binom{n}{\ell}$, for all $z \in \{0,1\}^{n-\ell}$, and for all $y \in \{0,1\}^n$, we have that

$$\Pr_r \left[r' = y \;\middle|\; L,\ [r]_{\bar{L}} = z \right] \leq 2^{-\ell/5} = 2^{-\ell'}.$$

Thus, by Lemma 1, we have that $\langle [r]_{\bar{L}}, h, f(r)\rangle = \langle [r]_{\bar{L}}, h, h(r')\rangle \cong_\epsilon \langle [r]_{\bar{L}}, h, R\rangle$, where R is the uniform distribution on $\{0,1\}^k$. This implies $\langle [r]_{\bar{L}}, f(r)\rangle \cong_\epsilon \langle [r]_{\bar{L}}, R\rangle$, completing the proof. □

We make a few remarks about the security of this construction:

Remark 6. Note that, in particular, we can choose ℓ to be anything super-logarithmic is n (e.g., n^ϵ for any $\epsilon > 0$), providing excellent security against partial key exposure. Seen another way, we can choose n to be essentially any size larger than ℓ.

Remark 7. The output size of our construction can be substantially improved by using recent strong extractors of [18]. In particular, we can achieve $k = \ell - o(\ell)$, provided $\ell = \omega(\log^3 n)$, or $k = (1 - \delta)\ell$ (for any $\delta > 0$), provided $\ell = \omega(\log^2 n)$. In both cases the statistical deviation can be made exponentially small in ℓ. As k must be less than ℓ, this is essentially optimal.

4.3 Computational ERF

The only limiting factor of the statistical construction is that the output size is limitted to $k < \ell$. By finally relaxing our requirement to *computational* security,

we are able to achieve an arbitrary output size, by using a pseudorandom generator (**PRG**) as the final outermost layer of our construction. We also show that any **ERF** with $k > \ell$ implies the existence of **PRG**'s (and thus, one-way functions), closing the loop. The proof of the following is straightforward, and therefore omitted:

Lemma 2. *Let $m, n = poly(k)$, $f : \{0,1\}^n \to \{0,1\}^k$ be a statistical ℓ-**ERF** (with negligible ϵ) and $G : \{0,1\}^k \to \{0,1\}^m$ be a **PRG**. Then $g : \{0,1\}^n \to \{0,1\}^m$ mapping $x \mapsto G(f(x))$ is a computational ℓ-**ERF**.*

Theorem 4. *Assume one-way functions exist. Then for any ℓ, any $n = poly(\ell)$ and $k = poly(n)$, there exists a computational ℓ-**ERF** mapping $\{0,1\}^n$ to $\{0,1\}^k$.*

Proof: Since $k = poly(\ell)$, one-way functions imply [12] the existence of a **PRG** $G : \{0,1\}^{\ell/10} \to \{0,1\}^k$. Theorem 3 implies the existence of a statistical ℓ-**ERF** f from $\{0,1\}^n$ to $\{0,1\}^{\ell/10}$ with negligible statistical deviation $2^{-\Omega(\ell)}$. By Lemma 2, $g(r) = G(f(r))$ is the desired computational ℓ-**ERF**. □

Lemma 3. *If there exists an ℓ-**ERF** f mapping $\{0,1\}^n$ to $\{0,1\}^k$, for $k > \ell$ (for infinitely many different values of ℓ, n, k), then one-way functions exist.*

Proof: The hypothesis implies the existence of the ensemble of distributions $A = \langle [r]_{\bar{L}}, f(r) \rangle$ and $B = \langle [r]_{\bar{L}}, R \rangle$, where R is uniform on $\{0,1\}^k$. By assumption, A and B are computationally indistinguishable ensembles. Note that A can have at most n bits of entropy (since the only source of randomness is r), while B has $n - \ell + k \geq n + 1$ bits of entropy. Thus, the statistical difference between A and B is at least $1/2$. By the result of Goldreich [10], the existence of a pair of efficiently samplable distributions that are computationally indistinguishable but statistically far apart, implies the existence of pseudorandom generators, and hence one-way functions. □

Theorem 1 now follows from Remark 7, Theorem 4 and Lemma 3.

4.4 Applications of ERF

As we said, ℓ-**ERF** $f : \{0,1\}^n \to \{0,1\}^k$ allows one to represent a random secret in an "exposure-resilient" way. In Section 5 we show how to construct **AONT**'s using **ERF**'s. Here we give some other applications.

As an immediate application, especially when $k > n$, it allows us to obtain a much stronger form of pseudorandom generator, which not only stretches n bits to k bits, but remains pseudorandom when any $(n - \ell)$ bits of the seed are revealed. As a natural extension of the above application, we can apply it to private-key cryptography. A classical one-time private-key encryption scheme over $\{0,1\}^k$ chooses a random shared secret key $r \in \{0,1\}^n$ and encrypts $x \in \{0,1\}^k$ by the pseudorandom "one-time pad" $G(r)$ (where G is a **PRG**), i.e. $E(x;\ r) = x \oplus G(r)$. We can make it resilient to the partial key exposure by replacing **PRG** G with **ERF** f.

For the next applications, we assume for convenience that **ERF** $f : \{0,1\}^k \to \{0,1\}^k$ is length-preserving. Using such f, we show how to obtain *exposure-resilient form of a pseudorandom function family* (**PRF**) [11]. Let $\mathcal{F} = \{F_s \mid s \in \{0,1\}^k\}$ be a regular **PRF** family. Defining $\tilde{F}_s = F_{f(s)}$, we get a new pseudorandom function family $\tilde{\mathcal{F}} = \{\tilde{F}_s \mid s \in \{0,1\}^k\}$, which remains pseudorandom even when all but ℓ bits of the seed s are known. We apply this again to private-key cryptography. The classical private-key encryption scheme selects a random shared key $s \in \{0,1\}^k$ and encrypts x by a pair $\langle x \oplus F_s(R), R\rangle$, where R is chosen at random. Again, replacing $\mathcal{F}$ by an exposure-resilient **PRF**, we obtain resilience against partial key exposure. Here our secret key is $s \in \{0,1\}^k$, but $f(s)$ is used as an index to a regular **PRF**.

In fact, we can achieve security even against what we call the *gradual key exposure* problem in the setting with shared random keys. Namely, consider a situation where the adversary is able to learn more and more bits of the secret key over time. We do not place any upper bound on the *amount* of information the adversary learns, but instead assume only that the *rate* at which the adversary can gain information is bounded. For example, suppose that every week the adversary somehow learns at most b bits of our secret r. We know that as long as the adversary misses ℓ bits of r, the system is secure[3]. To avoid ever changing the secret key, both parties periodically (say, with period slightly less than $(k-\ell)/b$ weeks) update their key by setting $r_{new} = f(r_{old})$. Since at the time of each update the adversary missed at least ℓ bits of our current key r, the value $f(r)$ is still pseudorandom, and thus secure. Hence, parties agree on the secret key *only once*, even if the adversary continuously learns more and more of the (current) secret!

5 All-Or-Nothing Transform (AONT)

As we pointed out, no **AONT** constructions with analysis outside the random oracle model were known. We give several such constructions. One of our constructions implies that for the interesting settings of parameters, the existence of ℓ-**AONT**'s, ℓ-**ERF**'s and one-way functions are equivalent. The other construction can be viewed as the special case of the OAEP construction of Bellare and Rogaway [2]. Thus, our result can be viewed as the first step towards abstracting the properties of the random oracle that suffice for this construction to work. Finally, we give a "worst-case/average-case" reduction for **AONT**'s that shows it suffices to design **AONT**'s that are secure only for *random* x_0, x_1.

5.1 Simple Construction using ERF

We view the process of creating ℓ-**AONT** as that of *one-time private-key encryption*, similarly to the application in Section 4.4. Namely, we look at the simplest

[3] We assume that our **ERF** is secure against *adaptive* key exposure, but our construction achieves this.

possible one-time private-key encryption scheme — the one-time pad, which is unconditionally secure. Here the secret key is a random string R of length k, and the encryption of $x \in \{0,1\}^k$ is just $x \oplus R$. We simply replace R by $f(r)$ where f is ℓ-**ERF** and r is our new secret. Thus, we obtain the following theorem, whose proof is omitted due to space constraints:

Theorem 5. *Let $f : \{0,1\}^n \to \{0,1\}^k$ be computational (statistical, perfect) ℓ-**ERF**. Define $T : \{0,1\}^k \to \{0,1\}^n \times \{0,1\}^k$ (that uses n random bits r) as follows: $T(x;\ r) = \langle r, f(r) \oplus x \rangle$. Then T is computational (statistical, perfect) ℓ-**AONT** with secret part r and public part $f(r) \oplus x$.*

Notice that the size of the secret part $s = n$ and size of the public part $p = k$. As an immediate corollary of Theorems 1 and 5, we have:

Theorem 6. *Assume $\ell \leq s \leq poly(\ell)$. There exist functions $T : \{0,1\}^k \to \{0,1\}^s \times \{0,1\}^k$ (with secret output of length s and public output of length k) such that*

1. *T is statistical ℓ-**AONT** with $k = \ell - o(\ell)$, or*
2. *T is computational ℓ-**AONT** with $\ell < k \leq poly(s)$.*

For example, we could set $\ell = s^\epsilon$ to have excellent exposure-resilience. The computational construction also allows us to have essentially any input size k we want (as long as it is polynomial in s), and have the total output size $N = s+k$ be dominated by k, which is close to optimal. A reasonable setting seems to be $s = o(k)$ (i.e., just slightly smaller than k) and $\ell = s^\epsilon$.

Remark 8. Observe that any ℓ-**AONT** with public and secret outputs of length p and s, respectively, also gives a *secret-only* ℓ'-**AONT** with output size $N = s+p$ and $\ell' = \ell + p$ (since if the adversary misses $\ell + p$ bits of the output, it must miss at least ℓ bits of the secret output). Applying this to our construction (where $p = k$), we see that $\ell' = \ell + k$ and we can achieve essentially any $N > \ell'$. In particular, we can still have excellent exposure-resilience $\ell' = N^\epsilon$, but now the output size $N = (\ell')^{1/\epsilon} > k^{1/\epsilon}$ is large compared to the input length k. See Section 5.3 for a possible solution to this problem. We also notice that we can have $\ell' = 2k + o(k) = O(k)$ (and essentially any N) even in the statistical setting.

Remark 9. Consider an ℓ-**AONT** with public output of size p and secret output of size s. We can interpret this as being a kind of "gap" *computational secret sharing scheme* [15]. For some secret x, we apply the **AONT** to obtain a secret output y_1 and public output y_2. Here, we think of y_2 as being a public share that is unprotected. We interpret the bits of y_2 as being tiny shares that are *only* 1 *bit long*, with one share given to each of s parties. We are guaranteed that if all the players cooperate, by the invertability of the **AONT**, they can recover the secret x. On the other hand, if $s - \ell$ or fewer of the players collude, they gain no computational information about the secret whatsoever. We call this a "gap" secret sharing scheme because there is a gap between the number of players needed to reconstruct the secret and the number of players that cannot

gain any information. Note that such a gap is unavoidable when the shares are smaller than the security parameter. Using our constructions, we can obtain such schemes for any value of ℓ larger than the security parameter, and essentially any value of s larger than ℓ (plus essentially any length k of the secret).

5.2 AONT implies OWFs

Theorem 7. *Assume we have a computational ℓ-***AONT** $T : \{0,1\}^k \to \{0,1\}^s \times \{0,1\}^p$ *where $\ell < k-1$. Then one-way functions exist.*

Proof: To show that **OWF**'s exist it is sufficient to show that *weak* **OWF**'s exist [9]. Fix $L = [\ell] \subseteq [s]$. Define $g(x_0, x_1, b, r) = \langle x_0, x_1, [y]_{\bar{L}} \rangle$, where $y = T(x_b;\ r)$. We claim that g is a weak **OWF**. Assume not. Then there is an inverter A such that when x_0, x_1, b, r are chosen at random, $y = T(x_b;\ r)$, $z = [y]_{\bar{L}}$, $\langle \tilde{b}, \tilde{r} \rangle = A(x_0, x_1, z)$, $\tilde{y} = T(x_{\tilde{b}};\ \tilde{r})$, $\tilde{z} = [\tilde{y}]_{\bar{L}}$, we have $\Pr(z = \tilde{z}) > \frac{3}{4}$.

To show that there exist x_0, x_1 breaking the indistinguishability property of T, we construct a distinguisher F for T that has non-negligible advantage for *random* $x_0, x_1 \in \{0,1\}^k$. Hence, the job of F is the following. x_0, x_1, b, r are chosen at random, and we set $y = T(x_b;\ r)$, $z = [y]_{\bar{L}}$. Then F is given the challenge z together with x_0 and x_1. Now, F has to predict b correctly with probability non-negligibly more than $1/2$. We let F run $A(x_0, x_1, z)$ to get $\tilde{b}, \tilde{r}$. Now, F sets $\tilde{y} = T(x_{\tilde{b}};\ \tilde{r})$, $\tilde{z} = [\tilde{y}]_{\bar{L}}$. If indeed $\tilde{z} = z$ (i.e. A succedeed), F outputs $\tilde{b}$ as its guess, else it flips a coin.

Let B be the event that A succeeds inverting. From the way we set up the experiment, we know that $\Pr(B) \geq \frac{3}{4}$. Call U the event that when x_0, x_1, b, r are chosen at random, $[T(x_b;\ r)]_{\bar{L}} \in [T(x_{1-b})]_{\bar{L}}$, i.e. there exists some r' such that $[T(x_{1-b};\ r')]_{\bar{L}} = z$ or $g(x_0, x_1, 1-b, r') = g(x_0, x_1, b, r)$. If U does not happen and A succeeded inverting, we know that $\tilde{b} = b$, as it is $1-b$ is an impossible answer. Thus, using $\Pr(X \wedge \overline{Y}) \geq \Pr(X) - \Pr(Y)$, we get:

$$\begin{aligned}\Pr(\tilde{b} = b) &\geq \frac{1}{2}\Pr(\overline{B}) + \Pr(B \wedge \overline{U}) \geq \frac{1}{2}\Pr(\overline{B}) + \Pr(B) - \Pr(U) \\ &= \frac{1}{2} + \frac{1}{2}\Pr(B) - \Pr(U) \geq \frac{1}{2} + \left(\frac{3}{8} - \Pr(U)\right)\end{aligned}$$

To get a contradiction, we show that $\Pr(U) \leq 2^{\ell-k}$, which is at most $\frac{1}{4} < \frac{3}{8}$ since $\ell < k-1$. To show this, observe that U measures the probability of the event that when we choose x, x', r at random and set $z = [T(x;\ r)]_{\bar{L}}$, there is r' such that $z = [T(x';\ r')]_{\bar{L}}$. However, for any fixed setting of z, there are only 2^ℓ possible completions $y \in \{0,1\}^{s+p}$. And for each such completion y, invertibility of T implies that there could be at most one $x' \in T^{-1}(y)$. Hence, for any setting of z, at most 2^ℓ out of 2^k possible x' have a chance to have the corresponding r'. Thus, $\Pr(U) \leq 2^{\ell-k}$ indeed. □

We note that the result is essentially optimal (up to the lower order term), since by Theorem 6 there are statistical **AONT**'s with $\ell = k + o(k)$. In fact, merging the secret and public parts of such an ℓ-**AONT** (the latter having length k) gives a statistical *secret-only* ℓ'-**AONT** with $\ell' = \ell + k = O(k)$ still.

5.3 Towards secret-only AONT

We also give another construction of an **AONT** based on any length-preserving function f such that both $[r \mapsto f(r)]$ and $[r \mapsto f(r) \oplus r]$ are **ERF**'s. The construction has the advantage of achieving *secret-only* **AONT**'s, while retaining a relatively short output length.

Recall that the OAEP construction of [2] sets $T(x;\ r) = \langle u, t\rangle$, where $u = G(r) \oplus x$, $t = H(u) \oplus r$, and $G : \{0,1\}^n \to \{0,1\}^k$ and $H : \{0,1\}^k \to \{0,1\}^n$ are some functions (e.g., random oracles). We analyze the following construction, which is a special case of the OAEP construction with $n = k$, and H being the identity function. Let $f : \{0,1\}^k \to \{0,1\}^k$, define $T(x;\ r) = \langle f(r) \oplus x, (f(r) \oplus r) \oplus x\rangle$, and note that the inverse is $I(u,t) = u \oplus f(u \oplus t)$. Due to space limitations, we omit the proof of the following:

Theorem 8. *Assume f is such that both $f(r)$ and $(f(r) \oplus r)$ are length-preserving computational ℓ-***ERF***s. Then T above is computational secret-only 2ℓ-***AONT***.*

We note, that random oracle f clearly satisfies the conditions of the Theorem. Thus, our analysis makes a step towards abstracting the properties of the random oracle needed to make the OAEP work as an **AONT**. We believe that the assumption of the theorem is quite reasonable, even though leave open the question of constructing such f based on standard assumptions.

5.4 Worst-case / Average-case Equivalence of AONT

In the definition of **AONT** we require that Equation (2) holds for any x_0, x_1. This implies (and is equivalent) to saying that it holds if one is to choose x_0, x_1 according to any distribution $q(x_0, x_1)$. A natural such distribution is the uniform distribution, which selects random x_0, x_1 uniformly and independently from $\{0,1\}^k$. We call an **AONT** secure against (possibly only) the uniform distribution an *average-case* **AONT**.[4] A natural question to ask is whether average-case **AONT** implies (regular) **AONT** with comparable parameters, which can be viewed as the worst-case/average case equivalence. We show that up to a constant factor, the notions are indeed identical in the statistical or computational settings. Below we assume without loss of generality that our domain is a finite field (e.g. $GF(2^k)$), so that addition and multiplication are defined. We omit the proof of the following due to space constraints:

Lemma 4. *Let $T : \{0,1\}^k \to \{0,1\}^s \times \{0,1\}^p$ be an average-case (statistical or computational) ℓ-***AONT***. Then the following $T' : \{0,1\}^k \to \{0,1\}^{4s} \times \{0,1\}^{4p}$ is a (statistical or computational) 4ℓ-***AONT***, where a_1, a_2, b are chosen uniformly at random subject to $a_1 + a_2 \neq 0$ (as part of the randomness of T'):*

$$T'(x) = \langle T(a_1), T(a_2), T(b), T((a_1 + a_2) \cdot x + b)\rangle$$

In the above output, we separately concatenate secret and public outputs of T. In particular, if T is secret-only, then so is T'.

[4] Note, for instance, the proof of Theorem 7 works for average-case **AONT**'s as well.

6 Conclusions

We have studied the problem of partial key exposure and related questions. We have proposed solutions to these problems based on new constructions of the All-Or-Nothing Transform in the standard model (without random oracles).

The key ingredient in our approach is an interesting new primitive which we call an Exposure-Resilient Function. This primitive has natural applications in combatting key exposure, and we believe it is also interesting in its own right. We showed how to build essentially optimal **ERF**'s and **AONT**'s (in the computational setting, based on any one-way function). We also explored many other interesting properties of **ERF**'s and **AONT**'s.

Acknowledgements. We would like to thank Madhu Sudan for several helpful discussions. Much of this work was performed while all authors were at the IBM T.J. Watson Research Center. Amit Sahai's research was also supported in part by a DOD NDSEG Fellowship.

References

1. M. Bellare, S. Miner. A Forward-Secure Digital Signature Scheme. In *Proc. of Crypto*, pp. 431–448, 1999.
2. M. Bellare, P. Rogaway. Optimal Asymetric Encryption. In *Proc. of EuroCrypt*, pp. 92–111, 1995.
3. G. Blackley. Safeguarding Cryptographic Keys. In *Proc. of AFIPS 1979 National Computer Conference*, 1979.
4. V. Boyko. On the Security Properties of the OAEP as an All-or-Nothing Transform. In *Proc. of Crypto*, pp. 503–518, 1999.
5. R. Canetti, O. Goldreich and S. Halevi. The Random-Oracle Model, Revisited. In *Proc. of STOC*, pp. 209–218, 1998.
6. B. Chor, J. Friedman, O. Goldreich, J. Hastad, S. Rudich, R. Smolensky. The Bit Extraction Problem or t-resilient Functions. In *Proc. of FOCS*, pp. 396–407, 1985.
7. W. Diffie, P. van Oorschot and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.
8. A. Dornan. New Viruses Search For Strong Encryption Keys. In *PlanetIT Systems Management News*, `http://www.planetit.com/techcenters/docs/systems_management/news/PIT19990317S0015`, March, 1999.
9. O. Goldreich. Foundations of Cryptography (Fragments of a Book). Available at `http://www.wisdom.weizmann.ac.il/home/oded/public_html/frag.html`
10. O. Goldreich. A Note on Computational Indistinguishability. In *IPL*, 34:277–281, 1990.
11. O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):210–217, 1986.
12. J. Hastad, R. Impagliazzo, L. Levin, M. Luby. A Pseudorandom generator from any one-way function. In *Proc. of STOC*, 1989.
13. R. Impagliazzo, L. Levin, M. Luby. Pseudorandom Generation from one-way functions. In *Proc. of STOC*, pp. 12–24, 1989.
14. M. Jakobsson, J. Stern, M. Yung. Scramble All, Encrypt Small. In *Proc. of Fast Software Encryption*, pp. 95–111, 1999.
15. H. Krawczyk. Secret Sharing Made Short. In *Proc. of Crypto*, pp. 136–146, 1993.

16. F. MacWilliams, J. Sloane. Theory of Error-Correcting Codes, Amsterdam, 1981.
17. J. Naor, M. Naor. Small-Bias Probability Spaces: Efficient Constructions and Applications. In *SIAM J. Computing*, 22(4):838-856, 1993.
18. R. Raz, O. Reingold, S. Vadhan. Error Reduction for Extractors. In Proc. of FOCS, pp. 191–201, 1999.
19. R. Rivest. All-or-Nothing Encryption and the Package Transform. In *Fast Software Encryption, LNCS*, 1267:210–218, 1997.
20. A. Shamir. How to share a secret. In *Communic. of the ACM*, 22:612-613, 1979.
21. A. Shamir, N. van Someren. Playing "hide and seek" with stored keys. In *Proc. of Financial Cryptography*, 1999.
22. S. U. Shin, K. H. Rhee. Hash functions and the MAC using all-or-nothing property. In *Proc. of Public Key Cryptography, LNCS*, 1560:263–275, 1999.
23. N. van Someren. How not to authenticate code. Crypto'98 Rump Session, Santa Barbara, 1998.
24. A. Srinivasan, D. Zuckerman. Computing with Very Weak Random Sources. In *Proc. of FOCS*, pp. 264–275, 1994.
25. D. Stinson. Something About All or Nothing (Transforms). Available from `http://cacr.math.uwaterloo.ca/~dstinson/papers/AON.ps`, 1999.
26. L. Trevisan. Construction of Extractors Using PseudoRandom Generators. In Proc. of STOC, pp. 141–148, 1999.

The Sum of PRPs is a Secure PRF

Stefan Lucks*

Theoretische Informatik, Universität Mannheim
68131 Mannheim, Germany
lucks@th.informatik.uni-mannheim.de

Abstract. Given d independent pseudorandom permutations (PRPs) $\pi_i, \ldots, \pi_d$ over $\{0,1\}^n$, it appears natural to define a pseudorandom function (PRF) by adding (or XORing) the permutation results: $\text{SUM}^d(x) = \pi_1(x) \oplus \cdots \oplus \pi_d(x)$. This paper investigates the security of SUM^d and also considers a variant that only uses one single PRP over $\{0,1\}^n$.

1 Introduction

Cryptography requires an encryption function to be invertible: Someone knowing the (secret) key must be able to recover the plaintext from the ciphertext. Accordingly, under a fixed key, a n-bit block cipher is a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$. The classical security requirement is that π must behave pseudorandomly, i.e. must be un-distinguishable from a random permutation over $\{0,1\}^n$ without knowing the secret key.

In practice, block ciphers are used in many different modes of operations, and not all of them need an invertible cipher. Sometimes, being invertible can even *hinder* the security of schemes using the cipher. One such example is the "cipher block chaining" (CBC) mode, a standard mode of operation for block ciphers: if more than about $2^{n/2}$ blocks are encrypted, the ciphertext leaks information about the plaintext [2]. So why not simply use a dedicated pseudorandom function (PRF) instead of a pseudorandom permutation (PRP) in such cases? Two reasons are:

- Applications may need both invertible ciphers *and* schemes where the cipher better would not be invertible. Double-using one primitive to implement both is less expensive in terms of memory or chip space.
- There exist quite a lot of "proven" block ciphers, i.e., block ciphers published years ago, intensively cryptanalysed and widely trusted today. There are not as many good candidates for dedicated PRFs.

Hence, instead of constructing pseudorandom functions from scratch, we consider creating them using pseudorandom permutations as underlying building

* Supported by DFG grant Kr 1521/3-1.

blocks. Recently, the question of how to do this has caught the attention of the cryptographic community [5, 7]. Let $\pi_1, \ldots, \pi_d$ denote random permutations over $\{0,1\}^n$ and $\oplus$ the bit-wise XOR. Bellare, Krovetz and Rogaway [5] point out that the construction $\text{SUM}^2(x) = \pi_1(x) \oplus \pi_2(x)$ has not (yet) been analysed. In the current paper, we generalise this and analyse $\text{SUM}^d : \{0,1\}^n \to \{0,1\}^n$ with $\text{SUM}^d(x) = \pi_1(x) \oplus \cdots \oplus \pi_d(x)$.

Organisation of this Paper:
Section 2 and Section 3 present the notation and the basic definitions we use in this paper and describe some previous research. Section 4 describes the security of the PRF $\text{SUM}^d(x) = \bigoplus_{1 \le i \le d} \pi_d(x)$. In the following section, we analyse the variant $\text{TWIN}^d : \{0,1\}^{n - \lceil \log_2(d) \rceil} \to \{0,1\}^n$ with $\text{TWIN}^d(x) = \pi(dx) \oplus \cdots \oplus \pi(dx + d - 1)$. Section 6 provides some comments and conclusions. For better tangibility, the appendix considers the two-dimensional special case SUM^2.

2 Preliminaries

We write $\mathbb{F}_{m,n}$ for the set of all functions $\{0,1\}^m \to \{0,1\}^n$ and $\mathbb{F}_n = \mathbb{F}_{n,n}$. For choosing a random value x, uniformly distributed in a set M, we write $x \in_R M$. A random function $\psi \in \mathbb{F}_{m,n}$ is a function $\psi \in_R \mathbb{F}_{m,n}$. If S_n is the set of all permutations in $\mathbb{F}_n$, a random permutation over $\{0,1\}^n$ is a function $\pi \in_R S_n$.

To measure the "pseudorandomness" of a function $f \in \mathbb{F}_{m,n}$, chosen "somehow randomly" but in general not uniformly distributed, we consider an adversary A trying to distinguish between f and a random function $R \in_R \mathbb{F}_{m,n}$. A has access to an oracle Q. A chooses inputs $x \in \{0,1\}^n$; Q responds $Q(x) \in \{0,1\}^n$. Q either simulates $R \in_R \mathbb{F}_{m,n}$, or f. A's output is $A(Q) \in \{0,1\}$. We view A as a probabilistic algorithm, hence the output $A(Q)$ is a random variable over $\{0,1\}$. $A(Q)$ depends on the random choice of f and the internal coin flips of A and Q. We evaluate the (unsigned) difference of the probabilities $\text{pr}[A(Q) = 1]$ for $Q = R$ and $Q = f$, i.e. A's "PRF advantage" $\text{Adv}^{\text{Fun}}_{A,f}$ with respect to f:

$$\text{Adv}^{\text{Fun}}_{A,f} = |\text{pr}[A(R) = 1] - \text{pr}[A(f) = 1]|.$$

A's "PRP advantage" $\text{Adv}^{\text{Perm}}_{A,\pi}$ is defined similarly. Here, the oracle Q simulates a random permutation $P \in_R S_n$ and $\pi \in S_n$.

$$\text{Adv}^{\text{Perm}}_{A,\pi} = |\text{pr}[A(P) = 1] - \text{pr}[A(\pi) = 1]|.$$

Definition 1. *A function $f \in \mathbb{F}_{m,n}$ is a (q,a)-secure PRF, if all adversaries A asking at most q oracle queries are restricted to $\text{Adv}^{\text{Fun}}_{A,f} \le a$. Similarly, we define a (q,a)-secure PRP π: $\text{Adv}^{\text{Perm}}_{A,\pi} \le a$.*

Note that "ideal" schemes are $(\infty, 0)$-secure: a random function is a $(\infty, 0)$-secure PRF, and a random permutation is a $(\infty, 0)$-secure PRP.

The notion of "(q,a)-security" is very strong, since the adversaries' running time is not limited. By simply searching the key-space, one could easily distinguish a block cipher from a random permutation. We claim that one can

approximatively describe a practically secure block cipher under a random key as an $(\infty, 0)$-secure PRP, see Section 6.1.

We interchangeably view b-bit strings $s = (s_{b-1}, \ldots, s_0) \in \{0,1\}^b$ as b-bit numbers $s = \sum_{0 \le i < b} s_i * 2^i$.

3 Previous Work

3.1 Using a PRP as PRF

It is widely known that a random permutation over $\{0,1\}^n$ is a $(q, q^2/2^n)$-secure PRF. Since it nicely fits to our later results, we formalise this here:

Theorem 1. *The random permutation $\pi \in \mathbb{F}_n$ is a (q,a)-secure PRF with $a = q^2/2^{n+1}$. An adversary A^* exists to distinguish π from a random function with an advantage of* $\mathrm{Adv}^{\mathrm{Fun}}_{A^*,\pi} = \theta(q^2/2^n)$.

Proof: [Sketch] If by chance a random function R behaves like a permutation, i.e., for all q pairs $(x_1, R(x_1))$ no collision $R(x_i) = R(x_j)$ with $x_i \neq x_j$ occurs, then no adversary can distinguish between R and a random permutation. On the other hand, any collision proves that R is no permutation. With q inputs, the probability to get a collision is $2^{-n} \sum_{1 \le i < q} i \le q^2/2^{n+1}$. □

Theorem 1 justifies to use a block cipher (i.e. a PRP) as a PRF – if the famous *birthday bound* $q \ll 2^{n/2}$ is observed. What about $q > 2^{n/2}$? Note that the function $f^{\oplus}$ with $f^{\oplus}(x) = \pi(x) \oplus x$ is unlikely to be invertible, but is not a better PRF since $\pi(x) = f^{\oplus}(x) \oplus x$ [7, 5].

3.2 Using simple operations and PRFs as Building Blocks

Much research dealt with constructing complex cryptographic operations from (seemingly) simple ones: Levin [8] constructed "pseudorandom bit generators" from "one-way functions", Goldreich, Goldwasser, and Micali [6] constructed PRFs from "pseudorandom bit generators", and Luby and Rackoff [9] constructed PRPs from PRFs. A lot of work has been done on improvements of the Luby-Rackoff construction, some recent examples are [10–12]. Now we are going into the opposite direction: We construct PRFs from PRPs.

Another direction of cryptographic research was how to construct PRFs from smaller PRFs. Aiello and Venkatesan [1] presented a construction for PRFs over $\{0,1\}^{2n}$ using PRFs over $\{0,1\}^n$ as building blocks.

3.3 Constructing a PRF from PRPs

"**Data dependent re-keying**" was proposed by Bellare, Krovetz, and Rogaway [5]. Here, a block cipher E with k-bit keys is a family of 2^k independent random permutations. Set $j := \lceil k/n \rceil$. For keys $K_1, \ldots, K_j \in \{0,1\}^k$, the function $f^{\mathrm{BKR}}_{K_1,\ldots,K_j}$ maps $x \in \{0,1\}^n$ to $f^{\mathrm{BKR}}_{K_1,\ldots,K_j}(x) \in \{0,1\}^n$ by the following algorithm:

K' $:= E_{K_1}(x)||\cdots||E_{K_j}(x)$; (* Concatenate the values $E_{k_i}(x)$. *)
K'' $:= K' \bmod 2^k$; (* We only need k of $nj \geq k$ bits. *)
$f^{\mathrm{BKR}}_{K_1,\ldots,K_j}(x) := E_{K''}(x)$; (* Use the derived key K'' to encrypt the input. *)

In a formal model, data dependent re-keying is provably more secure than simply using one PRP as a PRF [5]. The model is based on the adversary having access to the block cipher E by asking additional oracle queries: choose keys $K \in \{0,1\}^k$ and texts $T \in \{0,1\}^n$ and ask the oracle for $E_K(T)$ and $E^{-1}(T)$. [5, Theorem 5.2] indicates that $f^{\mathrm{BKR}}_{K_1,\ldots,K_j}$ is a (t,q,a)-secure PRF with $a \approx 0$ if $t \ll \min\{2^{4k/5}, 2^n\}$ and $q \ll \min\{2^{4k/5}, 2^{(n+k)/2}\}$. A variation of this scheme speeds up counter mode encryption: For a small constant d, the same K'' is used for 2^d steps.

Hall et. al. [7] examine two constructions. Let $d \in \{0,\ldots,n\}$ and π be a PRP over $\{0,1\}^n$. The "**truncate**" construction is defined by $f^{\mathrm{tr}}_d : \{0,1\}^n \to \{0,1\}^{n-d}$ by $f^{\mathrm{tr}}_d(x) = \pi(x) \operatorname{div} 2^d$. The PRF f^{tr}_d is provably secure if $q \ll \min\{2^{(n+d)/2}, 2^{2(n-d)/3}\}$ [7], i.e. if $q \ll 2^{4n/7}$ for $d \approx n/7$.

Given $d \in \{0,\ldots,n\}$ and a PRP π over $\{0,1\}^n$, the **order** construction realizes a PRF $f^{\mathrm{ord}}_d : \{0,1\}^{n-d} \to S_{2^d}$. Here, S_{2^d} denotes the set of permutations over 2^d elements. The function f^{ord}_d maps $x \in \{0,1\}^{n-d}$ to $f^{\mathrm{ord}}_d(x) \in S_{2^l}$ by sorting the 2^d values $\pi(0\cdots000||x)$, $\pi(0\cdots001||x)$, $\ldots$, $\pi(1\cdots111||x)$.[1] The order construction provably preserves the full security of π: if π is a $(\infty,0)$-secure PRP, then f^{ord}_d is a $(\infty,0)$-secure PRF. On the other hand, the order construction is quite slow, since computing $f^{\mathrm{ord}}_d(x)$ takes 2^d invocations of π.

Recently, Bellare and Impagliazzo [3] described a **general probabilistic lemma** to upper bound the advantage of an adversary in distinguishing between two families of functions.[2]

As an example for applying their general technique, they consider converting a PRP into a PRF. They analyse SUM^2, the two-dimensional special case of the SUM^d-construction we consider in the current paper. They also apply their general technique to analyse two more PRP→PRF constructions: the TWIN^2 variant of SUM^2 (not using the name "TWIN^2"), and the truncate construction from [7].

4 The Construction $\mathrm{SUM}^d(x) = \bigoplus_{i=1}^{d} \pi_i(x)$

Consider $d \geq 1$ permutations $\pi_1, \ldots, \pi_d$, we define $\mathrm{SUM}^d \in \mathbb{F}_n$ by

$$\mathrm{SUM}^d(x) = \pi_1(x) \oplus \cdots \oplus \pi_d(x).$$

In the appendix, we regard the the two-dimensional special case SUM^2. The proof of Theorem 5 in the appendix is similar to the proof of Theorem 2 in this

[1] In fact, [7] deals with a function $f^{\mathrm{ord}*}_d : \{0,1\}^{n-d} \to \{0,1\}^{2^d-1}$. Note that 2^{2^d-1} is the largest power of two dividing $(2^d)! = |S_{2^d}|$ [7, Lemma 1]. Computing $f^{\mathrm{ord}*}_d$ requires 2^d invocations of π and $2^d - 1$ comparisons.

[2] When the current paper was originally written, its author was unaware of [3]. An anonymous referee provided the reference.

section, but requires less technical details. It may be instructive for the reader to first skip to the appendix at page 488 and work through the proof of Theorem 5, and then to continue with the current section.

Theorem 2. *For $d \geq 1$ random permutations $\pi_1, \ldots, \pi_d \in \mathbb{F}_n$ and $q \leq 2^{n-1}/d$ is the function* SUM^d *a (q, a)-secure PRF with*

$$a \leq 2^{-d(n-1)} * \sum_{0 \leq i < q} i^d.$$

The proof of Theorem 2 requires some technical definitions and lemmas provided below. Set $N := \{0,1\}^n$.

Definition 2. *The set $T \subseteq N^d$ is* "fair"*, if for every $y \in N$*

$$\left| \{ (x_1, \ldots, x_d) \in T \mid x_1 \oplus \cdots \oplus x_d = y \} \right| = \frac{|T|}{|N|} = \frac{|T|}{2^n}.$$

If $(x_1, \ldots, x_d) \in_R T$, then $y = x_1 \oplus \cdots \oplus x_d$ is a uniformly distributed random value in N if and only if T is fair. To deal with sets that may be unfair, we also define a measurement of being "almost fair".

Definition 3. *$T \subseteq N^d$ is "z-fair":*

- *If a set $V \subseteq N^d$ exists with $|V| = z$ and $V \cap T = \{\}$, such that $V \cup T$ is fair. We call V a "completion set" (short: "c-set") for T.*
- *Or if a set $U \subseteq T$ with $|U| = z$ exists (an "overhanging set" or "o-set"), such that $T - U$ is fair. We also say: T is "z-overhanging-fair".*

Lemma 1.

(a) *Consider the sets $A \subseteq N^a$ and $B \subseteq N^b$. If either A or B or both are fair, then $A \times B \subseteq N^{ab}$ is fair, too.*
(b) *If the two sets $B \subseteq A \subseteq N^d$ are fair, then so is $A - B$.*
(c) *If A is fair and $B \subseteq A$, then $A - B$ is $|B|$-fair.*
(d) *If the two sets $A \subseteq N^d$ and $B \subseteq N^d$ are fair and $|A| \geq |B|$, then $A - B = A \cap \overline{B}$ is $|B - A|$-overhanging-fair.*

Proof: The proofs of (a) and (c) are trivial. Regarding (b), note that A is fair: $| \{ (x_1, \ldots, x_d) \in A \mid x_1 \oplus \cdots \oplus x_d = y \} | = |A|/2^n$ for every $y \in 2^n$. Similarly: $| \{ (x_1, \ldots, x_d) \in B \mid x_1 \oplus \cdots \oplus x_d = y \} | = |B|/2^n$. Thus we get $| \{ (x_1, \ldots, x_d) \in A \mid (x_1, \ldots, x_d) \notin B \text{ and } x_1 \oplus \cdots \oplus x_d = y \} | = |A - B|/2^n$, hence $A - B$ is fair.

To show (d), consider a fair set $B^* \subseteq A$ with $|B^*| = |B|$. B^* contains the elements $x \in (A \cap B)$, and, for every $(x_1, \ldots, x_d) \in (\overline{A} \cap B)$, the set B^* contains a unique representative $(y_1, \ldots, y_d) \in (A \cap \overline{B})$ with $x_1 \oplus \cdots \oplus x_d = y_1 \oplus \cdots \oplus y_d$. Note that such a set B^* exists since $|B| = |B^*| \leq |A|$ and both A and B are fair. By $R \subseteq B^*$, we denote the set of such representatives, I.e., $|R| = |\overline{A} \cap B|$. Since $A - B^* = (A - B) - R$ is fair, i.e., $A - B$ is $|R|$-overhanging-fair. □

Lemma 2. *Consider the sets $T' \subseteq N^{d-1}$ and $T'' \subseteq N$. Let $z'' = 2^n - |T''|$ (hence T'' is z''-fair). Let $T = T' \times T''$ and $|T| \geq z'z''$. If T' is z'-fair, then T is $z'z''$-fair. More exactly:*

(a) *If $V' \subseteq N^{d-1}$ with $|V'| = z'$ is a c-set for T', then an o-set $U \subseteq T$ for T exists with $|U| = z'z''$.*

(b) *If $U' \subseteq T'$ with $|U'| = z'$ is an o-set for T', then a c-set $V \subseteq N^d$ of size $|V| = z'z''$ exists for T.*

Proof: Note that $V'' = N - T''$ is a c-set for T'' with $|V''| = z''$.

For (a), let $V' \subseteq N^{d-1}$ with $|V'| = z'$ be a c-set for T'. Due to Lemma 1(a), both sets $T' \times (T'' \cup V'')$ and $(T' \cup V') \times V''$ are fair, and

$$\begin{aligned} T' \times T'' &= (T' \times (T'' \cup V'')) - ((T' \cup V') \times V'') \\ &= (T' \times T'') \cup (T' \times V'') - ((T' \times V'') \cup (V' \times V'')) \\ &= (T' \times T'') - (V' \times V''). \end{aligned}$$

Since $|T' \times T''| = |T| \geq z'z'' = |V' \times V''|$ and thus $|T' \times (T'' \cup V'')| \geq |(T' \cup V') \times V''|$, we can apply Lemma 1(d) and conclude: $T' \times T''$ is $|V' \times V''|$-fair, and $|V' \times V''| = z'z''$. Also, an o-set of size $|V' \times V''| = z'z''$ exists for $T' \times T''$.

Regarding (b), consider the o-set $U' \subseteq T'$ with $|U'| = z'$. As above, we argue that the sets $T' \times (T'' \cup V'')$ and $(T' - U') \times V''$ are fair, and

$$\begin{aligned} &(T' \times T'') \cup (U' \times V'') \\ &\quad = (T' \times (T'' \cup V'')) - ((T' - U') \times V'') \\ &\quad = (T' \times T'') \cup (T' \times V'') - ((T' \times V'') - (U' \times V'')). \end{aligned}$$

Since $((T' - U') \times V'') \subseteq T' \times V'' \subseteq T' \times (T'' \cup V'')$, we can apply Lemma 1(b): the set $(T' \times T'') \cup (U' \times V'')$ is fair. By Lemma 1(c) we find that $T' \times T''$ is $|U' \times V''|$-fair. Especially, $U' \times V''$ is a c-set for $T = T' \times T''$. □

Proof: [of Theorem 2] Our adversary asks $q \leq 2^{n-1}/d$ oracle queries. We write $x_1, \ldots, x_q$ for the inputs chosen by the adversary and $y_1, \ldots, y_q$ for the oracle's corresponding outputs. W. l. o. g., we assume $x_i \neq x_j$ for $i \neq j$. Evaluating SUM^d on these inputs may be thought of as choosing q values $\pi_k(x_1), \ldots, \pi_k(x_q)$ for every $k \in \{1, \ldots, d\}$. Since π_k is a random permutation over $\{0,1\}^n$, the values $\pi_k(x_1), \ldots, \pi_k(x_q)$ are random values in $N = \{0,1\}^n$, except that $\pi_k(x_i) \neq \pi_k(x_j)$ for $i \neq j$. We simply write $\pi_{k,j}$ for $\pi_k(x_j)$. Now, generating the random values $y_i = \text{SUM}^d(x_i)$ may be thought of as choosing $\pi_{k,i} \in_R N - \{\pi_{k,1}, \ldots, \pi_{k,i-1}\}$ for $k \in \{1, \ldots, d\}$ and evaluating $y_i = \pi_{1,i} \oplus \cdots \oplus \pi_{d,i}$. We may as well regard this as choosing the d-tuple $\pi_{1,i}, \ldots, \pi_{d,i} \in_R T_i \subseteq N^d$, where T_i is the set of all d-tuples still avaliable, i.e., $T_1 = N^d$ and $T_{i+1} \subseteq T_i$, or exactly:

$$\begin{aligned} T_{i+1} \;=\; N^d &- (\{\pi_{1,1}, \ldots, \pi_{1,i}\} \times N^{d-1}) \\ &- (N \times \{\pi_{2,1}, \ldots, \pi_{2,i}\} \times N^{d-2}) \\ &- \cdots \\ &- (N^{d-1} \times \{\pi_{d,1}, \ldots, \pi_{d,i}\}) \end{aligned} \tag{1}$$

Note that $|T_{i+1}| \geq 2^{dn} - (di * 2^{(d-1)n})$. We can simulate the generation of the values y_j as follows:

For $i := 1$ to q: choose $t_i = (\pi_{1,i}, \ldots, \pi_{d,i}) \in_R T_i$;
output $y_i = \pi_{1,i} \oplus \cdots \oplus \pi_{d,i}$.

The sets T_{i+1} are i^d-fair:
We set $j := i + 1$ and show that if d is odd, a c-set V_j exists for T_j with $|V_j| = (j-1)^d$, and, if d is even, an o-set U_j for T_j of size $|U_j| = (j-1)^d$ exists.

We prove this by induction. If $d = 1$, $V_j = \{y_1, \ldots, y_{j-1}\}$ is a c-set for T_j and $|V_j| = (j-1)^1$. For $d > 1$, we split the d-tuples $(\pi_{j,1}, \ldots, \pi_{j,d}) \in T_j$ up into a $(d-1)$-tuple $(\pi_{j,1}, \ldots, \pi_{j,d-1}) \in T'_j \subseteq N^{d-1}$ and a single value $\pi_{j,d} \in T''_j = N - V''_j$ with $V''_j = \{\pi_{d,1}, \ldots, \pi_{d,j-1}\}$. We know that T''_j is $(j-1)$-fair. Note that $j \le q \le 2^{n-1}/d$, hence $|T'_j| \ge (j-1)^{d-1}$, $|T''_j| \ge j - 1$, and, by induction, $|T_j| \ge (j-1)^d$. This will allow us to apply Lemma 2.

Let d be even. Then $d-1$ is odd. Assume that a c-set V'_j for T'_j exists of size $|V'_j| = (j-1)^{d-1}$. The claim follows from Lemma 2(a).

Now, let d be odd. Assume that T'_j is $(j-1)^{d-1}$-fair, and that an o-set $U'_j \subseteq T'_j$ exists with $|U'_j| = (j-1)^d$. The claim follows from Lemma 2(b).

Choosing the d-tuples $(\pi_{1,i}, \ldots, \pi_{d,i})$ from fair sets:
Since we know that c-sets or o-sets of size $(i-1)^d$ for T_i exist, we can simulate the generation of the y_i as described in Figure 1. Either, d is odd and a c-set V_i for T_i exists, or d is even and an o-set U_i exists.

Even d:

```
Set bad := 0;
for i := 1 to q:
    determine T_i and U_i;
    choose t_i = (π_{1,i}, ..., π_{d,i}) ∈_R T_i;
    if t_i ∈ U_i then
      bad := 1;
    output y_i := π_{1,i} ⊕ ··· ⊕ π_{d,i};
output bad.
```

Odd d:

```
Set bad := 0;
for i := 1 to q:
    determine T_i and V_i; S_i := T_i ∪ V_i;
    choose t_i = (π_{1,i}, ..., π_{d,i}) ∈_R S_i;
    if t_i ∉ T_i then
      bad := 1;
      choose t_i = (π_{1,i}, ..., π_{d,i}) ∈_R T_i;
    output y_i := π_{1,i} ⊕ ··· ⊕ π_{d,i};
output bad.
```

Fig. 1. Two simulations for the PRF SUM^d

When the output $y_i := \pi_{1,i} \oplus \cdots \oplus \pi_{d,i}$ is generated, the d-tuple $t_i = (\pi_{1,i}, \ldots, \pi_{d,i})$ is a uniformly distributed random value $t_i \in_R T_i$. The simulation generates an additional value bad $\in \{0, 1\}$. If bad $= 0$, all t_i are uniformly distributed random d-tuples chosen from fair sets, and thus $y_i \in_R N$. Thus, the advantage of every adversary is at most pr[bad $= 1$].

Evaluating pr[bad $= 1$]:
The simulation in Figure 1 outputs bad $= 1$ if and only if the then-clause is

executed at least once, i.e., $\mathrm{pr}[\mathrm{bad}=1] \le \sum_{1\le i\le q} \mathrm{pr}[\mathrm{then}]$. We get

$$\mathrm{pr}[\mathrm{then}] = \left\{ \begin{array}{ll} (i-1)^d/|T_i| & \text{for even } d \\ (i-1)^d/(|T_i|+(i-1)^d) & \text{for odd } d \end{array} \right\} \le \frac{(i-1)^d}{|T_i|}$$

Since $|T_{i+1}| \ge 2^{dn} - (di * 2^{(d-1)n})$ and $i \le q \le 2^{n-1}/d$, $|T_{i+1}| \ge 2^{dn} - 2^{dn-1} = 2^{dn-1}$ and thus

$$\frac{(i-1)^d}{|T_i|} \le \frac{(i-1)^d}{2^{dn-1}}$$

$$\Rightarrow \quad \mathrm{pr}[\mathrm{bad}=1] \le \sum_{1\le i\le q} \mathrm{pr}[\mathrm{then}] \le \sum_{1\le i\le q} \frac{(i-1)^d}{2^{dn-1}} \le \frac{1}{2^{dn-1}} \sum_{0\le i<q} i^d.$$

Hence $a \le \mathrm{pr}[\mathrm{bad}=1] \le 2^{-dn+1} \sum_{0\le i<q} i^d$. □

Note that $\sum_{1\le i<q} i^d = \theta(i^{d+1})$, hence $\mathrm{Adv}^{\mathrm{Fun}}_{A,\mathrm{SUM}^d} \le \theta(q^{d+1}/2^{nd})$. Depending on d, we provide some examples. For every adversary A, we get:

$$d=1\text{:} \quad \sum_{0\le i<q} i = \frac{q(q-1)}{2} \le \frac{q^2}{2} \;\Rightarrow\; \mathrm{Adv}^{\mathrm{Fun}}_{A,\mathrm{SUM}^1} \le \frac{q^2}{2^n} \tag{2}$$

$$d=2\text{:} \quad \sum_{0\le i<q} i^2 = \frac{2q^3-3q^2+q}{6} \le \frac{q^3}{3} \;\Rightarrow\; \mathrm{Adv}^{\mathrm{Fun}}_{A,\mathrm{SUM}^2} \le \frac{q^3}{3*2^{2n-1}} \tag{3}$$

$$d=3\text{:} \quad \sum_{0\le i<q} i^3 = \frac{q^2(q-1)^2}{4} \le \frac{q^4}{4} \;\Rightarrow\; \mathrm{Adv}^{\mathrm{Fun}}_{A,\mathrm{SUM}^3} \le \frac{q^4}{2^{3n+1}}. \tag{4}$$

In general, SUM^d is secure against adversaries asking $q \ll \sqrt[d+1]{2^{dn-1}}$ queries. If a pessimistic estimate of q gives a value $q \ll 2^n$, we can choose d accordingly. In practice, d will be small, e.g., $d \le 10$.

5 The Construction $\mathrm{TWIN}^d(x) = \bigoplus_{i=0}^{d-1} \pi(dx+i)$

The SUM^d-construction requires d independent PRPs $\pi_1, \ldots, \pi_d$. We may use one block cipher running under d different keys to implement the π_i. Depending on our choice of block cipher and on hardware limitations, frequently changing between encryption under d different keys may be costly, though. Can we construct a secure PRF using a single PRP π over $\{0,1\}$? Consider the function $\mathrm{TWIN}^d : \{0,1\}^{n-\lceil \log_2(d) \rceil} \to \{0,1\}^n$:

$$\mathrm{TWIN}^d(x) = \pi(dx) \oplus \cdots \oplus \pi(dx+d-1).$$

(Recall that we interchangeably view b-bit strings $s \in \{0,1\}^b$ as numbers $s \in \{0,\ldots,2^b-1\}$. Thus, $x \in \{0,1\}^{n-\lceil \log_2(d) \rceil}$ represents a number $x \le (2^{n-\log_2(d)}-1) = 2^n/d-1$, the product dx is at most 2^n-d, and hence $dx+d-1 \le 2^n-1$ can be written as an n-bit string.)

Theorem 3. *For $d \geq 1$, a random permutation $\pi \in \mathbb{F}_n$ and $q \leq 2^{n-1}/d^2$ is* $\text{TWIN}^d(x) = \bigoplus_{i=0}^{d-1} \pi(dx+i)$ *a (q,a)-secure PRF with*

$$a \leq \frac{qd^2}{2^n} + \frac{1}{2^{dn-1}} \sum_{0 \leq i < q} i^d.$$

Proof: As in the proof of Theorem 2, the adversary asks q queries $x_1, \ldots, x_q$, w.l.o.g. $x_i \neq x_j$ for $i \neq j$, and learns q responses $y_i = \text{TWIN}(x_i)$. We define $\pi_{di-d+1} = \pi(dx_i), \ldots, \pi_{di} = \pi(dx_i + d - 1)$, $N = \{0,1\}^n$, and

$$C^* = \{\, (s_1, \ldots, s_d) \in N^d | \exists (i,j) : i \neq j, s_i = s_j \,\}.$$

Clearly, the d-tuples $(\pi_{di-d+1}, \ldots, \pi_{di})$ are not in C^*. Note that $|C^*| = 2^{(d-1)n} * d(d-1)/2 \leq 2^{(d-1)n} d^2/2$. Similar to Equation (1), we define a set T_i^* of the d-tuples still avaliable: $T_i^* = T_i^{**} - C^*$, $T_1^{**} = N^d$, and

$$\begin{aligned} T_{i+1}^{**} \;=\; N^d \;-\; & (\{\pi_1, \ldots, \pi_{di}\} \times N^{d-1}) \\ & - (N \times \{\pi_1, \ldots, \pi_{di}\} \times N^{d-2}) \\ & - \quad \cdots \\ & - \quad (N^{d-1} \times \{\pi_1, \ldots, \pi_{di}\}). \end{aligned} \tag{5}$$

Note that $|T_{i+1}^{**}| \geq 2^{dn} - (d^2 * i * 2^{(d-1)n})$. We simulate generating the y_i:

For $i := 1$ to q: choose $(\pi_{di}, \ldots, \pi_{di+d-1}) \in_R T_i^*$
output $y_i = \pi_{di} \oplus \cdots \oplus \pi_{di+d-1}$.

The sets T_{i+1}^{**} are $(di)^d$-fair:

Compare Equations (1) and (5). Set $j := di + 1$ and show the $(j-1)^d$-fairness of the sets T_j^{**} as in the proof of Theorem 2.

Choosing the d-tuples $(\pi_{di}, \ldots, \pi_{di+d-1})$ from fair sets:

The sets T_i^{**} are $(di-d)$-fair, i.e., c-sets or o-sets of size $(di-d)^d$ for T_i^{**} exist. We argue as in the proof of Theorem 2: If d is odd, then a c-set V_i for T_i^{**} exists. If d is even, an o-set U_i exists. Figure 2 describes the corresponding simulations.

In addition to Figure 1, the simulation in Figure 2 takes care that t_i is in T_i^*, not just in T_i^{**}. If the last output is $bad = 0$, all d-tuples t_i used to generate the y_i are uniformly chosen values from fair sets $T_i^{**} - U_i$ or $V_i \cup T_i^{**}$, hence $a \leq \text{pr}[\text{bad} = 1]$.

Evaluating pr[bad = 1]:

We get bad = 1 if and only if one of the two then-clauses is executed at least once. By B_i^1 we denote the event that the then-clause marked by $(*)$ is executed in round i, B^1 denotes the event that this clause is executed in any round $i \in \{1, \ldots, q\}$, i.e., $\text{pr}[B^1] \leq \sum_{i=1}^q B_i^1$. For the then-clause marked by $(**)$, we define the similar events B_i^2 and B^2. Thus $\text{pr}[\text{bad} = 1] \leq \text{pr}[B^1] + \text{pr}[B^2]$. We start with $\text{pr}[B^1]$:

$$\text{pr}[B_i^1] = \left\{ \begin{array}{ll} (di-d)^d / |T_i^{**}| & \text{for even } d \\ (di-d)^d / (|T_i^{**}| + (di-d)^d) & \text{for odd } d \end{array} \right\} \leq \frac{(di-d)^d}{|T_i^{**}|}.$$

Even d:

Set bad := 0;
for $i := 1$ to q:
  determine T_i^{**} and U_i;
  $t_i = (\pi_{di-d+1}, \ldots, \pi_{di}) \in_{\mathsf{R}} T_i^{**}$;
  if $t_i \in U_i$ then $(*)$
    bad := 1;
  if $t \in C^*$ then $(**)$
    $t_i \in_{\mathsf{R}} T_i^{**} \cap \overline{C^*}$;
  output $y_i := \pi_{di-d+1} \oplus \cdots \oplus \pi_{di}$;
output bad.

Odd d:

Set bad := 0;
for $i := 1$ to q:
  determine T_i^{**} and V_i;
  $t_i = (\pi_{di-d+1}, \ldots, \pi_{di}) \in_{\mathsf{R}} T_i^{**} \cup V_i$;
  if $t \notin T_i^{**}$ then $(*)$
    bad := 1; choose $t_i \in_{\mathsf{R}} T_i^{**}$;
  if $t \in C^*$ then $(**)$
    $t_i \in_{\mathsf{R}} T_i^{**} \cap \overline{C^*}$;
  output $y_i := \pi_{di-d+1} \oplus \cdots \oplus \pi_{di}$;
output bad.

Fig. 2. Two simulations for TWIN^d

Since $|T_{i+1}^{**}| \geq 2^{dn} - (d^2 * i * 2^{(d-1)n})$ and $i \leq q \leq 2^{n-1}/d^2$, we get $|T_{i+1}^{**}| \geq 2^{dn} - 2^{dn-1} \geq 2^{dn-1}$ and thus

$$\frac{(i-1)^d}{|T_i|} \leq \frac{(i-1)^d}{2^{dn-1}} \quad \Rightarrow \quad \text{pr}[B^1] \leq \sum_{1 \leq i \leq q} \frac{(i-1)^d}{2^{dn-1}} = \frac{1}{2^{dn-1}} \sum_{0 \leq i < q} i^d.$$

Now we bound $\text{pr}[B^2]$. Since $|T_i^{**}| \geq 2^{nd} - (d^2) * i * 2^{(d-1)n} \geq 2^{dn-1}$ for $i \leq q \leq 2^{n-1}/d^2$ and $|C^*| \leq 2^{(d-1)n} * d^2/2$ we get

$$\text{pr}[B_i^2] \leq \frac{|C^*|}{|T_i^{**}|} \leq \frac{2^{(d-1)n} * \frac{d^2}{2}}{2^{dn-1}} = \frac{2^{(d-1)n} * d^2}{2^{dn}} = \frac{d^2}{2^n}$$

$$\Rightarrow \quad \text{pr}[B^2] \leq \sum_{1 \leq i \leq q} \text{pr}[B_i^2] \leq \sum_{1 \leq i \leq q} \frac{d^2}{2^n} = \frac{q * d^2}{2^n}$$

hence $a \leq \text{pr}[\text{bad} = 1] \leq 2^{-dn+1} \sum_{0 \leq i < q} i^d + q * d^2/2^n$. □

Consider $d \in \{1, 2, 3\}$. Based on Equations (2)–(4), we get

$$\begin{aligned} d = 1&: \ a \leq \ q/2^n + q^2/2^n \\ d = 2&: \ a \leq 4q/2^n + q^3/(3 * 2^{2n-1}) \\ d = 3&: \ a \leq 9q/2^n + q^4/2^{3n+1}. \end{aligned}$$

The $(2^{-dn+1} \sum_{0 \leq i < q} i^d)$-term determines the maximum size of q, at least if d is such small and for practically interesting $n \geq 64$. We conclude: for small d, the PRF-security of TWIN^d is close to the PRF-security of SUM^d.

6 Final Comments

6.1 Practical Security

We presented constructions for PRFs from permutations, and we proved our PRFs to be (q, a)-secure if the permutations are $(\infty, 0)$-secure (or "ideal") PRPs.

In practice, our PRPs (i.e. block ciphers) are not ideal ones. What we actually are interested in is a close relationship between the derivation of the underlying permutations from being ideal $(\infty, 0)$-secure PRPs, and the derivation of the constructed PRF from being $(\infty, 0)$-secure. This is quite straightforward, and we exemplify this for the TWIN^d-construction:

Theorem 4. *Let q and a be chosen such that TWIN^d is (q, a)-secure in the ideal case. Let B be a (t, qd, δ)-secure PRP. The function $f : \{0,1\}^{n-\lceil d \rceil} \to \{0,1\}^n$ defined by*

$$f(x) = B(dx) \oplus \cdots \oplus B(dx + d - 1) \tag{6}$$

is $(t - qt', q, a + \delta)$-secure. Here, t', denotes the time to evaluate Expression (6).

Note that the function f is indeed an instantiation of the PRF TWIN^d using the concrete (non-ideal) PRP B.

Proof: [of Theorem 4] Assume an adversary A_f running at most $t - qt'$ units of time, asking for q values $f(x_1), \ldots f(x_q)$, achieves an advantage $\text{Adv}^{\text{Fun}}_{A_f, f} > a + \delta$. We describe an adversary A_B for B, using A_f as some kind of "subroutine". The performance of A_B disproves the (t, qd, δ)-security of B.

Whenever A_f chooses $x \in \{0,1\}^{n-\lceil d \rceil}$ and asks for $f(x)$, A_B asks for the values $B(dx)$, $\ldots$, $B(dx + d - 1)$ and evaluates Expression (6). A_B uses the output-bit produced by A_f as its own output-bit.

Running A_B requires the running time for A_f plus the additional time qt' for q evaluations of (6), and q queries for the f-oracle are translated into dq queries for the B-oracle. Since TWIN^d is (q, a)-secure in the ideal case, and since A_f is assumed to achieve an advantage of more than $a + \delta$, the advantage of A_B in distinguishing between B and an ideal block cipher exceeds δ. □

Given an estimate of the number q of plaintext/ciphertext pairs the adversary can learn, and given the block size n, the security architect must decide on the size of the parameter d. Our analysis provides precise bounds (instead of asymptotic estimates) to help her making a reasonable decision. This kind of reasoning, the "concrete security analysis", was initiated in [4].

6.2 Super Pseudorandom Permutations

Luby and Rackoff [9] introduced a distinction between *super* PRPs and (ordinary) PRPs: For ordinary PRPs, the adversary may only choose values x and ask the oracle Q for $Q(x)$. Such adversaries are "chosen plaintext" adversaries. On the other hand, super PRPs need to resist "combined chosen plaintext / chosen ciphertext" adversaries, i.e., adversaries also able to choose y and ask for $Q^{-1}(y)$. For our constructions we don't need super PRPs – ordinary PRPs are sufficient. This makes our results all the more meaningful.

6.3 Comparison and Conclusion

This paper deals with the construction of PRFs from PRPs. We propose two constructions, $\text{SUM}^d : \{0,1\}^n \to \{0,1\}^n$ and $\text{TWIN}^d\{0,1\}^{n-\lceil \log_2(d) \rceil} \to \{0,1\}^n$, based on PRPs over $\{0,1\}^n$.

Our constructions preserve the security of the underlying PRP better than the truncate construction from [7] and are much more efficient than the order construction, also from [7].

The truncate construction from [7] is re-considered in [3], claiming an improved security analysis compared to [7]. Also, [3] deals with SUM^2 and TWIN^2 – the two-dimensional variants of the constructions we scrutinise here. In short, if the number q of oracle queries is $q \ll 2^n/O(n)$, both the SUM^2 and the TWIN^2 construction are claimed to be secure. (For TWIN^2, a short sketch of proof is given.) Note that the results in claimed in [3] are significantly better than the results provided in the current paper.

Now consider data dependent re-keying, (DDRK) [5]. If k is the key size of the underlying block cipher, the result on the security of DDRK [5, Theorem 5.2] requires $q \ll 2^{4k/5}$. In fact, that result depends on the assumption that exhaustively searching $2^{4k/5}$ keys is infeasible. If, say, $k = 80$, the effective key-length guaranteed by the result is only $4*80/5$ bit $= 64$ bit. This is a disadvantage, compared to our schemes. (Note though: [5] conjecture that the bound on q can be improved to $q \ll 2^{k(1-\epsilon)}$.) Depending on which block cipher is used and on hardware constraints, the very frequent key changes needed for DDRK can constitute another disadvantage.

For some applications, e.g. on low-end smartcards, even the effort to switch between only d fixed secret keys may be prohibitive. In this case, the TWIN^d construction is superior to SUM^d, if a PRF with only $n - \lceil \log_2(d) \rceil$ input bits is acceptable.

Acknowledgements

Thanks to the anonymous referees for their helpful comments.

References

1. W. Aiello, R. Venkatesan, "Foiling Birthday Attacks in Length Doubling Transformations", Eurocrypt 96, 307–320, Springer LNCS 1070.
2. M. Bellare, A. Desai, E. Jokipii, P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption: The DES Modes of Operation", FoCS 97, IEEE press.[3]
3. M. Bellare, R. Impagliazzo, "A Tool for Obtaining Tighter Security Analyses of Pseudorandom Function Based Constructions, with Applications to PRP→PRF Conversion" (FOCS '99 submission), *Theory of Cryptography Library*[4], record 99-24 (1999).
4. M. Bellare, J. Kilian, P. Rogaway, "The Security of the Cipher Block Chaining Message Authentication Code", Crypto 94 Proceedings, Springer LNCS 839.[5]
5. M. Bellare, T. Krovetz, P. Rogaway, "Luby-Rackoff Backwards: Increasing Security by Making Block Ciphers Non-Invertible", Eurocrypt 98, Springer LNCS 1403.[6]

[3] Full version online: `http://wwwcsif.cs.ucdavis.edu/~rogaway/papers/list.html`
[4] The library is online: `http://philby.ucsd.edu/cryptolib/`
[5] Full version online: `http://www-cse.ucsd.edu/users/mihir/crypto-papers.html`
[6] Full version online: `http://www-cse.ucsd.edu/users/mihir/crypto-papers.html`

6. O. Goldreich, S. Goldwasser, S. Micali, "How to Construct Random Functions", Journal of the ACM, Vol. 33, No 4, 792–807 (1986).
7. C. Hall, D. Wagner, J. Kelsey, B. Schneier, "Building PRFs from PRPs", Crypto 98, Springer LNCS 1462.[7]
8. L. Levin, "One Way Functions and Pseudorandom Generators", *Combinatorica*, 7 (4), 357–363 (1987).
9. M. Luby, C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions", *SIAM J. Comput.*, Vol. 17, No. 2, 373–386, (1988).
10. S. Lucks, "Faster Luby-Rackoff Ciphers", Fast Software Encryption 1996, Springer LNCS 1039, 189–205.
11. M. Naor, O. Reingold, "On the Construction of Pseudo-Random Permutations: Luby-Rackoff Revisited", *J. of Cryptology*, Vol. 12, No. 1, 29–66 (1999).
12. J. Patarin, "About Feistel Schemes with Six (or More) Rounds", Fast Software Encryption 1997, Springer LNCS 1372, 103–121.

Appendix: The 2-Dimensional Construction $\mathrm{SUM}^2(x) = \sigma(x) \oplus \tau(x)$

To improve the tangibility of this paper, the abstract deals with a simple but non-trivial special case of SUM^d, the 2-dimensional variant

$$\mathrm{SUM}^2(x) = \sigma(x) \oplus \tau(x),$$

depending on two permutations $\sigma, \tau\{0,1\}^n \to \{0,1\}^n$. Not surprisingly, SUM^2 is not a $(\infty, 0)$-secure PRF. In fact, collisions are too probable. E.g., the probability that the first two pseudorandom values y_1 and y_2 generated by using SUM^2 to collide is too high: $\mathrm{pr}[y_1 = y_2] > 2^{-n}$. To see this, consider simulating SUM^2.

Initially, there are 2^{2n} pairs $(s,t) \in \{0,1\}^n$ to choose for $(\sigma(x_1), \tau(x_1))$. For every value $y \in \{0,1\}^n$, there exist exactly 2^n pairs (s,t) with $\sigma(x_1) \oplus \tau(x_1) = y_1$.

Let $x_2 \neq x_1$. In the second step, a pair $(s',t') = (\sigma(x_2), \tau(x_2))$ is chosen with $s' \neq s$ and $t' \neq t$. There are 2^{n-1} values $s' \neq s$ and as much values $t' \neq t$, hence the number of such pairs is $(2^{n-1})^2$. For every value $s' \neq s$, exactly one value $t' \neq t$ exists with $s' \oplus t' = s \oplus t$, and $y_1 = y_2$ if and only if $s' \oplus t' = s \oplus t$. Hence, exactly $2^n - 1$ of the $(2^{n-1})^2$ possible pairs (s',t') induce $y_2 = y_2$, and thus

$$\mathrm{pr}[y_1 = y_2] = \frac{2^n - 1}{(2^n - 1)^2} = \frac{1}{2^n - 1}.$$

If SUM^2 where an ideal random function, we had $\mathrm{pr}[y_1 = y_2] = 2^{-n}$. But how good is the PRF SUM^2 actually?

Theorem 5. *For random permutations $\sigma, \tau \in \mathbb{F}_n$ and $q \leq 2^{n-1}$, the function f with $f(x) = \mathrm{SUM}^2(x) = \sigma(x) \oplus \tau(x)$ is a (q,a)-secure PRF with $a = q^3/2^{2n-1}$.*

[7] Full version online: `http://www.counterpane.com/publish-1998.html`

Proof: The set $T \subseteq (\{0,1\}^n)^2$ is "fair",[8] if for every value $y \in \{0,1\}^n$

$$\left|\{\,(\sigma_*, \tau_*) \in T \,|\, \sigma_* \oplus \tau_* = y\,\}\right| = \frac{|T|}{2^n}.$$

The adversary A asks $q \leq 2^{n-1}$ oracle queries $x_1, \ldots, x_q$, w. l. o. g. $x_i \neq x_j$ for $i \neq j$. We write $y_1, \ldots, y_q$ for the corresponding oracle responses.

Consider evaluating SUM^2 by choosing a pair $(\sigma_i, \tau_i) = (\sigma(x_i), \tau(x_i))$ and computing $y_i = \sigma_i \oplus \tau_i$. If all (σ_i, τ_i) where randomly chosen from a fair set and uniformly distributed, then the sums $y_i = \sigma_i \oplus \tau_i$ would be uniformly distributed random values – un-distinguishable from the output of a random function.

The remainder of this proof is organised as follows:

1. We describe the sets $T_i \subseteq (\{0,1\}^n)^2$ the pairs (σ_i, τ_i) are chosen from, and we specify fair subsets $U_i \subseteq T_i$ with $|U_i| = |T_i| - (i-1)^2$.
2. We describe how to choose the pairs (σ_i, τ_i) from the fair sets U_i, except when a "bad" event happens.
3. We calculate the probability of the "bad" event.

Let $i \neq j$. Since σ and τ are permutations, $\sigma_i \neq \sigma_j$ and $\tau_i \neq \tau_j$. Thus, by choosing the pair (σ_i, τ_i), all pairs (s,t) with $s = \sigma_i$ or $t = \tau_i$ are "consumed", i.e., cannot be chosen for (σ_j, τ_j).

By S_i, we denote the set of consumed pairs before the choice of (σ_i, τ_i). By $T_i = (\{0,1\}^n)^2 - S_i$, we denote the set of un-consumed pairs. Note that (T_i is fair) $\Leftrightarrow$ (S_i is fair). Since $S_1 = \{\}$, both S_1 and T_1 are fair and y_1 is a uniformly distributed random value. Given $(\sigma_1, \tau_1), \ldots, (\sigma_k, \tau_k)$ we define $U_{k+1} \subseteq T_{k+1}$. Consider the following $2k$ fair sets of pairs:

$$\begin{aligned} &\{(\sigma_1, \tau_*) \mid \tau_* \in \{0,1\}^n\}, \ \ldots, \{(\sigma_k, \tau_*) \mid \tau_* \in \{0,1\}^n\} \\ \text{and } &\{(\sigma_*, \tau_1) \mid \sigma_* \in \{0,1\}^n\}, \ \ldots, \{(\sigma_*, \tau_k) \mid \sigma_* \in \{0,1\}^n\}. \end{aligned}$$

S_{k+1} is the union of the above $2k$ sets of pairs. If the above 2k sets were all disjoint, S_{k+1} would be fair. But actually, exactly k^2 pairs are contained in two of the above sets, namely all pairs (σ_i, τ_j) with $i, j \in \{1, \ldots, k\}$. We arbitrarily choose k^2 unique representatives (σ'_i, τ'_j) for (σ_i, τ_j) with $(\sigma'_i, \tau'_j) \in T_{k+1}$ and $\sigma'_i \oplus \tau'_j = \sigma'_i \oplus \tau'_j$. We define U_{k+1} to be the set of all pairs in T_{k+1} except for the representatives (σ'_i, τ'_j). Hence $|U_{k+1}| = |T_{k+1}| - k^2$. By induction one can see that for every $y \in \{0,1\}^n$ the set U_{k+1} contains exactly $2^n - 2k$ pairs (σ_s, τ_t) with $\sigma_s \oplus \tau_t = y$. Since $k \leq q \leq 2^{n-1}$, it is possible run the simulation described in Figure 3, especially, a set U_i exists.

The distribution of the values y_i is as required for SUM^2. The simulation generates an additional value "bad". If bad $= 0$, each of the pairs (σ_i, τ_i) is

[8] This notion of fairness is the two-dimensional special case of Definition 2. If T is fair and we choose $(s,t) \in_R T$, the sum $y = s \oplus t$ is a uniformly distributed random value in $\{0,1\}^n$.

Set bad := 0;
for $i := 1$ to q: determine the sets T_i and U_i;
choose $(\sigma_i, \tau_i) \in_R T_i$;
if $(\sigma_i, \tau_i) \notin U_i$ then bad := 1;
output $y_i = \sigma_i \oplus \tau_i$;
output bad.

Fig. 3. A simulation for the PRF SUM^2

chosen from a fair set U_i, and the sums y_i are un-distinguishable from the output of a random function. Thus $\text{Adv}^{\text{Fun}}_{A,f} \leq \text{pr}[\text{bad} = 1]$ for every adversary A. Using

$$\text{pr}[(\sigma_{i+1}, \tau_{i+1}) \notin U_{i+1}] = \frac{|T_{i+1}| - |U_{i+1}|}{|T_{i+1}|} = \frac{i^2}{(2^n - i)^2},$$

we bound the probability $\text{pr}[\text{bad} = 1]$:

$$\begin{aligned} \text{pr}[\text{bad} = 1] &\leq \sum_{1 \leq i \leq q} \text{pr}[(\sigma_i, \tau_i) \notin U_i] = \sum_{0 \leq i < q} \frac{i^2}{(2^n - i)^2} \\ &\leq \sum_{0 \leq i < q} \frac{i^2}{(2^n - q)^2} = (2^n - q)^{-2} * \sum_{0 \leq i < q} i^2. \end{aligned}$$

Since $q \leq 2^{n-1}$

$$\text{pr}[\text{bad} = 1] \leq (2^{n-1})^{-2} * \sum_{0 \leq i < q} i^2. \tag{7}$$

By using $\sum_{0 \leq i < q} i^2 = (q(q-1)(2q-1))/6 \leq 2q^3/6$ we get

$$\text{pr}[\text{bad} = 1] \leq \frac{q^3}{3 * (2^n - q)^2}$$

and hence $\text{pr}[\text{bad} = 1] \leq q^3/2^{2n-1}$. □

Note that Theorem 5 provides a marginally better bound than Theorem 2 for $d = 2$. This is, because the Theorem 2 considers the general case (and because the current author tried to avoid overcrowding its proof with too many technical details). The general outline of the proofs of Theorems 2, 3, and 5 is quite similar.

Construction of Nonlinear Boolean Functions with Important Cryptographic Properties

Palash Sarkar[1] and Subhamoy Maitra[2]

[1] Applied Statistics Unit, Indian Statistical Institute,
203, B T Road, Calcutta 700 035, INDIA
palash@isical.ac.in
[2] Computer and Statistical Service Center, Indian Statistical Institute,
203, B T Road, Calcutta 700 035, INDIA
subho@isical.ac.in

Abstract. This paper addresses the problem of obtaining new construction methods for cryptographically significant Boolean functions. We show that for each positive integer m, there are infinitely many integers n (both odd and even), such that it is possible to construct n-variable, m-resilient functions having nonlinearity greater than $2^{n-1} - 2^{\lfloor \frac{n}{2} \rfloor}$. Also we obtain better results than all published works on the construction of n-variable, m-resilient functions, including cases where the constructed functions have the maximum possible algebraic degree $n - m - 1$. Next we modify the Patterson-Wiedemann functions to construct balanced Boolean functions on n-variables having nonlinearity strictly greater than $2^{n-1} - 2^{\frac{n-1}{2}}$ for all odd $n \geq 15$. In addition, we consider the properties strict avalanche criteria and propagation characteristics which are important for design of S-boxes in block ciphers and construct such functions with very high nonlinearity and algebraic degree.

1 Introduction

The following four factors are important in designing Boolean functions for stream cipher applications.

Balancedness. An n-variable Boolean function f is said to be balanced if $wt(f) = 2^{n-1}$, where $wt(.)$ gives the Hamming weight and f is considered to be represented by a binary string of length 2^n.

Nonlinearity. The nonlinearity of an n-variable Boolean function f, denoted by $nl(f)$, is the (Hamming) distance of f from the set of all n-variable affine functions. We denote by $nlmax(n)$ the maximum possible nonlinearity of n-variable functions.

Algebraic Degree. An n-variable Boolean function f can be represented as a multivariate polynomial over $GF(2)$. This polynomial is called the Algebraic Normal Form (ANF) of f. The degree of this polynomial is called the algebraic degree or simply the degree of f and is denoted by $deg(f)$. It is easy to see that the maximum algebraic degree of an n-variable balanced function is $n - 1$.

Correlation Immunity. An n-variable Boolean function $f(X_n, \ldots, X_1)$ is said to be correlation immune (CI) of order m if $Prob(f = 1 \mid X_{i_1} = c_1, \ldots, X_{i_m} = c_m) = Prob(f = 1)$, for any choice of distinct $i_1, \ldots, i_m$ from $1, \ldots, n$ and $c_1, \ldots, c_m$ belong to $\{0, 1\}$. A balanced m-th order correlation immune function is called m-resilient. Siegenthaler [16] proved a fundamental relation between the number of variables n, degree d and order of correlation immunity m of a Boolean function : $m + d \leq n$. Moreover, if the function is balanced then $m + d \leq n - 1$.

The set of all n-variable Boolean functions is denoted by Ω_n. We denote by $A_n(m)$ the set of all balanced n-variable functions which are CI of order m. By an (n, m, d, x) function we mean an n-variable, m-resilient function having degree d and nonlinearity x. By an $(n, 0, d, x)$ function we mean an n-variable, degree d, balanced function with nonlinearity x.

A good Boolean function must possess a "good combination" of the above properties to be used in stream ciphers. Previous works to construct such good functions have proceeded in two ways.
1. In the first approach the degree is ignored and the number of variables and correlation immunity are fixed. One then tries to get a function having as high nonlinearity as possible. This approach has been considered in [15, 2] and we call this the $Type - A$ approach.
2. The second approach considers the degree. However, by Siegenthaler's inequality, the maximum possible degree of an n-variable, m-resilient function is $n - m - 1$. Functions achieving this degree have been called *optimized* [7]. As in the first approach one then tries to get as high nonlinearity as possible for optimized functions. Design methods for this class of functions have been considered in [4, 7, 8, 18] and we call this the $Type - B$ approach.

Previous efforts at obtaining resilient functions have sometimes employed *heuristic search techniques* [4, 8]. In certain cases these have provided better results than *constructive techniques* [15, 7]. The list of all such known cases are as follows : (a) $(7, 0, 6, 56)$, $(9, 0, 7, 240)$ and $(9, 2, 6, 224)$ functions from [4] and (b) $(9, 1, 7, 236)$, $(10, 1, 8, 480)$ and $(11, 1, 9, 976)$ functions from [8]. These examples are indicative of the inadequacies of the current constructive techniques. However, heuristic searches cannot be conducted for moderate to large number of variables.

Here we provide a systematic theory for the design of resilient functions. Our techniques are sharp enough to obtain general results which are better than all the examples mentioned above. Corresponding to the list given above we have $(7, 0, 6, 56)$, $(9, 0, 8, 240)$, $(9, 2, 6, 232)$, $(9, 1, 7, 240)$, $(10, 1, 8, 484)$ and $(11, 1, 9, 992)$ functions. Also we are able to prove some difficult results on the nonlinearity of resilient functions. Here for the first time we show that for each order of resiliency m, there are infinitely many n (both odd and even), such that it is possible to construct n-variable, m-resilient functions having nonlinearity greater than $2^{n-1} - 2^{\lfloor \frac{n}{2} \rfloor}$. One consequence of this result is that it completely disproves the conjecture on nonlinearity made in [8]. We use our techniques to present design algorithms for optimized resilient functions and obtain superior results to all known work in this area (see Section 6 for details). The functions

constructed by our methods have a nice representation and though they have quite complicated algebraic normal forms they can be implemented efficiently in hardware. See [12] for details of the hardware implementation.

Next we describe the other contributions of this paper. In Section 7, we use a randomized heuristic to construct for the first time balanced functions with nonlinearity greater than $2^{n-1} - 2^{\frac{n-1}{2}}$ for $n = 15, 17, 19, 21, 23, 25, 27$. We use the functions provided in [9] as the basic input to our algorithm. Earlier these functions [9] were used to obtain balanced functions with nonlinearity greater than $2^{n-1} - 2^{\frac{n-1}{2}}$ only for odd $n \geq 29$ [14]. Also the functions we construct posses maximum algebraic degree $(n-1)$.

S-boxes can be viewed as a set of Boolean functions [10, 6]. Propagation Characteristic(PC) and Strict Avalanche Criteria(SAC) are important properties of Boolean functions to be used in S-boxes. Preneel et al [10] provided basic construction techniques for Boolean functions with these properties.

Propagation Characteristic and Strict Avalance Criteria. Let $\overline{X}$ be an n tuple $X_1, \ldots X_n$ and $\overline{\alpha} \in \{0,1\}^n$. A function $f \in \Omega_n$ is said to satisfy
(1) SAC if $f(\overline{X}) \oplus f(\overline{X} \oplus \overline{\alpha})$ is balanced for any $\overline{\alpha}$ such that $wt(\overline{\alpha}) = 1$.
(2) SAC(k) if any function obtained from f by keeping any k input bits constant satisfies SAC.
(3) PC(l) if $f(\overline{X}) \oplus f(\overline{X} \oplus \overline{\alpha})$ is balanced for any $\overline{\alpha}$ such that $1 \leq wt(\overline{\alpha}) \leq l$.
(4) PC(l) of order k if any function obtained from f by keeping any k input bits constant satisfies PC(l).

In [10], it has been shown that for balanced SAC(k) functions on n variables, $deg(f) \leq n - k - 1$. Recently in [6], balanced SAC(k) functions on n variables with $deg(f) = n - k - 1$ has been identified for $n - k - 1 =$ odd. However, construction of such functions for $n - k - 1 =$ even has been left as an open problem. In [6], balanced SAC(k) functions with high algebraic degree have been proposed. However, balanced SAC(k) functions with both high algebraic degree and high nonlinearity have not been studied. PC(l) of order k functions with good nonlinearity and algebraic degree have been reported in [6].

In Section 8, first we improve the algebraic degree and nonlinearity results of the PC(l) of order k functions reported in [6]. Then motivated by the construction methods of SAC(k) functions in [6], we introduce a new cryptographic criterion called the *restricted balancedness* of Boolean functions and show that certain types of bent functions satisfy this property. Also we modify the functions provided by Patterson and Wiedemann [9] to obtain restricted balancedness while keeping the nonlinearity unchanged. For the first time we consider the properties of balancedness, SAC(k), algebraic degree and nonlinearity together. We construct balanced (using the functions with restricted balancedness) SAC(k) functions in Ω_n with maximum possible algebraic degree $n-k-1$ and very high nonlinearity for $k \leq \frac{n}{2} - 1$. This also shows that there exists balanced SAC(k) functions on n variables with $deg(f) = n-k-1 =$ even, which was posed as an open question in [6]. Also, we present an interesting result on resilient functions satisfying PC(k). In a previous work [15], it was shown that resilient functions

satisfy propagation characteristics with respect to a set of input vectors, but not PC(k) for some k.

2 Preliminaries

The Hamming weight (or simply the weight) of a binary string s is denoted by $wt(s)$ and is the number of ones in the string s. The length of a string s is denoted by $\mid s \mid$ and the concatenation of two strings s_1 and s_2 is written as $s_1 s_2$. Given a string s, we define s^c to be the string which is the bitwise complement of s. The operation $x \oplus y$ on two strings x, y performs the bitwise exclusive OR of the strings x and y.

Let s_1, s_2 be two bit strings of length n each. Then $\#(s_1 = s_2)$ (resp. $\#(s_1 \neq s_2)$) denotes the number of positions where s_1 and s_2 are equal (resp. unequal). The Hamming distance between two strings s_1 and s_2, is denoted by $d(s_1, s_2)$ and is given by $d(s_1, s_2) = \#(s_1 \neq s_2) = wt(s_1 \oplus s_2)$. The Walsh distance between the strings s_1 and s_2 is denoted by $wd(s_1, s_2)$ and is given by $wd(s_1, s_2) = \#(s_1 = s_2) - \#(s_1 \neq s_2)$. The relation between these two measures is as follows. Let s_1, s_2 be two binary strings of length x each. Then $wd(s_1, s_2) = x - 2d(s_1, s_2)$.

Given a bit b and a string $s = s_0 \ldots s_{n-1}$, the string b AND $s = s'_0 \ldots s'_{n-1}$, where $s'_i = b$ AND s_i. The Kronecker product of two strings $x = x_0 \ldots x_{n-1}$ and $y = y_0 \ldots y_{m-1}$ is a string of length nm, denoted by
$x \otimes y = (x_0$ AND $y) \ldots (x_{n-1}$ AND $y)$. The direct sum of two strings x and y, denoted by $x\$y$ is given by $x\$y = (x \otimes y^c) \oplus (x^c \otimes y)$. As an example, if $f = 01$, and $g = 0110$, then $f\$g = 01101001$. Note that both the Kronecker product and the direct sum are not commutative operations. The following result will prove to be important later.

Lemma 1. *Let f_1, f_2 be strings of equal length and g a string of length n. Then $d(f_1\$g, f_2\$g) = n \times d(f_1, f_2)$.*

Four basic properties of direct sum of Boolean functions are given below without proof (see also [9, 15]).

Proposition 1. *Let $f(X_n, \ldots, X_1) \in \Omega_n$ and $g(Y_m, \ldots, Y_1) \in \Omega_m$, with $\{X_n, \ldots, X_1\} \cap \{Y_m, \ldots, Y_1\} = \emptyset$. Then $f\$g$ is in Ω_{n+m} and*
(a) The ANF of $f\$g$ is given by $f(X_n, \ldots, X_1) \oplus g(Y_m, \ldots, Y_1)$.
(b) $f\$g$ is balanced iff at least one of f and g is balanced.
(c) Let f be k_1-resilient and g be k_2-resilient. Then $f\$g$ is $\max(k_1, k_2)$-resilient. Also $f\$g$ is m-resilient if at least one of f or g is m-resilient.
(d) $nl(f\$g) = 2^n nl(g) + 2^m nl(f) - 2nl(f)nl(g)$.

An n-variable Boolean function $f(X_n, \ldots X_1)$ is said to be affine if the ANF of f is of the form $f(X_n, \ldots, X_1) = \bigoplus_{i=1}^{n} a_i X_i \oplus b$ for $a_i, b \in \{0, 1\}$. If b is 0, then the function is said to be linear. Also f is said to be nondegenerate on t variables if t out of n a_i's are 1 and rest are 0. Next we define the following subsets of linear/affine functions.
1. The set $L_n(k)$ (resp. $F_n(k)$) is the set of all n-variable linear functions (resp.

affine functions) which are non-degenerate on exactly k variables.
2. $UL_n(k) = L_n(k) \cup \ldots \cup L_n(n)$ and $DL_n(k) = L_n(1) \cup \ldots \cup L_n(k)$.
3. $UF_n(k) = F_n(k) \cup \ldots \cup F_n(n)$ and $DF_n(k) = F_n(1) \cup \ldots \cup F_n(k)$.
4. $L_n = L_n(0) \cup L_n(1) \cup \ldots \cup L_n(n)$ and $F_n = F_n(0) \cup F_n(1) \cup \ldots \cup F_n(n)$.
The sets L_n and F_n are respectively the sets of all linear and affine functions of n variables. The following result states three useful properties of affine functions.

Lemma 2. *(a) Let $l \in F_n(m)$ and k $(1 \leq k \leq n)$ be an integer. Then $l = l_1 \$ l_2$ for some $l_1 \in L_{n-k}(r)$ and $l_2 \in F_k(m-r)$ for some $r \geq 0$.*
(b) Let $l_1, l_2 \in F_n$. Then $d(l_1, l_2) = 0, 2^n, 2^{n-1}$ (resp. $wd(l_1, l_2) = 2^n, -2^n, 0$) according as $l_1 = l_2$, $l_1 = l_2^c$ or $l_1 \neq l_2$ or l_2^c.
(c) If l is in $UF_n(m+1)$, then l is m-resilient.

Siegenthaler [16] was the first to define CI functions and point out its importance in stream ciphers [17]. A useful characterization of correlation immunity based on Walsh Transform was obtained in [5]. The following result translates the Walsh transform characterization of correlation immunity to Walsh distances.

Theorem 1. *A n-variable Boolean function f is correlation immune of order m, iff $wd(f, l) = 0$, for all $l \in DF_n(m)$.*

3 Construction Ideas for Resilient Functions

3.1 Basic Results

We first define two subsets of Ω_n. Later we will provide construction methods for certain subsets of these sets which have good cryptographic properties.

Definition 1.
1. $\Gamma(n, k, m) = \{f \in \Omega_n : f = f_0 \ldots f_{2^{n-k}-1}, f_i \in A_k(m),\ wt(f_i) = 2^{k-1}\}$.
2. $\Gamma_1(n, k, m) = \{f \in \Omega_n : f = f_0 \ldots f_{2^{n-k}-1}, f_i \in UF_k(m+1)\}$.

Theorem 2. $\Gamma(n, k, m) \subseteq A_n(m)$.

Proof : Observe that if f and g are resilient of order m then so is fg. The result then follows from repeated application of this fact. □

Since any function in $UF_k(m+1)$ is m-resilient, we have the following result.

Lemma 3. $\Gamma_1(n, k, m) \subset \Gamma(n, k, m)$.

The set $\Gamma_1 = \bigcup_{n \geq 3} \bigcup_{1 \leq m \leq n-1} \bigcup_{m+1 \leq k \leq n} \Gamma_1(n, k, m)$ was first obtained by Camion et al in [1], though in an entirely different form. We will show that the extension obtained in Theorem 2 is important and provides optimized functions with significantly better nonlinearities.

Theorem 3. *Let $f \in \Gamma(n, k, m)$ be of the form $f_0 \ldots f_{2^{n-k}-1}$. Let the logical AND of r variables, $X_{i_1} \ldots X_{i_r}$ $(i_1, \ldots, i_r \in \{1, \ldots, k\})$ be a term which occurs in the ANF of an odd number of the f_i's. Then the term $X_n \ldots X_{k+1} X_{i_1} \ldots X_{i_r}$ occurs in the algebraic normal form of f.*

Corollary 1. *Let $f \in \Gamma_1(n,k,m)$ be of the form $f_0 \ldots f_{2^{n-k}-1}$ and let X_i ($i \in \{1,\ldots,k\}$) be a variable which occurs in an odd number of the f_i's. Then the term $X_n \ldots X_{n-k+1}X_i$ occurs in the algebraic normal form of f and hence f is of degree $n-k+1$. Moreover, the maximum degree $n-m-1$ is attained when $k = m+2$.*

Corollary 1 was obtained in [15] and it places a restriction on the value of k for optimized functions in $\Gamma_1(n,k,m)$. However, this restriction can be lifted by using Theorem 3.

Lemma 4. *A degree optimized $(n, m, n-m-1, x)$ function is always nondegenerate.*

The ANF of the functions in Γ and Γ_1 are not simple. This is important from a cryptographic point of view. Given n, m, k, in most cases it is possible to choose two functions f_1 and f_2, such that the ANF's of both f_1 and f_2 are complicated and $f_1 \oplus f_2$ is nondegenerate and has a complicated ANF. In particular, one can choose f_1 and f_2, such that all three functions f_1, f_2 and $f_1 \oplus f_2$ do not depend linearly on any input variable. It is also possible to design functions such that each variable occurs in a maximum degree term. This is possible by ensuring each variable occurs an odd number of times as mentioned in Corollary 1.

In the next four subsections we present the ideas behind the basic construction techniques to be used in this paper. In the later sections we combine several of these ideas to construct resilient functions with very high nonlinearities.

3.2 Method Using Direct Sum with Nonlinear Functions

We first consider the set $\Gamma_1(n,k,m)$. A function f in $\Gamma_1(n,k,m)$ is a concatenation of affine functions in $UF_k(m+1)$. Since there are 2^{n-k} slots to be filled and a maximum of $p = \binom{k}{m+1} + \ldots + \binom{k}{k}$ linear functions in $UL_k(m+1)$, it follows that at least one linear function and its complement must together be repeated at least $t = \lceil \frac{2^{n-k}}{p} \rceil$ times. *We call a linear function and its complement a linear couple. When we say that a linear couple is repeated t times, we mean that the corresponding linear function and its complement are repeated t times in total.* Using Lemma 2, any affine function l in F_n can be considered to be a concatenation of some linear couple in F_k. Thus if one is not careful in constructing f, it may happen that f and l agree at all places for some linear couple repeated t times in f. This means that the nonlinearity drops by $t2^{k-1}$ and gives a lower bound of $2^{n-1} - t2^{k-1}$ on the nonlinearity of f. This is the bound obtained in [15]. However, one can construct $f \in \Gamma_1(n,k,m)$ with significantly better nonlinearities. The following result is the key to the construction idea.

Theorem 4. *Let $f \in \Gamma_1(n,k,m)$ be of the form $f_1 \ldots f_p$ where, $p = 2^{n-k-r}$ for some r and for each i, f_i is in Ω_{k+r} and is of the form $f_i = g_i \$ \lambda_i$, where g_i is a maximum nonlinear function on r variables and λ_i is in $UL_k(m+1)$. Also the λ_i's are distinct. Then $nl(f) = 2^{n-1} - (2^r - 2 \times nlmax(r))2^{k-1}$.*

Proof : By construction f is a concatenation of linear couples λ_i, λ_i^c from $UF_k(m+1)$. Let l be in F_n and is a concatenation of linear couple μ, μ^c for some μ in L_k. If $\lambda_i \neq \mu$ for any i, then $d(f,l) = 2^{n-1}$. On the other hand if $\lambda_i = \mu$, for some i, then $d(f,l) = (2^{n-k} - 2^r)2^{k-1} + d(g_i\$\lambda_i, \eta_i\$\mu)$, for some η_i in F_r. From Lemma 1, $d(g_i\$\lambda_i, \eta_i\$\mu) = 2^k d(g_i, \eta_i)$ and so $d(f,l) = 2^{n-1} - (2^r - 2d(g_i,\eta_i))2^{k-1}$. Since g_i is a maximum nonlinear function on r variables, $nl(g_i) = nlmax(r)$ and so $nlmax(r) \leq d(g_i,\eta_i) \leq 2^r - nlmax(r)$. Hence we get, $2^{n-1} - (2^r - 2nlmax(r))2^{k-1} \leq d(f,l) \leq 2^{n-1} + (2^r - 2nlmax(r))2^{k-1}$. This gives $nl(f) = 2^{n-1} - (2^r - 2 \times nlmax(r))2^{k-1}$. □

3.3 Fractional Nonlinearity and its Effect

In the previous section we considered the case when each linear couple is repeated t times, where t is a power of 2. In general it might be advantageous to repeat a linear couple t times even when t is not a power of 2. To see the advantage we need to introduce the notion of nonlinearity of "fractional functions". Let $2^{r-1} < t \leq 2^r$. Given a string l of length 2^r, let $First(l,t)$ be a string consisting of the first t bits of l. The (fractional) nonlinearity of a string g of length t is denoted by $fracnl(g)$ and defined as $fracnl(g) = \min_{l \in F_r} d(First(l,t), g)$. Given a positive integer t, the maximum possible fractional nonlinearity attainable by any string of length t is denoted by $Fracnlmax(t)$ and defined as $Fracnlmax(t) = \max_{g \in \{0,1\}^t} fracnl(g)$. When $t = 2^r$, $Fracnlmax(t) = nlmax(r)$. Also $Fracnlmax(2^r + 1) = nlmax(r)$ and $Fracnlmax(2^r - 1) = nlmax(r) - 1$. It is clear that $Fracnlmax(t)$ is a nondecreasing function. If a linear couple is repeated 2^r times, then by Theorem 4, the fall in nonlinearity is by a factor of $(2^r - 2 \times nlmax(r))$. Motivated by this we define $\mathit{Effect}(t) = t - 2Fracnlmax(t)$ as the factor by which nonlinearity falls when a linear couple is repeated t times. In the construction of a function f in $\Gamma_1(n,k,m)$ if the distinct linear couples are repeated $t_1, \ldots, t_p$ times then $nl(f) = \min_{1 \leq i \leq p}(2^{n-1} - 2^{k-1}\mathit{Effect}(t_i))$. The interesting point about $\mathit{Effect}(t)$ is that it is not a monotone increasing function. An important consequence of this is that the nonlinearity may fall by a lesser amount when a linear couple is repeated more times.

1. $\mathit{Effect}(2^r - 1) = 2^r - 1 - 2(nlmax(r) - 1) = 2^r + 1 - 2nlmax(r) = \mathit{Effect}(2^r + 1) > \mathit{Effect}(2^r)$.
2. $\mathit{Effect}(2^r) \geq \mathit{Effect}(2^{r-1})$ and $\mathit{Effect}(2^r) > \mathit{Effect}(2^{r-2} + 1)$.
3. If r is odd, $\mathit{Effect}(2^r) > \mathit{Effect}(2^{r-1} + 1)$.
4. If r is even, $\mathit{Effect}(2^r) < \mathit{Effect}(2^{r-1} + 1)$, assuming $nlmax(r-1) = 2^{r-2} - 2^{\frac{r-2}{2}}$. If $r - 1 \geq 15$, the calculations are more complicated because of the existence of functions in [9].

One can also define fractional nonlinearity and $\mathit{Effect}()$ for balanced strings (provided t is even). We believe that the idea of fractional nonlinearity is important and to the best of our knowledge it has not appeared in the literature before.

3.4 Use of All Linear Functions

Here we show how to extend the set Γ_1. To construct a function $f \in \Gamma_1(n,k,m)$ we have to concatenate affine functions in $UF_k(m+1)$. However, it is possible to use all the affine functions in F_k to construct n-variable, m-resilient functions. Let l be a function in L_k which is nondegenerate on r $(1 \leq r \leq m)$ variables. Then ll^c is 1-resilient and repeating this procedure $m-r+1$ times one can construct a function g in $UL_{k+m-r+1}(m+1)$. The linear couple g, g^c can then be used in the construction of m-resilient functions. The importance of this technique lies in the fact that it helps in reducing the repetition factor of linear couples in $UF_k(m+1)$. However, one should be careful in ascertaining that the loss in nonlinearity due to the use of affine functions from $DF_k(m)$ does not exceed the loss in repeating linear couples from $UF_k(m+1)$. In Theorem 9 and Theorem 10, we show examples of how this technique can be used to construct optimized functions.

3.5 Use of Nonlinear Resilient Function

Here also we extend Γ_1, though in a different way. Corollary 1 places a restriction on the value of k in $\Gamma_1(n,k,m)$ for optimized functions : $k = m+2$. This in turn restricts the number of linear couples to be used in the construction to $m+3$, thus increasing the repetition factor. However, if we allow $k > m+2$, the problem is that the degree will fall. To compensate this we use one nonlinear m-resilient function on k variables and having degree $k-m-1$ with the maximum possible nonlinearity. By Theorem 3, the overall function will have degree $n-m-1$ but the number of available linear couples increases to $| UF_k(m+1) | > | UF_{m+2}(m+1) |$. This reduces the repetition factor. In Subsection 5.2, we outline a design procedure for optimized functions based on this idea. Also in Section 4, we show how all the above ideas can be combined to disprove the conjecture of Pasalic and Johansson [8] for optimized functions.

4 Nonlinearity of Resilient Functions

A proper subset S of Γ_1 was considered in [2], where only concatenation of linear (not affine) functions were used to construct functions in Γ_1. In particular, it was shown in [2] that the maximum possible nonlinearity for n-variable resilient functions in S is $2^{n-1} - 2^{\lfloor \frac{n}{2} \rfloor}$. In a more recent paper, Pasalic and Johansson [8] have shown that the maximum possible nonlinearity of 6-variable, 1-resilient functions is 24. The same paper conjectured that the maximum possible nonlinearity of n-variable, 1-resilient functions is $2^{n-1} - 2^{\lfloor \frac{n}{2} \rfloor}$. *We provide infinite counterexamples to this conjecture.* In fact, we show that given a fixed order of resiliency m, one can construct n-variable functions which are m-resilient and have nonlinearity greater that $2^{n-1} - 2^{\lfloor \frac{n}{2} \rfloor}$. Moreover, the conjecture is disproved for optimized functions as well as for functions in Γ_1.

Theorem 5. *Let m be a fixed positive integer. Then there are infinitely many odd positive integers n_o (resp. even positive integers n_e), such that one can construct functions f of n_o (resp. n_e) variables which are m-resilient and $nl(f) > 2^{n_o-1} - 2^{\frac{n_o-1}{2}}$ (resp $nl(f) > 2^{n_e-1} - 2^{\frac{n_e}{2}}$).*

Proof : First note that if we can prove the result for odd number of variables and for all $m \geq 1$, then the result is proved for even number of variables and all $m > 1$. We also need a proof for even number of variables and $m = 1$. These we proceed to do via the following sequence of results. □

Theorem 6. *Let m be a fixed positive integer. Choose ϵ, n_1, n_2 such that (a) $n_1 + n_2$ is even, (b) $n_2 - n_1 = \epsilon n_1 = 2k$, for some $k \geq 4$, (c) $\frac{1}{2} \leq \epsilon \leq 1$, (d) $\binom{n_1}{m} + \ldots + \binom{n_1}{0} \leq 2^{(1-\epsilon)n_1} - 1$. Then it is possible to construct an m-resilient function on $n = n_1 + n_2 + 15$ variables having nonlinearity greater than $2^{n-1} - 2^{\frac{n-1}{2}}$. Moreover, it is possible to construct such functions having maximum degree $n - m - 1$.*

Proof : First we construct an m-resilient function g on $q = n_1 + n_2$ variables having nonlinearity $nl(g) = 2^{q-1} - 2^{\frac{q}{2}-1} - 2^{n_1-1}$. Then we let $f = h\$g$, where h is a function on 15 variables having nonlinearity $nl(h) = 16276 = 2^{14} - 108$. This h can be constructed using the method of [9]. The function f is m-resilient (from Proposition 1) and the overall nonlinearity of f is obtained as $nl(f) = nl(h)2^q + nl(g)2^{15} - 2nl(h)nl(g)$. Simplifying, we get $nl(f) = 2^{q+14} - 108(2^{\frac{n_2-n_1}{2}} + 1)2^{n_1}$. Using $n_2 - n_1 = 2k$, this simplifies to $nl(f) = 2^{q+14} - 108(2^k + 1)2^{n_1}$. On the other hand, $\frac{n-1}{2} = 7 + n_1 + k$. Since $108(2^k + 1) < 2^{7+k}$ for $k \geq 4$, we get $nl(f) > 2^{n-1} - 2^{\frac{n-1}{2}}$. Thus if we show how to construct g then the proof will be complete.

The function g is in $\Gamma_1(q, n_1, m)$ and is constructed in a way similar to that in Theorem 4. Since g is to be m-resilient we are restricted to using linear couples from $UF_{n_1}(m+1)$ and there are $2^{n_1} - p$ linear couples in $UF_{n_1}(m+1)$, where, $p = \binom{n_1}{m} + \ldots + \binom{n_1}{0} \leq 2^{(1-\epsilon)n_1} - 1$ These have to be used to fill up 2^{n_2} slots and so the maximum repetition factor for each linear couple is $\lceil \frac{2^{n_2}}{p} \rceil = 2^{n_2-n_1} + 1$ by choice of the parameters ϵ, n_1, n_2. Thus each linear couple is repeated either $2^{n_2-n_1} + 1$ times or $2^{n_2-n_1}$ times. Suppose a linear couples are repeated $2^{n_2-n_1} + 1$ times and b linear couples are repeated $2^{n_2-n_1}$ times. Let $\lambda_1, \ldots, \lambda_a$ be distinct linear functions from $UL_{n_1}(m+1)$ and $\mu_1, \ldots, \mu_b$ be distinct linear functions from $UL_{n_1}(m+1)$ which are also distinct from $\lambda_1, \ldots, \lambda_a$. Let $\alpha_1, \ldots, \alpha_a$ and $\beta_1, \ldots, \beta_b$ be bent functions of $n_2 - n_1$ variables. The function g is a concatenation of the following sequence of functions: $\alpha_1\$\lambda_1, \ldots, \alpha_a\$\lambda_a, \beta_1\$\mu_1, \ldots, \beta_b\$\mu_b, \lambda_1, \ldots, \lambda_a$.
Using the same idea as in the proof of Theorem 4 one can show that $nl(g) = 2^{q-1} - (2^{\frac{q}{2}-1} + 2^{n_1-1})$. This completes the proof of the first part of the Theorem.

To obtain maximum possible degree $n - m - 1$ in the above construction we do the following. In the constructed function f, replace the last 2^{n_1} bits by an n_1-variable, m-resilient optimized function. Using Theorem 3 it follows that f becomes optimized. Also nonlinearity remains greater than $2^{n-1} - 2^{\frac{n-1}{2}}$. □

Example: For $m = 1$, choose $n_1 = 10, n_2 = 16$ and $\epsilon = \frac{3}{5}$. This provides 1-resilient, 41-variable functions f with nonlinearity $2^{40} - 2^{20} + 52 \times 2^{10} > 2^{40} - 2^{20}$. To obtain maximum degree 39, replace the last $2^{n_1} = 1024$ bits of such a function f by a nonlinear 10-variable, 1-resilient, degree 8 function (see Theorem 9 later). This provides $(41, 1, 39, x)$ function with $x > 2^{40} - 2^{20} + 51 \times 2^{10}$. For $m = 2$, choose $n_1 = 16, n_2 = 24$ and $\epsilon = \frac{1}{2}$. This provides 2-resilient, 55-variable functions with nonlinearity $2^{54} - 2^{27} + 212 \times 2^{16}$. As before we can obtain $(55, 2, 52, y)$ functions with $y > 2^{54} - 2^{27} + 211 \times 2^{16}$.

Corollary 2. *The functions f and g constructed in the proof of Theorem 6 belong to Γ_1.*

Corollary 3. *For odd n, let f be an n-variable, m-resilient function having $nl(f) > 2^{n-1} - 2^{\frac{n-1}{2}}$ and let g be a $2k$-variable bent function. Then $f\$g$ is an $n + 2k$-variable, m-resilient function with $nl(f\$g) > 2^{n+2k-1} - 2^{\frac{n+2k-1}{2}}$. Consequently, if Theorem 6 holds for some odd n_0, then it also holds for all odd $n > n_0$.*

To prove Theorem 5, the only case that remains to be settled is $m = 1$ for even number of variables.

Theorem 7. *For each even positive integer $n \geq 12$, one can construct 1-resilient functions f of n-variables having $nl(f) > 2^{n-1} - 2^{\frac{n}{2}}$. Moreover, f is in Γ_1.*

Proof : Let $n = 2p$ and consider the set $\Gamma_1(2p, p-1, 1)$. We show how to construct a function in $\Gamma_1(2p, p-1, 1)$ having nonlinearity $2^{2p-1} - 3 \times 2^{p-2}$ which is greater than $2^{2p-1} - 2^p$. Since we are constructing functions in $\Gamma_1(2p, p-1, 1)$ we have to use linear couples from the set $UF_{p-1}(2)$. The number of available linear couples is $2^{p-1} - p$. Since there are 2^{p+1} slots to be filled the maximum repetition factor is $\lceil \frac{2^{p+1}}{2^{p-1}-p} \rceil = 5$. Thus the linear couples are to be repeated either 5 times or 4 times. Then as in the construction of g in the proof of Theorem 6, one can construct a function f having nonlinearity $2^{2p-1} - 3 \times 2^{p-2}$. Since f is a concatenation of linear couples from $UF_{p-1}(2)$ it follows that f is 1-resilient. □

The above constructions can be modified to get optimized functions also. We illustrate this by providing construction methods for $(2p, 1, 2p-2, x)$ functions with $x > 2^{2p-1} - 2^p$ for $p \geq 6$. The constructed functions are not in Γ_1.

Theorem 8. *For $p \geq 6$, it is possible to construct $(2p, 1, 2p-2, x)$ functions with x greater than $2^{2p-1} - 2^p$.*

Proof : As in the proof of Theorem 7, we write $2p = (p+1) + (p-1)$ and try to fill up 2^{p+1} slots using 1-resilient $(p-1)$-variable functions to construct a function $f \in \Omega_{2p}$. As before we use linear couples from $UF_{p-1}(2)$, but here we use these linear couples to fill up only $2^{p+1} - 1$ slots. The extra slot is filled up by a balanced $(p-1, 1, p-3, y)$ function g. The repetition factor for each linear couple is again at most 5 and the construction is again similar to Theorem 6. The nonlinearity is calculated as follows. Let l be in F_{2p}. The function g contributes

at least y to $d(f,l)$. Ignoring the slot filled by g, the contribution to $d(f,l)$ from the linear couples is found as in Theorem 4. This gives the following inequality $2^{2p-1}-2^p+y \leq d(f,l) \leq 2^{2p-1}-y < 2^{2p-1}+2^p-y$. Hence $d(f,l) = 2^{2p-1}-2^p+y$. An estimate of y is obtained as follows. If $p-1$ is odd we use Theorem 10. If $p-1$ is even, then we recursively use the above construction. □

It is also possible to construct 1-resilient, 10-variable functions having nonlinearity $484 > 2^9 - 2^5$. This construction for optimized function combines all the construction ideas given in Section 3. The result disproves the conjecture of Pasalic and Johansson [8] for 10-variable functions.

Theorem 9. *It is possible to construct* $(10, 1, 8, 484)$ *functions.*

Proof : We write $10 = 6 + 4$ and concatenate affine functions of 4 variables to construct the desired function f. However, if we use only affine functions then the degree of f is less than 8. To improve the degree we use exactly one nonlinear $(4, 1, 2, 4)$ function h. By Theorem 3, this ensures that the degree of the resulting function is 8. This leaves $2^6 - 1$ slots to be filled by affine functions of 4 variables. If we use only functions from $UF_4(2)$, then the maximum repetition factor is 6 and the resulting nonlinearity is low. Instead we repeat the 11 linear couples in $UF_4(2)$ only 5 times each. This leaves $2^6 - 1 - 55 = 8$ slots to be filled up. We now use functions from $F_4(1)$. However, these are not resilient. But for $l \in F_4(1)$, ll^c is resilient. Since there are exactly 4 functions in $F_4(1)$ and each is repeated exactly 2 times, this uses up the remaining 8 slots. Let $g_1, \ldots, g_{11}$ be bent functions on 2 variables and let $\lambda_1, \ldots, \lambda_{11}$ be the 11 linear functions in $UL_4(2)$. Also let $\mu_1, \ldots, \mu_4$ be the 4 linear functions in $L_4(1)$. Then the function f is concatenation of the following sequence of functions: $g_1\$\lambda_1, \ldots, g_{11}\$\lambda_{11}, \mu_1\mu_1^c, \ldots, \mu_4\mu_4^c, \lambda_1, \ldots, \lambda_{11}h$. The nonlinearity calculation of f is similar to the previous proofs. Let l be in F_{10}. The worst case occurs when l is concatenation of λ_i and λ_i^c for some $1 \leq i \leq 11$. In this case $d(f,l) = (2^6 - 1 - 5)2^3 + 2^4 + 4 = 484$. □

The functions constructed by the methods of Theorem 9 and Theorem 8 are not in Γ_1 and do not require the use of a 15-variable nonlinear function from [9]. *It is important to note that the nonlinearity of functions constructed using Theorem 9 cannot be achieved using concatenation of only affine functions. Moreover, in this construction it is not possible to increase the nonlinearity by relaxing the optimality condition on degree, i.e., allowing the degree to be less than* 8.

The maximum possible nonlinearity of Boolean functions is equal to the covering radius of first order Reed-Muller codes. Patterson and Weidemann showed that for odd $n \geq 15$ the covering radius and hence the maximum possible nonlinearity of an n-variable function exceeds $2^{n-1} - 2^{\frac{n-1}{2}}$. Seberry et al [14] showed that for odd $n \geq 29$, it is possible to construct balanced functions with nonlinearity greater than $2^{n-1} - 2^{\frac{n-1}{2}}$. Theorem 6 establishes a similar result for optimized resilient functions of odd number of variables n for $n \geq 41$.

5 Construction of Optimized Resilient Functions

Here we consider construction of optimized functions. We start with the following important result.

Theorem 10. *It is possible to construct (a)* $(2p+1, 0, 2p, 2^{2p} - 2^p)$ *functions for* $p \geq 1$, *(b)* $(2p+1, 1, 2p-1, 2^{2p} - 2^p)$ *functions for* $p \geq 2$, *(c)* $(2p, 1, 2p-2, 2^{2p-1} - 2^p)$ *functions for* $p \geq 2$ *and (d)* $(2p, 2, 2p-3, 2^{2p-1} - 2^p)$ *functions for* $p \geq 3$.

Proof : We present only the constructions (proofs are similar to Section 4).
(a) If $p = 1$, let $f = X_3 \oplus X_1X_2$. For $p \geq 2$ consider the following construction. Let $\lambda_1, \lambda_2, \lambda_3$ be the functions in $UL_2(1)$ and λ_4 the (all zero) function in $L_2(0)$. Let h_1 be a bent function on $2p-2$ variables, h_2 be a maximum nonlinear balanced function on $2p-3$ variables. If $p = 2$ let h_3, h_4 be strings of length 1 each and for $p \geq 3$ let h_3, h_4 be maximum nonlinear strings of length $2^{2p-4} + 1$ and $2^{2p-4} - 1$ respectively. Let f be a concatenation of the following sequence of functions: $h_1\$\lambda_1, h_2\$\lambda_4, h_3\$\lambda_2, h_4\$\lambda_3$. It can be shown that f is a $(2p+1, 0, 2p, 2^{2p} - 2^p)$ function.
(b) Let $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ be the functions in $UL_3(2)$ and μ_1, μ_2, μ_3 the functions in $L_3(1)$. For $p = 2$, let $f = \lambda_1\lambda_2\lambda_3\lambda_4$. For $p = 3$, let f be the concatenation of the following sequence of functions.
$h_1\$\lambda_1, h_2\$\lambda_2, \mu_1\mu_1^c, \mu_2\mu_2^c, \mu_3\mu_3^c, \lambda_3, \lambda_4$, where h_1 and h_2 are 2-variable bent functions. For $p \geq 4$, we have the following construction. Let $g_i = \mu_i\mu_i^c$, for $1 \leq i \leq 3$,. Let h_1, h_2 be bent functions of $2p-4$ variables, h_3, h_4, h_5 be bent functions of $2p-6$ variables and h_6, h_7 be two strings of lengths $2^{2p-6} + 1$ and $2^{2p-6} - 1$ and (fractional) nonlinearity $nlmax(2p-6)$ and $nlmax(2p-6) - 1$ respectively. Let f be a concatenation of the following sequence of functions.
$h_1\$\lambda_1, h_2\$\lambda_2, h_3\$g_1, h_4\$g_2, h_5\$g_3, h_6\$\lambda_3, h_7\$\lambda_4$. It can be shown that f is a $(2p+1, 1, 2p-1, 2^{2p} - 2^p)$ function.
(c) and (d) follow from (a) and (b) on noting that if f is a $(2p+1, m, 2p-m, x)$ function then ff^c is a $(2p+2, m+1, 2p-m, 2x)$ function. □

Note that item (a), (b) of Theorem 10 can also be proved using different techniques by modifying a special class of bent functions. See [13] for the detailed construction methods.

5.1 Method Using Direct Sum with a Nonlinear Function

Here we consider the set $\Gamma_1(n, k, m)$ and show how to construct optimized functions with very high nonlinearities in this set. We build upon the idea described in Subsection 3.2. Since we consider optimized functions, Corollary 1 determines $k = m+2$ and at least one variable in $\{X_k, \ldots, X_1\}$ must occur in odd number of the f_i's. We recall from Subsection 3.3, that $Fracnlmax(2^r - 1) = nlmax(r) - 1$, $Fracnlmax(2^r + 1) = nlmax(r)$, $Fracnlmax(2^r) = nlmax(r)$ and $Effect(t) = t - 2Fracnlmax(t)$.

Given n and m, we construct an optimized function f in $\Gamma_1(n, m+2, m)$. We define a variable *template* to be a list of the form $(s, (s_1, t_1), \ldots, (s_k, t_k))$,

where $\sum_{j=1}^{k} s_j = s$ and $\sum_{j=1}^{k} s_j t_j = 2^{n-m-2}$. The value s is the number of distinct linear couples to be used from the set $UF_{m+2}(m+1)$ and for each j, $(1 \leq j \leq k)$, s_j linear couples are to be used t_j times each. While constructing *template* one has to be careful in ascertaining that at least one variable occurs in an odd number of functions overall. This gives rise to the various cases in Algorithm A. Since an n-variable, $(n-2)$-resilient function must have degree 1 and hence be linear, we consider only the cases $1 \leq m < n-2$.

ALGORITHM A

input: (n, m) with $1 \leq m < n-2$.

output: A $(n, m, n-m-1, x)$ function f. We determine x in Theorem 11.

BEGIN

1. Let $p = m+3$ and $2^{r-1} < \lceil \frac{2^{n-m-2}}{m+3} \rceil \leq 2^r$. Let $i = p - 2^{n-m-2-r}$, i.e., $(p-i)2^r = 2^{n-m-2}$. Now several cases arise.

2. $r = 0, i > 0$: Here f is the concatenation of $(p-i-1)$ functions containing X_1 and the one function not containing X_1 from the set $UL_{m+2}(m+1)$. Output f and STOP.

3. $r = 0, i = 0$, $m+2$ is odd: $template = (p, (p, 1))$.

4. $r > 0, i = 0$, r is even: $template = (p, (p-2, 2^r), (1, 2^r + 1), (1, 2^r - 1))$.

5. $r > 0, i = 0$, r is odd: $template = (\frac{p}{2} + 2, (\frac{p}{2} - 1, 2^{r+1}), (1, 2^r), (1, 2^{r-1} + 1), (1, 2^{r-1} - 1))$.

6. $r = 1, i > 0$: $template = (p - i + 1, (p - i - 1, 2), (2, 1))$.

7. $r = 2, i > 1$: $template = (p - i + 2, (p - i - 1, 4), (1, 2), (2, 1))$.

8. $r \geq 2, i = 1$, r is even: $template = (p, (p-2, 2^r), (1, 2^{r-1} + 1), (1, 2^{r-1} - 1))$.

9. $r \geq 2, i = 1$, r is odd:
$template = (\frac{p+3}{2}, (\frac{p-3}{2}, 2^{r+1}), (1, 2^r), (1, 2^{r-1} + 1), (1, 2^{r-1} - 1))$.

10. $r > 2, i > 1$: $template = (p - i + 2, (p - i - 1, 2^r), (1, 2^{r-1}), (1, 2^{r-2} + 1), (1, 2^{r-2} - 1))$.

11. Let $template = (s, (s_1, t_1), \ldots, (s_k, t_k))$. For each j, choose $l_j^1, \ldots, l_j^{s_j}$ to be distinct linear functions from $UL_{m+2}(m+1)$ and $g_j^1, \ldots, g_j^{s_j}$ to be strings of length t_j and having maximum possible nonlinearity. (Note that the g's may be fractional strings.) Then f is the concatenation of the following sequence of functions
$g_1^1 \$ l_1^1, \ldots, g_1^{s_1} \$ l_1^{s_1}, g_2^1 \$ l_2^1, \ldots, g_2^{s_2} \$ l_2^{s_2}, \ldots, g_k^1 \$ l_k^1, \ldots, g_k^{s_k} \$ l_k^{s_k}$.

END.

Theorem 11. *Algorithm A constructs a $(n, n-m-1, m, x)$-function f in $\Gamma_1(n, m+2, m)$, where the values of x in different cases (corresponding to the line numbers of Algorithm A) are as follows. (2) $2^{n-1} - 2^{m+1}$ (3) $2^{n-1} - 2^{m+1}$ (4) $2^{n-1} - 2^{m+1} Effect(2^r + 1)$ (5) $2^{n-1} - 2^{m+1} Effect(2^{r+1})$ (6) $2^{n-1} - 2^{m+2}$ (7) $2^{n-1} - 2^{m+2}$ (8) $2^{n-1} - 2^{m+1} Effect(2^{r-1} + 1)$ (9) $2^{n-1} - 2^{m+1} Effect(2^{r+1})$ (10) $2^{n-1} - 2^{m+1} Effect(2^r)$.*

Example: Using Algorithm A it is possible to construct $(9, 3, 5, 224)$ functions having template $= (6, (3, 4), (1, 2), (2, 1))$.

5.2 Use of Nonlinear Resilient Function

Here we use the idea of Subsection 3.5 to provide a construction method for optimized resilient functions. The constructed functions are not in Γ_1.

Let $nla(n,m)$ be the nonlinearity of a function obtained by Algorithm A with (n,m) as input. Similarly, let $nlb(n,m)$ be the highest nonlinearity of a function obtained using Algorithm B (described below) on input (n,m) and ranging c from 1 to $n-m-2$. We obtain an expression for $nlb(n,m)$ in Theorem 12. Let $nlx(n,m)$ be the maximum of $nla(n,m)$ and $nlb(n,m)$.

ALGORITHM B

input : (n,m,c), with $1 \leq m < n-2$ and $1 \leq c \leq n-m-2$.
output : A balanced $(n,m,n-m-1,x_c)$ function f_c. The value of x_c is given in Lemma 5.
BEGIN
1. If $n <= 5$, use Algorithm A with input (n,m) to construct a function f. Output f and stop.
2. Let $p = \binom{m+c+2}{m+1} + \ldots + \binom{m+c+2}{m+c+2}$ and $2^{r-1} < \lceil \frac{2^{n-(m+c+2)}}{p} \rceil \leq 2^r$. Let $i = p - 2^{n-(m+c+2)-r}$, i.e., $(p-i)2^r = 2^{n-(m+c+2)}$.
3. $i=0, r=0$: $template = (p-1,(p-1,1))$.
4. $i>0, r=0$: $template = (p-i-1,(p-i-1,1))$.
5. $i>0, r=1$: $template = (p-i,(p-i-1,2),(1,1))$.
6. $i=0, r>0$, r is even: $template = (p,(p-1,2^r),(1,2^r-1))$.
7. $i=0, r>0$, r is odd:
$template = (\frac{p}{2}+2,(\frac{p}{2}-1,2^{r+1}),(1,2^r),(1,2^{r-1}),(1,2^{r-1}-1))$.
8. $i>0, r=2$: $template = (p+1,(p-1,4),(1,2),(1,1))$.
9. $i=1, r>2$, r is even: $template = (p,(p-2,2^r),(1,2^{r-1}),(1,2^{r-1}-1))$.
10. $i=1, r \geq 2$, r is odd:
$template = (\frac{p+3}{2},(\frac{p-3}{2},2^{r+1}),(1,2^r),(1,2^{r-1}),(1,2^{r-1}-1))$.
11. $i>1, r>2$:
$template = (p-i+2,(p-i-1,2^r),(1,2^{r-1}),(1,2^{r-2}),(1,2^{r-2}-1))$.
12. Using *template* and linear couples from $UF_{m+c+2}(m+1)$, we first build a string f_1 as in Algorithm A. Then the function f_c is $f_1 g$, where g is a $(m+c+2, m, 1+c, y)$ function, where $y = nlx(m+c+2, m)$.
END.

Note that the use of the function $nlx(n,m)$ makes Algorithm B a recursive function. Let the nonlinearity of a function f_c constructed by Algorithm B on input (n,m,c) be $nlbs(n,m,c)$.

Lemma 5. *Let f_c be constructed by Algorithm B. Then f_c is a balanced $(n, m, n-m-1, x_c)$ function, where $x_c = nlbs(n,m,c)$ and the values of x_c in the different cases (corresponding to the line numbers of Algorithm B) are as follows : (3) $2^{n-1} - 2^k + y$ (4) $2^{n-1} - 2^k + y$ (5) $2^{n-1} - 3 \times 2^{k-1} + y$ (6) $2^{n-1} - (1 + Effect(2^r - 1))2^{k-1} + y$ (7) $2^{n-1} - (1 + Effect(2^{r+1}))2^{k-1} + y$ (8) $2^{n-1} - 3 \times 2^{k-1} + y$ (9) $2^{n-1} - (1 + Effect(2^{r-1} - 1))2^{k-1} + y$ (10) $2^{n-1} - (1 + Effect(2^{r+1}))2^{k-1} + y$ (11) $2^{n-1} - (1 + Effect(2^r))2^{k-1} + y$ where $k = m+c+2$, $y = nlx(m+c+2, m)$.*

Algorithm B is used iteratively over the possible values of c from 1 to $n-m-2$ and the function with the best nonlinearity is chosen. The maximum possible nonlinearity $nlb(n,m)$ obtained by using Algorithm B in this fashion is given by the following theorem.

Theorem 12. $nlb(n,m) \geq max_{1 \leq c \leq n-m-2} nlbs(n,m,c)$.

Example: Using Algorithm B one can construct $(9,2,6,232)$ *functions in* $\Gamma(9,5,2)$ *having template* $(15,(15,1))$ *and a* $(5,2,2,8)$ *function* g *is used to fill the* $16th$ *slot.*

6 Comparison to Existing Research

Here we show the power of our techniques by establishing the superiority of our results over all known results in this area.

The best known results for $Type-A$ approach follows from the work of [2]. However, they considered only a proper subset S of Γ_1 and obtained a bound of $2^{n-1} - 2^{\lfloor \frac{n}{2} \rfloor}$ on the nonlinearity of resilient functions. Also in [8], it was conjectured that this is the maximum possible nonlinearity of resilient functions. All the results in Section 4 provide higher nonlinearities than this bound. In particular, this bound is broken and hence the conjecture is disproved for the set Γ_1 as well as for optimized functions.

n	$m=1$			$m=2$			$m=3$			$m=4$		
	[7]	[8]	Our	[7]	[4]	Our	[7]	*nla*	*nlb*	[7]	*nla*	*nlb*
8	108	112	112^a	88	-	112^b	80	96	80	32	96	32
9	220	236	240^b	216	224	232^e	176	224	208	160	192	160
10	476	480	484^c	440	-	480^b	432	448	464	352	448	416
11	956	976	992^b	952	-	984^e	880	960	944	864	896	928
12	1980	-	1996^d	1912	-	1984^b	1904	1920	1968	1760	1920	1888

a:Algorithm A; b: Theorem 10; c: Theorem 9; d: Theorem 8; e: Algorithm B.

For the $Type-B$ approach the best known results follow from the work of [15, 4, 7, 8, 18]. In [4], exhaustive search techniques are used to obtain $(5,0,4,12)$ and $(7,0,6,56)$ functions. For 9 variables, they could only obtain $(9,0,7,240)$ functions and not $(9,0,8,240)$ functions. Also such techniques cannot be used for large number of variables. In contrast, Theorem 10 can be used to construct $(2p+1,0,2p,2^{2p}-2^p)$ functions for all $p \geq 1$ and hence is clearly superior to the results of [4].

In the Table, we compare the nonlinearities of optimized $(n,m,n-m-1,x)$ functions. The columns *nla* and *nlb* are the nonlinearities obtained by Algorithm A and Algorithm B respectively. We do not compare results with [15], since it is clear that Algorithm A significantly improves on the lower bound on nonlinearity obtained in [15].

The table clearly shows the superiority of our method compared to the previous methods. Also it can be checked that the nonlinearities obtained in Theorem 11 are better than those obtained in [7] for all orders of resiliency. We

can construct $(9, 3, 5, 224)$ functions and $(9, 2, 6, 232)$ functions using Algorithm A and Algorithm B respectively. These improve over the $(9, 2, 6, 224)$ functions of [4] both in terms of order of resiliency and nonlinearity.

7 Nonlinearity of Balanced Functions

In this section we discuss the nonlinearity and algebraic degree for balanced functions. Patterson and Wiedemann [9] constructed 15-variable functions with nonlinearity 16276 and weight 16492. Seberry, Zhang and Zheng [14] used such functions to construct balanced functions with nonlinearity greater than $2^{n-1} - 2^{\frac{n-1}{2}}$ for odd $n \geq 29$. In [14], there was an unsuccessful attempt to construct balanced 15-variable functions having nonlinearity greater than $16256 = 2^{14} - 2^7$. First let us provide the following two technical results.

Proposition 2. *Let $f \in \Omega_n$ and $f = f_1 f_2$, where $f_1, f_2 \in \Omega_{n-1}$. If $wt(f)$ is odd then algebraic degree of f is n. Moreover, if both $wt(f_1)$ and $wt(f_2)$ are odd then the algebraic degree of f is $n - 1$.*

Proposition 3. *Given a balanced function $f \in \Omega_n$ with $nl(f) = x$, one can construct balanced $f' \in \Omega_n$ with $nl(f') \geq x - 2$ and $deg(f') = n - 1$.*

Now, we identify an important result which is the first step towards constructing a balanced 15-variable function with nonlinearity greater than 16256.

Proposition 4. *It is possible to construct $f \in \Omega_{15}$ with nonlinearity 16276 and weight 16364.*

Proof : Consider a function $f_1 \in \Omega_{15}$ with $nl(f_1) = 16276$ and $wt(f_1) = 16492$. From [9], we know that there are 3255 linear functions in L_{15} at a distance 16364 from f_1. Let l be one of these 3255 linear functions. Define $f = f_1 \oplus l$. Then $f \in \Omega_{15}$, $nl(f) = nl(f_1) = 16276$ and $wt(f) = wt(f_1 \oplus l) = d(f_1, l) = 16364$. □

Next we have the following randomized heuristic for constructing highly nonlinear balanced functions for odd $n \geq 15$.

Algorithm 1 : RandBal(n)

1. Let f be a function constructed using Proposition 4. Let $n = 2k + 15$, $k \geq 0$ and let $F \in \Omega_n$ be defined as follows. For $k = 0$, take $F = f$, and for $k > 0$, take $F = f(X_1, \ldots, X_{15}) \oplus g(X_{16}, \ldots, X_n)$, where $g \in \Omega_{2k}$ is a bent function. Note that $nl(F) = 2^{n-1} - 2^{\frac{n-1}{2}} + 20 \times 2^k$ and $wt(F) = 2^{n-1} - 20 \times 2^k$.
2. Divide the string F in Ω_n into 20×2^k equal contiguous substrings, with the last substring longer than the rest.
3. In each substring choose a position with 0 value uniformly at random and change that to 1. This generates a balanced function $F_b \in \Omega_n$.
4. If $nl(F_b) > 2^{n-1} - 2^{\frac{n-1}{2}}$, then report. Go to step 1 and continue.

We have run this experiment number of times and succeeded in obtaining plenty of balanced functions with nonlinearities $2^{14} - 2^7 + 6$, $2^{16} - 2^8 + 18$, $2^{18} - 2^9 + 46$ and $2^{20} - 2^{10} + 104$ respectively for 15, 17, 19 and 21 variables. It is possible to distribute the 0's and 1's in the function in a manner (changing step

2, 3 in Algorithm 1) such that weight of the upper and lower half of the function are odd. This provides balanced functions with maximum algebraic degree $(n-1)$ and the same nonlinearity as before. Note that, running Algorithm 1 for large n is time consuming. However, we can extend the experimental results in a way similar to that in [9]. Consider a bent function $g(Y_1, \ldots, Y_{2k}) \in \Omega_{2k}$ and $f(X_1, \ldots, X_{21})$ with nonlinearity $2^{20} - 2^{10} + 104$ as obtained from Algorithm RandBal(). Let $h \in \Omega_{21+2k}$ such that $h = g \oplus f$. Then it can be checked that $nl(h) = 2^{20+2k} - 2^{10+k} + 104 \times 2^k$. These functions can be modified to get algebraic degree $(n-1)$ as in Proposition 3. Thus we get the following result.

Theorem 13. *One can construct balanced Boolean functions on $n = 15 + 2k$ $(k \geq 0)$ variables with nonlinearity greater than $2^{n-1} - 2^{\frac{n-1}{2}}$. Moreover, such functions can have algebraic degree $(n-1)$.*

Dobbertin [3] provided a recursive procedure for modifying a general class of bent functions to obtain highly nonlinear balanced Boolean functions on even number of variables. A special case of this procedure which modifies Maiorana-McFarland class of bent functions was provided in [14]. For even n, it is conjectured in [3] that the maximum value of nonlinearity of balanced functions, which we denote by $nlbmax()$, satisfies the recurrence: $nlbmax(n) = 2^{n-1} - 2^{\frac{n}{2}} + nlbmax(\frac{n}{2})$.

We next provide a combined interlinked recursive algorithm to construct highly nonlinear balanced functions for both odd and even n. Note that for even number of variables, Algorithm 2 uses a special case of the recursive construction in [3]. Further we show how to obtain maximum algebraic degree. The input to this algorithm is n and the output is balanced $f \in \Omega_n$ with currently best known nonlinearity.

Algorithm 2 : BalConstruct(n)

1. If n is odd
   a) if $3 \leq n \leq 13$ construct f using Theorem 10(a).
   b) if $15 \leq n \leq 21$ return f to be the best function constructed by RandBal(n).
   c) if $n \geq 23$
      (i) Let $h_1 \in \Omega_{n-21}$ be bent and $g_1 \in \Omega_{21}$ be the best nonlinear function constructed by RandBal(n).
      Let $f_1 \in \Omega_n$ be such that $f_1 = h_1 \oplus g_1$.
      (ii) Let $h_2 =$ BalConstruct$(n-15)$ and $g_2 \in \Omega_{15}$ as in Proposition 4.
      Let $f_2 \in \Omega_n$ be such that $f_2 = h_2 \oplus g_2$.
      (iii) If $nl(f_1) \geq nl(f_2)$ return f_1 else return f_2.
2. If n is even
   Let $h =$ BalConstruct$(\frac{n}{2})$. Let f be the concatenation of h followed by $2^{\frac{n}{2}} - 1$ distinct nonconstant linear functions on $\frac{n}{2}$ variables. Return f.

End Algorithm.

The following points need to be noted for providing the maximum algebraic degree $n-1$.

1. For odd $n \leq 13$, Theorem 10(a) guarantees degree $(n-1)$.
2. For odd n, $15 \leq n \leq 21$, modification of algorithm RandBal() guarantees algebraic degree $(n-1)$ without dropping nonlinearity.
3. For odd $n \geq 23$, using Proposition 3, degree $(n-1)$ can be achieved sacrificing

nonlinearity by at most 2.
4. For even n, recursively ensure that algebraic degree of h (in Step 2 of BalConstruct()) is $\frac{n}{2} - 1$.

In this section we have shown how to heuristically modify the Patterson-Wiedemann functions to obtain balancedness while retaining nonlinearity higher than the bent concatenation bound. However, the question of mathematically constructing such functions remains open. Also settling the conjecture in [3] is an important unsolved question.

8 Propagation Characteristics, Strict Avalanche Criteria

In this section we provide important results on propagation characteristics and strict avalanche criteria. The following is a general construction of Boolean functions introduced in [6].

$$f(X_1, \ldots, X_s, Y_1, \ldots, Y_t) = [X_1, \ldots, X_s] Q [Y_1, \ldots, Y_t]^T \oplus g(X_1, \ldots, X_s), \quad \textbf{(*)}$$

where Q is an $s \times t$ binary matrix and $g(X_1, \ldots, X_s)$ is any function.
Under certain conditions on Q, the function f satisfies PC(l) of order k (see [6]). Moreover, according to the proof of [6, Theorem 16], $nl(f) = 2^t nl(g)$ and $deg(f) = deg(g)$. It is possible to significantly improve the results of [6] by using functions constructed by the methods of Section 7.

Theorem 14. *For odd s, it is possible to construct PC(l) of order k function f such that (a) $deg(f) = s - 1$ and $nl(f) \geq 2^{t+s-1} - 2^{t+\frac{s-1}{2}}$ for $3 \leq s \leq 13$, (b) $deg(f) = s$ and $nl(f) > 2^{t+s-1} - 2^{t+\frac{s-1}{2}}$ for $s \geq 15$.*

Proof : For $3 \leq s \leq 13$, s odd, we can consider $g \in \Omega_s$ as the function available from Theorem 10(a) with algebraic degree $s-1$ and nonlinearity $2^{s-1} - 2^{\frac{s-1}{2}}$. For $s \geq 15$, one can consider $g \in \Omega_s$ with nonlinearity $2^{s-1} - 2^{\frac{s-1}{2}} + 20 \times 2^{\frac{s-15}{2}} - 1$ and algebraic degree s. This can be obtained by considering a function on s variables with maximum known nonlinearity and then making $wt(g)$ odd by toggling one bit. This will provide the full algebraic degree and decrease the nonlinearity by at most 1 only. □

For odd s, the corresponding result in [6] is $deg(f) = \frac{s-1}{2}$ and $nl(f) \geq 2^{t+s-1} - 2^{t+\frac{s-1}{2}}$ which is clearly improved in Theorem 14.

Now we show how to obtain maximum algebraic degree in this construction at the cost of small fall in nonlinearity. For odd s between 3 and 13, $deg(g)$ can be made s by changing one bit of g. This decreases $nl(g)$ by one. The corresponding parameters of f are $deg(f) = s$ and $nl(f) \geq 2^{t+s-1} - 2^{t+\frac{s-1}{2}} - 2^t$. For even s, the result in [6] is $deg(f) = \frac{s}{2}$ and $nl(f) \geq 2^{t+s-1} - 2^{t+\frac{s}{2}-1}$. As before by changing one bit of g we can ensure $deg(f) = s$ and $nl(f) \geq 2^{t+s-1} - 2^{t+\frac{s}{2}-1} - 2^t$. Also in [13], we show that it is possible to construct PC(1) functions with nonlinearity strictly greater than $2^{n-1} - 2^{\frac{n-1}{2}}$ for all odd $n \geq 15$.

Next we turn to the study of SAC(k) combined with the properties of balancedness, degree and nonlinearity. *This is the first time that all these properties are being considered together with SAC(k).* The proofs for the next few results

are quite involved. Hence we present the constructions clearly and only sketch the proofs.

In [6], **(*)** has been used for the construction of SAC(k) function by setting $s = n-k-1, t = k+1$ and Q to be the $(n-k-1)\times(k+1)$ matrix whose all elements are 1. Under these conditions the function f takes the form $f(X_1, \ldots, X_n) = (X_1 \oplus \ldots \oplus X_{n-k-1})(X_{n-k} \oplus \ldots \oplus X_n) \oplus g(X_1, \ldots, X_{n-k-1})$. Moreover, it was shown that f is balanced if $\mid \{\overline{X} \mid g(\overline{X}) = 0, \overline{X}Q = 0\} \mid = \mid \{\overline{X} \mid g(\overline{X}) = 1, \overline{X}Q = 0\} \mid$ where $\overline{X} = (X_1, \ldots, X_{n-k-1})$. It is important to interpret this idea with respect to the truth table of g. This means that f is balanced if $\#\{\overline{X} \mid g(\overline{X}) = 0, wt(\overline{X}) = \text{ even }\} = \#\{\overline{X} \mid g(\overline{X}) = 1, wt(\overline{X}) = \text{ even }\}$. Thus, in the truth table we have to check for balancedness of g restricted to the rows where the weight of the input string is even. In half of such places g must be 0 and in the other half g must be 1. Motivated by this discussion we make the following definition of *brEven* (restricted balancedness with respect to inputs with even weight) and *brOdd* (restricted balancedness with respect to inputs with odd weight).

Definition 2. *Let $g \in \Omega_p$, $\overline{X} = (X_1, \ldots, X_p)$. Then g is called brEven (resp. brOdd) if $\#\{g(\overline{X}) = 0 \mid wt(\overline{X}) = even\} = \#\{g(\overline{X}) = 1 \mid wt(\overline{X}) = even\} = 2^{p-2}$ (resp. $\#\{g(\overline{X}) = 0 \mid wt(\overline{X}) = odd\} = \#\{g(\overline{X}) = 1 \mid wt(\overline{X}) = odd\} = 2^{p-2}$).*

The next result is important as it shows that certain types of bent functions can be brEven. This allows us to obtain balanced SAC(k) functions with very high nonlinearity which could not be obtained in [6].

Proposition 5. *For p even, it is possible to construct bent functions $g \in \Omega_p$ which are brEven.*

Proof : First note that g is brEven iff g^c is brEven. Let $q = 2^{\frac{p}{2}}$. For $0 \leq i \leq q-1$ let $l_i \in L_{\frac{p}{2}}$ be the linear function $a_{\frac{p}{2}} X_{\frac{p}{2}} \oplus \ldots \oplus a_1 X_1$, where $a_{\frac{p}{2}} \ldots a_1$ is the $\frac{p}{2}$-bit binary expansion of i. We provide construction of bent functions $g(X_1, \ldots, X_p)$ which are brEven. Let $\overline{X} = (X_1, \ldots, X_p)$.
Case 1: $\frac{p}{2} \equiv 1 \bmod 2$. Let $g = l_0 f_1 \ldots f_{q-2} l_{q-1}$, where
$f_1, \ldots, f_{q-2} \in \{l_1, \ldots, l_{q-2}, l_1^c, \ldots, l_{q-2}^c\}$ and for $i \neq j$, $f_i \neq f_j$ and $f_i \neq f_j^c$. It is well known that such a g is bent [11]. We show that g is brEven. First we have the following three results which we state without proofs.
(a) $\#\{l_0(X_1, \ldots, X_{\frac{p}{2}}) = 0 \mid wt(X_1, \ldots, X_{\frac{p}{2}}) = \text{ even }\} = 2^{\frac{p}{2}-1}$ and
$\#\{l_0(X_1, \ldots, X_{\frac{p}{2}}) = 1 \mid wt(X_1, \ldots, X_{\frac{p}{2}}) = \text{ even }\} = 0$.
(b) Since the f_i's are degenerate affine functions in $L_{\frac{p}{2}}$, it is possible to show that individually they are both brEven and brOdd.
(c) Using the fact that $q = \frac{p}{2}$ is odd and $l_{q-1} = X_1 \oplus \ldots \oplus X_{\frac{p}{2}}$, it is possible to show, $\#\{l_{q-1}(X_1, \ldots, X_{\frac{p}{2}}) = 0 \mid wt(X_1, \ldots, X_{\frac{p}{2}}) = \text{ even }\} = 0$ and
$\#\{l_{q-1}(X_1, \ldots, X_{\frac{p}{2}}) = 1 \mid wt(X_1, \ldots, X_{\frac{p}{2}}) = \text{ even }\} = 2^{\frac{p}{2}-1}$. Then using $wt(X_1, \ldots, X_p) = wt(X_1, \ldots, X_{\frac{p}{2}}) + wt(X_{\frac{p}{2}+1}, \ldots, X_p)$ and the fact that g is concatenation of $l_0, f_1, \ldots, f_{q-2}, l_{q-1}$ it is possible to show that g is brEven.
Case 2: For $\frac{p}{2} \equiv 0 \bmod 2$, the result is true for bent functions of the form $g = l_0^c f_1 \ldots f_{q-2} l_{q-1}$. □

In [6, Theorem 32] it has been stated that for $n-k-1 =$ even, there exists balanced SAC(k) functions such that $deg(f) = n-k-2$. The question whether such functions with algebraic degree $n-k-1$ exists has been left as an open question. The next result shows the existence of such functions which proves that the bound on algebraic degree provided in [10] is indeed tight for $k \leq \frac{n}{2}-1$.

Theorem 15. *Let $(n-k-1) \geq (k+1)$, i.e. $k \leq \frac{n}{2}-1$ and $n-k-1 =$ even. Then it is possible to construct balanced SAC(k) function $f \in \Omega_n$ such that $deg(f) = n-k-1$. Moreover $nl(f) = 2^{n-1} - 2^{\frac{n+k-1}{2}} - 2^{k+1}$.*

Proof : Use a bent function $g \in \Omega_{n-k-1}$ which is brEven. Out of the 2^{n-k-1} bit positions in g (in the output column of the truth table), there are 2^{n-k-2} positions where $wt(X_1, \ldots, X_{n-k-1}) =$ odd and the value of g at these positions can be toggled without disturbing the brEven property. Since g is bent, $wt(g) =$ even. Thus we choose a row j in the truth table where $wt(X_1, \ldots, X_{n-k-1}) =$ odd and construct g' by toggling the output bit. Thus $wt(g') = wt(g) \pm 1 =$ odd. Hence by Proposition 2, $deg(g') = n-k-1$. Thus, $f(X_1, \ldots, X_n) = (X_1 \oplus \ldots \oplus X_{n-k-1})(X_{n-k} \oplus \ldots \oplus X_n) \oplus g'(X_1, \ldots, X_{n-k-1})$ is balanced SAC(k) with algebraic degree $n-k-1$. Also $nl(g') = nl(g) - 1 = 2^{n-k-2} - 2^{\frac{n-k-1}{2}-1} - 1$. Now, it can be checked that $nl(f) = 2^{k+1} \times nl(g') = 2^{n-1} - 2^{\frac{n+k-1}{2}} - 2^{k+1}$. □

Next we provide similar results for odd $n-k-1$. The result is extremely important in the sense that the functions constructed in [9] can be modified to get restricted balancedness and hence can be used in the construction of highly nonlinear, balanced SAC(k) functions. *We know of no other place where the functions provided by Patterson and Wiedemann [9] have been used in the construction of SAC(k) functions.*

Proposition 6. *For p odd, it is possible to construct brEven $g \in \Omega_p$ with nonlinearity (i) $2^{p-1} - 2^{\frac{p-1}{2}}$ for $p \leq 13$ and (ii) $2^{p-1} - 2^{\frac{p-1}{2}} + 20 \times 2^{\frac{p-15}{2}}$ for $p \geq 15$.*

Proof : For $p \leq 13$, the idea of bent concatenation and similar techniques as in the proof of Proposition 5 can be used. For $p \geq 15$ the construction is different. We just give an outline of the proof. Let $f_1 \in \Omega_{15}$ be one of the functions constructed in [9]. Note that $nl(f_1) = 2^{14} - 2^7 + 20$. Now consider the 32768 functions of the form $f_1 \oplus l$, where $l \in L_{15}$. We have found functions among these which are brOdd (but none which are brEven). Let $f_2(X_1, \ldots, X_{15})$ be such a brOdd function. It is then possible to show that $f_3(X_1, \ldots, X_{15}) = f_2(X_1 \oplus \alpha_1, \ldots, X_{15} \oplus \alpha_{15})$ is brEven when $wt(\alpha_1, \ldots, \alpha_{15})$ is odd. Note that $nl(f_2) = nl(f_3) = nl(f_1)$. Let $g(Y_1, \ldots, Y_{2k})$ be a bent function on $2k$ variables. Define $F \in \Omega_{15+2k}$ as follows. $F = (Y_1 \oplus \ldots \oplus Y_{2k})(g \oplus f_2) \oplus (1 \oplus Y_1 \oplus \ldots \oplus Y_{2k})(g \oplus f_3)$. It can be proved that F is brEven and $nl(F) = 2^{14+2k} - 2^{7+k} + 20 \times 2^k$. □

Theorem 16. *Let $(n-k-1) \geq (k+1)$, i.e. $k \leq \frac{n}{2}-1$ and $n-k-1 =$ odd. Then it is possible to construct balanced SAC(k) function $f \in \Omega_n$ such that $deg(f) = n-k-1$. Moreover, for $3 \leq n-k-1 \leq 13$, $nl(f) = 2^{n-1} - 2^{\frac{n+k}{2}} - 2^{k+1}$ and for $n-k-1 \geq 15$, $nl(f) = 2^{n-1} - 2^{\frac{n+k}{2}} + 20 \times 2^{\frac{n+k-14}{2}} - 2^{k+1}$.*

This shows that it is possible to construct highly nonlinear balanced functions satisfying SAC(k) with maximum possible algebraic degree $n-k-1$. Functions with all these criteria at the same time has not been considered earlier.

Now we present an interesting result combining resiliency and propagation characteristics. In [15, Theorem 15], propagation criterion of m-resilient functions has been studied. Those functions satisfy propagation criteria with a specific set of vectors. However, they do not satisfy even PC(1) as propagation criteria is not satisfied for some vectors of weight 1. For n even, we present a construction to provide resilient functions in Ω_n which satisfy PC($\frac{n}{2}-1$).

Theorem 17. *It is possible to construct* 1*-resilient functions in* Ω_n*,* n *even, with nonlinearity* $2^{n-1}-2^{\frac{n}{2}}$ *and algebraic degree* $\frac{n}{2}-1$ *which satisfy PC*($\frac{n}{2}-1$).

Proof : Let $f \in \Omega_{n-2}$ be a bent function, n even. Then it can be checked that $F(X_1,\ldots,X_{n-1}) = (1\oplus X_{n-1})f(X_1,\ldots,X_{n-2})\oplus X_{n-1}(1\oplus f(X_1\oplus\alpha_1,\ldots,X_{n-2}\oplus\alpha_{n-2}))$ is balanced and satisfies propagation criterion with respect to all nonzero vectors except $(\alpha_1,\ldots,\alpha_{n-2},1)$. Also $nl(F) = 2^{n-2}-2^{\frac{n-2}{2}}$.

Let $G(X_1,\ldots,X_n) = (1\oplus X_n)F(X_1,\ldots,X_{n-1})\oplus X_n(F(X_1\oplus\beta_1,\ldots,X_{n-1}\oplus\beta_{n-1}))$. Then it can be checked that G is balanced and satisfies propagation criterion with respect to all nonzero vectors except $\overline{\alpha} = (\alpha_1,\ldots,\alpha_{n-2},\alpha_{n-1}=1,\alpha_n=0)$, $\overline{\beta} = (\beta_1,\ldots,\beta_{n-1},\beta_n=1)$ and $\overline{\alpha}\oplus\overline{\beta}$. Also G is balanced and $nl(G) = 2^{n-1}-2^{\frac{n}{2}}$.

Take $(\alpha_1,\alpha_2,\ldots,\alpha_{n-2})$ in the construction of F in Ω_{n-1} from $f\in\Omega_{n-2}$ so that $wt(\alpha_1,\alpha_2,\ldots,\alpha_{n-2}) = \frac{n}{2}-1$.
Also $G(X_1,\ldots,X_n) = (1\oplus X_n)F(X_1,\ldots,X_{n-1})\oplus X_n(F(X_1\oplus 1,\ldots,X_{n-1}\oplus 1)$ is correlation immune [1]. Since F is balanced, G is also balanced which proves that G is 1-resilient. Now consider $\overline{\alpha} = (\alpha_1,\ldots,\alpha_{n-2},\alpha_{n-1}=1,\alpha_n=0)$, $\overline{\beta} = (\beta_1=1,\ldots,\beta_{n+1}=1,\beta_{n+2}=1)$. Since $wt(\overline{\alpha}) = \frac{n}{2}-1+1$ and $wt(\overline{\beta}) = n$ we get, $wt(\overline{\alpha}\oplus\overline{\beta}) = \frac{n}{2}$. Note that G satisfies propagation criterion with respect to all the nonzero vectors except $\overline{\alpha},\overline{\beta},\overline{\alpha}\oplus\overline{\beta}$ and hence G satisfies PC($\frac{n}{2}-1$).

Since $f\in\Omega_{n-2}$ is bent, it is possible to construct f with algebraic degree $\frac{n}{2}-1$. It can be checked that $deg(G) = deg(f)$. □

9 Conclusion

In this paper we have considered cryptographically important properties of Boolean functions such as balancedness, nonlinearity, algebraic degree, correlation immunity, propagation characteristics and strict avalanche criteria. The construction methods we propose here are new and they provide functions which were not known earlier.

References

1. P. Camion, C. Carlet, P. Charpin, and N. Sendrier. On correlation immune functions. In *Advances in Cryptology - CRYPTO'91*, pages 86–100. Springer-Verlag, 1992.

2. S. Chee, S. Lee, D. Lee, and S. H. Sung. On the correlation immune functions and their nonlinearity. In *Advances in Cryptology, Asiacrypt 96*, number 1163 in Lecture Notes in Computer Science, pages 232–243. Springer-Verlag, 1996.
3. H. Dobbertin. Construction of bent functions and balanced Boolean functions with high nonlinearity. In *Fast Software Encryption*, number 1008 in Lecture Notes in Computer Science, pages 61–74. Springer-Verlag, 1994.
4. E. Filiol and C. Fontaine. Highly nonlinear balanced Boolean functions with a good correlation-immunity. In *Advances in Cryptology - EUROCRYPT'98*. Springer-Verlag, 1998.
5. X. Guo-Zhen and J. Massey. A spectral characterization of correlation immune combining functions. *IEEE Transactions on Information Theory*, 34(3):569–571, May 1988.
6. K. Kurosawa and T. Satoh. Design of SAC/PC(l) of order k Boolean functions and three other cryptographic criteria. In *Advances in Cryptology - EUROCRYPT'97*, Lecture Notes in Computer Science, pages 434–449. Springer-Verlag, 1997.
7. S. Maitra and P. Sarkar. Highly nonlinear resilient functions optimizing Siegenthaler's inequality. In *Advances in Cryptology - CRYPTO'99*, number 1666 in Lecture Notes in Computer Science, pages 198–215. Springer Verlag, August 1999.
8. E. Pasalic and T. Johansson. Further results on the relation between nonlinearity and resiliency of Boolean functions. In *IMA Conference on Cryptography and Coding*, number 1746 in Lecture Notes in Computer Science, pages 35–45. Springer-Verlag, 1999.
9. N. J. Patterson and D. H. Wiedemann. Correction to - the covering radius of the $(2^{15}, 16)$ Reed-Muller code is at least 16276. *IEEE Transactions on Information Theory*, IT-36(2):443, 1990.
10. B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle. Propagation characteristics of Boolean functions. In *Advances in Cryptology - EUROCRYPT'90*, Lecture Notes in Computer Science, pages 161–173. Springer-Verlag, 1991.
11. O. S. Rothaus. On bent functions. *Journal of Combinatorial Theory, Series A*, 20:300–305, 1976.
12. P. Sarkar and S. Maitra. Construction of nonlinear resilient Boolean functions. *Indian Statistical Institute, Technical Report No. ASD/99/30*, November 1999.
13. P. Sarkar and S. Maitra. Highly nonlinear balanced Boolean functions with important cryptographic properties. *Indian Statistical Institute, Technical Report No. ASD/99/31*, November 1999.
14. J. Seberry, X. M. Zhang, and Y. Zheng. Nonlinearly balanced Boolean functions and their propagation characteristics. In *Advances in Cryptology - CRYPTO'93*, pages 49–60. Springer-Verlag, 1994.
15. J. Seberry, X. M. Zhang, and Y. Zheng. On constructions and nonlinearity of correlation immune Boolean functions. In *Advances in Cryptology - EUROCRYPT'93*, pages 181–199. Springer-Verlag, 1994.
16. T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, IT-30(5):776–780, September 1984.
17. T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, C-34(1):81–85, January 1985.
18. Y. V. Tarannikov. On a method for the constructing of cryptographically strong Boolean functions. *Moscow State University, French-Russian Institute of Applied Mathematics and Informatics, Preprint No. 6*, October 1999.

Propagation Characteristics and Correlation-Immunity of Highly Nonlinear Boolean Functions

Anne Canteaut[1], Claude Carlet[2], Pascale Charpin[1], and Caroline Fontaine[3]

[1] INRIA projet CODES
B.P. 105
78153 Le Chesnay Cedex - France.
{Anne.Canteaut,Pascale.Charpin}@inria.fr
[2] GREYC, Université de Caen
14032 Caen Cedex - France.
Claude.Carlet@info.unicaen.fr
[3] LIFL, Université des Sciences et Technologies de Lille
59655 Villeneuve d'Ascq Cedex - France.
Caroline.Fontaine@lifl.fr

Abstract. We investigate the link between the nonlinearity of a Boolean function and its propagation characteristics. We prove that highly nonlinear functions usually have good propagation properties regarding different criteria. Conversely, any Boolean function satisfying the propagation criterion with respect to a linear subspace of codimension 1 or 2 has a high nonlinearity. We also point out that most highly nonlinear functions with a three-valued Walsh spectrum can be transformed into 1-resilient functions.

1 Introduction

The design of conventional cryptographic systems relies on two fundamental principles introduced by Shannon [25]: confusion and diffusion. Confusion aims at concealing any algebraic structure in the system. Diffusion consists in spreading out the influence of a minor modification of the input data over all outputs. Most conventional primitives are concerned with these essential principles: secret-key ciphers (block ciphers and stream ciphers) as well as hash functions. Confusion and diffusion can be quantified by some properties of the Boolean functions describing the system. Confusion corresponds to the nonlinearity of the involved functions, *i.e.*, to their Hamming distances to the set of affine functions. Diffusion is related to the propagation characteristics of the considered Boolean function f: these properties describe the behaviors of the derivatives $x \mapsto f(x+a)+f(x)$. The relevant cryptographic quantities are the biases of the output probability distributions of the derivatives relatively to the uniform distribution; they are measured by the auto-correlation coefficients of the function. Diffusion is therefore estimated by complementary indicators: propagation

criterion, distance to the set of all Boolean functions with a linear structure and sum-of-squares indicator. All these quantities will be here considered in a unified approach.

A major link between diffusion and confusion criteria was pointed out by Meier and Staffelbach [18]. They proved that maximal nonlinearity and perfect propagation characteristics are equivalent requirements for Boolean functions with an even number of variables. Unfortunately those functions which achieve perfect diffusion and perfect confusion (called bent functions) are not balanced; that means that they do not have a uniform output distribution. The construction of balanced Boolean functions having a high nonlinearity and good propagation characteristics then remains an open problem although such functions are essential components of cryptographic primitives.

In this paper we further investigate the link between diffusion and confusion criteria for Boolean functions. We show that highly nonlinear functions usually coincide with the functions having remarkable propagation characteristics. In this context, we point out the major role played by the highly nonlinear functions whose Walsh spectrum takes three values. We exhibit general constructions of such functions and we prove that they can easily be transformed into balanced first-order correlation-immune functions. They are therefore well-suited combining functions for pseudo-random generators since they ensure a high resistance to fast correlation attacks.

2 Cryptographic criteria for Boolean functions

A Boolean function with n variables is a function from the set of n-bit vectors, $\mathbf{F}_2^n$, into $\mathbf{F}_2$. Such a function f can be expressed as a unique polynomial in $x_1, \ldots, x_n$ called its *algebraic normal form* (see e.g. [14]). Some cryptographic applications require that this polynomial has a high degree. For instance, when f is used as a combining function in a pseudo-random generator, its degree conditions the linear complexity of the produced running-key. The following notation will be intensively used in the paper. The usual dot product between two vectors x and y is denoted by $x \cdot y$. For any $\alpha \in \mathbf{F}_2^n$, ϕ_α is the linear function with n variables defined by $\phi_\alpha(x_1, \ldots, x_n) = \alpha \cdot x = \sum_{i=1}^{n} \alpha_i x_i$. The *Walsh transform* of a Boolean function f refers to the Fourier transform of the corresponding sign function $x \mapsto (-1)^{f(x)}$. In this context we denote by $\mathcal{F}(f)$ the value in 0 of the Walsh transform of f:

$$\mathcal{F}(f) = \sum_{x \in \mathbf{F}_2^n} (-1)^{f(x)} = 2^n - 2wt(f)$$

where $wt(f)$ is the Hamming weight of f, *i.e.*, the number of $x \in \mathbf{F}_2^n$ such that $f(x) = 1$.

The *Walsh spectrum* of a Boolean function f with n variables therefore consists of all values $\{\mathcal{F}(f + \phi_\alpha), \alpha \in \mathbf{F}_2^n\}$. Since linear attacks on blocks ciphers and correlation attacks on stream ciphers equally search for a linear or an affine approximation of the involved function, the signs of the Walsh coefficients have

no cryptographic relevance. We then often consider the set $\{\mathcal{F}(f+\phi_\alpha+\varepsilon), \alpha \in \mathbf{F}_2^n, \varepsilon \in \mathbf{F}_2\}$. The values of this spectrum, called the *extended Walsh spectrum*, are symmetric with respect to 0 since $\mathcal{F}(f+\phi_\alpha+1) = -\mathcal{F}(f+\phi_\alpha)$.

We now recall the main cryptographic criteria for Boolean functions and we express all of them in terms of Walsh spectrum. A first obvious requirement in most applications is that the output of the used Boolean function be uniformly distributed. This corresponds to *balancedness*:

Definition 1. *A Boolean function f is balanced if $\mathcal{F}(f) = 0$.*

A second usual criterion is that f should be far from all affine functions (regarding Hamming distance). In stream ciphers applications, when f is used in a pseudo-random generator as a combining function or as a filtering function, the existence of a "good" approximation of f by an affine function makes fast correlation attacks feasible [17, 13, 12]. Similarly, if f is used in a block cipher as an S-box component, this would lead to successful linear attacks [15].

Definition 2. *The* nonlinearity *of a Boolean function f with n variables is its Hamming distance to the set of affine functions. It can be expressed as*

$$\mathcal{NL}(f) = 2^{n-1} - \frac{1}{2}\mathcal{L}(f) \text{ where } \mathcal{L}(f) = \max_{\alpha \in \mathbf{F}_2^n} |\mathcal{F}(f+\phi_\alpha)| \ .$$

Any Boolean function f with n variables satisfies $\mathcal{L}(f) \geq 2^{n/2}$; the functions for which equality holds are called *bent functions* [23]. This lower bound can only be achieved for even values of n. When n is odd, the lowest achievable value of $\mathcal{L}(f)$ is unknown in the general case: there always exist some functions with $\mathcal{L}(f) = 2^{(n+1)/2}$ and this value corresponds to the minimum possible nonlinearity for any $n \leq 7$. On the other hand some functions with $\mathcal{L}(f) = \frac{27}{32}2^{(n+1)/2}$ are known for any odd $n \geq 15$ [20, 21]. From now on, we will focus on highly nonlinear Boolean functions in the following sense:

Definition 3. *Let f be a Boolean function with n variables. Then f is said to be* almost optimal *if $\mathcal{L}(f) \leq 2^{(n+1)/2}$ when n is odd, and $\mathcal{L}(f) \leq 2^{(n+2)/2}$ when n is even.*

Besides its maximum value, the whole Walsh spectrum of a Boolean function has a great cryptographic significance. When f is used in a combining pseudo-random generator, the distribution probability of its output should be unaltered when any t of its inputs are fixed [27]. This property, called *t-th order correlation-immunity* [26], is characterized by the set of zero values in the Walsh spectrum [29]:

Definition 4. *Let f be a Boolean function with n variables.*

- *f is* correlation-immune with respect to a subset *E of $\mathbf{F}_2^n$ if $\mathcal{F}(f+\phi_\alpha) = 0$ for all $\alpha \in E$.*
- *f is* t-th order correlation-immune (t-CI) *if it is correlation-immune with respect to $\{x \in \mathbf{F}_2^n, 1 \leq wt(x) \leq t\}$, where $wt(x)$ denotes the Hamming weight of the n-bit vector x,* i.e.*, the number of its nonzero components.*

Balanced t-th order correlation-immune functions are called t-resilient *functions.*

These criteria may not be compatible in general: there are necessary tradeoffs between the degree, the nonlinearity and the correlation-immunity order of a function.

Some other criteria consider the probability distribution of the output difference of the Boolean function for a fixed input difference. They then focus on the properties of the functions $D_a f : x \mapsto f(x+a) + f(x)$ for $a \in \mathbf{F}_2^n$. The function $D_a f$ is called the *derivative of f with respect to direction a.* The *auto-correlation function of f* refers to the function $\alpha \mapsto \mathcal{F}(D_\alpha f)$. The auto-correlation coefficient $\mathcal{F}(D_\alpha f)$ then measures the statistical bias of the output distribution of $D_\alpha f$ relatively to the uniform distribution. The propagation characteristics of a Boolean function can then be estimated by several indicators. Some applications require that the output difference of a function be uniformly distributed for low-weight input differences. This property, referred as *propagation criterion* [22], is notably important when the function is used in a hash function or in a block cipher.

Definition 5. *Let f be a Boolean function with n variables.*

- *f satisfies the* propagation criterion with respect to a subset E *of* $\mathbf{F}_2^n$ *if* $\mathcal{F}(D_\alpha f) = 0$ *for all* $\alpha \in E$.
- *f satisfies the* propagation criterion of degree k (PC(k)) *if it satisfies the propagation criterion with respect to* $\{x \in \mathbf{F}_2^n, 1 \leq wt(x) \leq k\}$.

The strict avalanche criterion (SAC) [28] actually corresponds to the propagation criterion of degree 1. It is also recommended that the output distribution of all derivatives be close to the uniform distribution: the existence of a derivative whose output takes a constant value with a high probability leads to differential attacks [3, 2]. This means that $|\mathcal{F}(D_\alpha f)|$ should be small for all nonzero $\alpha \in \mathbf{F}_2^n$. Recall that the *linear space* of f is the subspace of those α such that $D_\alpha f$ is a constant function. Such $\alpha \neq 0$ is said to be a *linear structure* for f. The maximum value $|\mathcal{F}(D_\alpha f)|$ over all nonzero α, called the *absolute indicator* [30], then quantifies the distance of f to the set of all Boolean functions with a linear structure [18]. The only functions whose absolute indicator equals 0 are the bent functions.

The output distributions of the derivatives can also be studied in average through the second moment of the auto-correlation coefficients, called the *sum-of-squares indicator* [30]:

Definition 6. *The* sum-of-squares indicator *of a Boolean function f with n variables, denoted by $\mathcal{V}(f)$, is defined by*

$$\mathcal{V}(f) = \sum_{\alpha \in \mathbf{F}_2^n} \mathcal{F}^2(D_\alpha f) \ .$$

The above presented criteria are invariant under certain transformations.

Proposition 1. *The degree, the extended Walsh spectrum (and the nonlinearity), the absolute indicator and the sum-of-squares indicator are invariant under addition of an affine function.*

The invariance of the propagation characteristics is derived from $\mathcal{F}(D_\alpha(f + \phi_\beta)) = (-1)^{\alpha\cdot\beta}\mathcal{F}(D_\alpha f)$ for any α and β in $\mathbf{F}_2^n$. Most notably, this proposition implies that if there exists $\alpha \in \mathbf{F}_2^n$ such that $\mathcal{F}(f + \phi_\alpha) = 0$, then $f + \phi_\alpha$ is a balanced function having the same degree, extended Walsh spectrum, absolute indicator and sum-of-squares indicator as f.

Proposition 2. *The weight, the degree, the Walsh spectrum (and the nonlinearity), the absolute indicator and the sum-of-squares indicator are invariant under right composition by a linear permutation of* $\mathbf{F}_2^n$.

Both of these types of transformations change neither the size nor the rank of the sets $E_{CI}(f) = \{\alpha \in \mathbf{F}_2^n,\ \mathcal{F}(f+\phi_\alpha) = 0\}$ and $E_{PC}(f) = \{\alpha \in \mathbf{F}_2^n,\ \mathcal{F}(D_\alpha f) = 0\}$. The first-order correlation immunity and the propagation criterion of degree 1 can therefore be studied up to the previous equivalences:

Proposition 3. *Let f be a Boolean function with n variables. If $E_{CI}(f)$ (resp., $E_{PC}(f)$) has rank n, then there exists a linear permutation π of $\mathbf{F}_2^n$ such that the Boolean function $f \circ \pi$ is first-order correlation-immune (resp., satisfies the propagation criterion of degree 1).*

The rest of the paper is organized as follows. We observe in Section 3 that the nonlinearity of a Boolean function provides an upper bound on its sum-of-squares indicator. Moreover, we completely characterize the functions achieving this bound: their extended Walsh spectra take at most 3 values. In Section 4 we derive a lower bound on the number of zero auto-correlation coefficients of a function from its nonlinearity. Section 5 is devoted to the nonlinearity of Boolean functions with a linear structure. We essentially show that these functions are not almost optimal when the dimensions of their linear spaces exceed 1 for odd n, and 2 for even n. Conversely, Section 6 focuses on the functions which satisfy the propagation criterion with respect to a linear subspace of codimension 1 or 2. We prove that these functions are almost optimal and that they have a three-valued extended Walsh spectrum when n is odd. For even n we obtain new characterizations of bent functions. In the last section we study the correlation-immunity order of Boolean functions with a three-valued Walsh spectrum. Such functions are 1-resilient (up to a linear permutation) unless n is odd and they satisfy PC$(n-1)$. We deduce that for any odd n and any degree $d \leq (n+1)/2$, there exist 1-resilient functions of degree d, with n variables, and with nonlinearity $2^{n-1} - 2^{(n-1)/2}$.

3 Relation between the sum-of-squares indicator and the Walsh spectrum

The auto-correlation coefficients of a Boolean function are related to its Walsh spectrum through the following formulas. Proofs of these results can notably be found in [5] and [30].

Lemma 1. *Let f be a Boolean function with n variables. For any $\alpha \in \mathbf{F}_2^n$,*

$$\mathcal{F}^2(f+\phi_\alpha) = \sum_{\beta \in \mathbf{F}_2^n} (-1)^{\alpha \cdot \beta} \mathcal{F}(D_\beta f) \ .$$

Lemma 2. *Let f be a Boolean function with n variables. Then*

$$\sum_{\alpha \in \mathbf{F}_2^n} \mathcal{F}^4(f+\phi_\alpha) = 2^n \mathcal{V}(f) \ .$$

We now point out that the nonlinearity of a function obviously provides an upper bound on its sum-of-squares indicator, $\mathcal{V}(f)$. Moreover, some further information on the Walsh spectrum of a function can be derived from the value of $\mathcal{V}(f)$. The following result was proved independently in [32, Theorem 5]. We give here a much simpler proof.

Theorem 1. *Let f be a Boolean function with n variables and let $\mathcal{L}(f) = \max_{\alpha \in \mathbf{F}_2^n} |\mathcal{F}(f+\phi_\alpha)|$. Then we have*

$$\mathcal{V}(f) \leq 2^n \mathcal{L}(f)^2$$

with equality if and only if the extended Walsh spectrum of f takes at most three values, 0, $\mathcal{L}(f)$ and $-\mathcal{L}(f)$.

Proof: Let us consider the following quantity

$$\mathcal{I}(f) = \sum_{\alpha \in \mathbf{F}_2^n} \mathcal{F}^2(f+\phi_\alpha) \left[\mathcal{F}^2(f+\phi_\alpha) - \mathcal{L}(f)^2 \right] \ .$$

By Parseval's relation we have $\sum_{\alpha \in \mathbf{F}_2^n} \mathcal{F}^2(f+\phi_\alpha) = 2^{2n}$. It then follows from Lemma 2 that $\mathcal{I}(f) = 2^n(\mathcal{V}(f) - 2^n \mathcal{L}(f)^2)$. By definition $\mathcal{I}(f)$ consists of a sum of terms T_α, $\alpha \in \mathbf{F}_2^n$, which satisfy $T_\alpha \leq 0$ if and only if $|\mathcal{F}(f+\phi_\alpha)| \leq \mathcal{L}(f)$. Since $|\mathcal{F}(f+\phi_\alpha)| \leq \mathcal{L}(f)$ for any α, all terms T_α in $\mathcal{I}(f)$ are non positive, and thus $\mathcal{I}(f) \leq 0$. The equality holds if and only if all terms T_α in $\mathcal{I}(f)$ vanish. This only occurs if $|\mathcal{F}(f+\phi_\alpha)| \in \{0, \mathcal{L}(f)\}$ for all α. □

Following Definition 3, the sum-of-squares indicator of an almost optimal function f with n variables then satisfies $\mathcal{V}(f) \leq 2^{2n+1}$ if n is odd, and $\mathcal{V}(f) \leq 2^{2n+2}$ if n is even.

Example 1. We consider the following function of degree 5 with 7 variables:

$$f(x_1, \ldots, x_7) = x_1x_2x_3x_4x_5 + x_1x_3x_7 + x_1x_2 + x_3x_4 + x_5x_6 \ .$$

This function is almost optimal and its extended Walsh spectrum takes exactly 5 values, $0, \pm 8, \pm 16$. Let A_i denote the number of α such that $|\mathcal{F}(f+\phi_\alpha)| = i$. We have $A_0 = 40$, $A_8 = 32$ and $A_{16} = 56$. It follows that $\mathcal{V}(f) = 29696 < 2^{15}$.

This function f can be added to a bent function with $(n-7)$ variables for any odd $n \geq 7$. This provides an almost optimal function g with n variables whose extended Walsh spectrum takes the following 5 values: $0, \pm 2^{(n-1)/2}, \pm 2^{(n+1)/2}$. Moreover, we have $A_0 = 5 \cdot 2^{n-4}$, $A_{2^{(n-1)/2}} = 2^{n-2}$ and $A_{2^{(n+1)/2}} = 7 \cdot 2^{n-4}$; thus $\mathcal{V}(g) = 2^{2n+1} - 3 \cdot 2^{2n-4}$.

The functions whose extended Walsh spectra take at most three values are very specific since their extended Walsh spectrum is completely determined by their nonlinearity. In this case the values of the Walsh transform belong to $0, \pm\mathcal{L}(f)$.

Theorem 2. *Let f be a Boolean function with n variables. Assume that the extended Walsh spectrum of f takes at most three values, 0 and $\pm\mathcal{L}(f)$. Then $\mathcal{L}(f) = 2^i$ with $i \geq n/2$ and*

$$\#\{\alpha \in \mathbf{F}_2^n,\ |\mathcal{F}(f+\phi_\alpha)| = \mathcal{L}(f)\} = \frac{2^{2n}}{\mathcal{L}(f)^2} = 2^{2n-2i}\ ;$$

$$\#\{\alpha \in \mathbf{F}_2^n,\ |\mathcal{F}(f+\phi_\alpha)| = 0\} = \frac{2^n(\mathcal{L}(f)^2 - 2^n)}{\mathcal{L}(f)^2} = 2^n - 2^{2n-2i}\ .$$

Moreover, the degree of f is less than or equal to $n - i + 1$.

Proof: Since $\mathcal{F}^2(f+\phi_\alpha)$ lies in $\{0, \mathcal{L}(f)^2\}$ for all $\alpha \in \mathbf{F}_2^n$, we have from Parseval's relation

$$\sum_{\alpha \in \mathbf{F}_2^n} \mathcal{F}^2(f+\phi_\alpha) = \mathcal{L}(f)^2 A_{\mathcal{L}(f)} = 2^{2n}$$

where $A_{\mathcal{L}(f)} = \#\{\alpha \in \mathbf{F}_2^n,\ |\mathcal{F}(f+\phi_\alpha)| = \mathcal{L}(f)\}$. It follows that $\mathcal{L}(f) = 2^i$. Since $A_{\mathcal{L}(f)} \leq 2^n$, we deduce that $i \geq n/2$. The upper-bound on the degree of f comes from the divisibility of the Walsh coefficients [6, Lemma 3]. □

Note that any Boolean function of degree 2 satisfies the hypotheses of the previous theorem [14, p. 441]. Theorem 2 now implies that the only almost optimal functions having a three-valued extended Walsh spectrum satisfy $\mathcal{L}(f) = 2^{(n+1)/2}$ when n is odd and $\mathcal{L}(f) = 2^{(n+2)/2}$ when n is even (bent functions have a two-valued extended Walsh spectrum).

4 Propagation criterion on highly nonlinear functions

We have pointed out that the nonlinearity of a Boolean function provides an upper bound on its sum-of-squares indicator, *i.e.*, on the second moment of the auto-correlation coefficients. We now show that it also gives a lower bound on the number of zero auto-correlation coefficients.

Proposition 4. *Let f be a Boolean function of degree d with n variables and let $E_{PC}(f) = \{\alpha \in \mathbf{F}_2^n,\ \mathcal{F}(D_\alpha f) = 0\}$. Then*

$$|E_{PC}(f)| \geq 2^n - 1 - 2^{n-4-2\lfloor \frac{n-2}{d-1} \rfloor} \left(\mathcal{L}(f)^2 - 2^n\right)\ .$$

Proof: Since any derivative $D_\alpha f$ of f is a function of degree $(d-1)$ with a linear structure, $\mathcal{F}(D_\alpha f)$ is divisible by $2^{\lfloor \frac{n-2}{d-1} \rfloor + 2}$ [16]. We then deduce

$$\begin{aligned}\mathcal{V}(f) &= \sum_{\alpha \notin E_{PC}(f)} \mathcal{F}^2(D_\alpha f) = 2^{2n} + \sum_{\alpha \notin E_{PC}(f), \alpha \neq 0} \mathcal{F}^2(D_\alpha f) \\ &\geq 2^{2n} + (2^n - 1 - |E_{PC}(f)|) 2^{2\lfloor \frac{n-2}{d-1} \rfloor + 4}\end{aligned}$$

We know from Theorem 1 that $\mathcal{V}(f) \leq 2^n \mathcal{L}(f)$. We therefore deduce the expected result. □

This bound is essentially relevant for functions having a high nonlinearity and a low degree. For instance we deduce that almost optimal functions of degree 3 satisfy $|E_{PC}(f)| \geq 2^{n-2} - 1$ when n is even and $|E_{PC}(f)| \geq 2^{n-1} - 1$ when n is odd.

Corollary 1. *Let n be an odd integer. Let f be an almost optimal function of degree 3 with n variables. Then there exists a permutation π of $\mathbf{F}_2^n$ such that $f \circ \pi$ satisfies PC(1) unless there exists an affine subspace $\overline{H}$ of $\mathbf{F}_2^n$ of codimension 1 such that $\mathcal{F}^2(D_\alpha f) = 2^{n+1}$ for any $\alpha \in \overline{H}$.*

Proof: It follows from Proposition 3 that f can be transformed into a function satisfying PC(1) if $E_{PC}(f)$ has rank n. Since the previous theorem implies that $|E_{PC}(f) \cup \{0\}| \geq 2^{n-1}$, $E_{PC}(f)$ has full rank except if $E_{PC}(f) \cup \{0\}$ is an hyperplane of $\mathbf{F}_2^n$, *i.e.*, a linear subspace of codimension 1. In this case, the lower bound on the size of $E_{PC}(f)$ is achieved. It is clear from the proof of the previous theorem that this occurs if and only if $\mathcal{V}(f) = 2^{2n+1}$ and $\mathcal{F}^2(D_\alpha f) = 2^{n+1}$ for any nonzero $\mathbf{F}_2^n \setminus E_{PC}(f)$. □

This corollary therefore provides a fast algorithm for obtaining almost optimal functions of degree 3 which satisfy PC(1) when the number of variables is odd.

5 Walsh spectrum of Boolean functions with a linear structure

Theorem 1 also enables us to characterize almost optimal functions which have a linear structure.

Theorem 3. *Let f be a Boolean function with n variables. Assume that f has a linear space V of dimension $k \geq 1$. Then*

$$\mathcal{L}(f) \geq 2^{\frac{n+k}{2}}$$

with equality if and only if f satisfies the propagation criterion with respect to $\mathbf{F}_2^n \setminus V$.

In this case, k and n have the same parity and f has a three-valued extended Walsh spectrum.

Proof: If f has a linear space of dimension k, the sum-of-squares indicator satisfies

$$\mathcal{V}(f) = 2^{2n+k} + \sum_{\alpha \notin V} \mathcal{F}^2(D_\alpha f) \geq 2^{2n+k} \ .$$

Thus $\mathcal{L}(f) \geq 2^{(n+k)/2}$ according to Theorem 1 with equality if and only f has a three-valued extended Walsh spectrum and $\mathcal{L}(f) = 2^{(n+k)/2}$. This implies that n and k have the same parity. □

Corollary 2. *Let n be an odd integer and let f be a Boolean function with n variables. The following assertions are equivalent:*

(i) *f is almost optimal and it has a linear structure.*
(ii) *there exists a linear permutation π of $\mathbf{F}_2^n$ such that $f \circ \pi$ satisfies PC($n-2$).*
(iii) *there exists a linear permutation π of $\mathbf{F}_2^n$ such that $f \circ \pi$ satisfies PC($n-1$).*

Proof: Carlet [7, Prop. 1] proved that the second and third assertions are equivalent. Moreover, any function satisfying PC($n-1$) has a linear structure e and all its derivatives with respect to direction $\alpha \notin \{0, e\}$ are balanced. The previous theorem then proves the equivalence with the first assertion. □

The extended Walsh spectrum of an almost optimal function which has a linear structure is then completely determined unless the number of variables is even and the linear space has dimension 1. We now give an example of this situation:

Example 2. Let f_1 and f_2 be the following almost optimal functions with 8 variables:

$$f_1(x_1, \ldots, x_8) = x_1x_2x_3x_4x_5 + x_1x_3x_7 + x_1x_2 + x_3x_4 + x_5x_6 + x_8 \ ,$$
$$f_2(x_1, \ldots, x_8) = x_1x_3x_4x_6 + x_4x_6x_7 + x_1x_2 + x_3x_4 + x_5x_6 + x_8 \ .$$

Both of these functions have a linear space of dimension 1. From Example 1 we know that f_1 has a 5-valued extended Walsh spectrum and $\mathcal{V}(f_1) = 2^{2n+2} - 3 \cdot 2^{2n-3}$. On the other hand f_2 has a 3-valued extended Walsh spectrum and satisfies $\mathcal{V}(f_2) = 2^{2n-2}$.

6 Functions satisfying the propagation criterion with respect to a linear subspace

The previous 3 sections have shown that almost optimal functions generally have good propagation characteristics regarding all indicators. We now conversely focus on the Walsh spectra of the Boolean functions f which have the following remarkable propagation property: f satisfies the propagation criterion with respect to any nonzero element of a linear subspace of $\mathbf{F}_2^n$ of codimension 1 or 2.

Proposition 5. *Let V be a linear subspace of $\mathbf{F}_2^n$ of dimension k. Let $V^\perp$ denote its dual,* i.e., *$V^\perp = \{x \in \mathbf{F}_2^n, x \cdot y = 0$ for all $y \in V\}$. For any Boolean function f with n variables, we have*

$$\sum_{\alpha \in V} \mathcal{F}^2(f + \phi_\alpha) = 2^k \sum_{\beta \in V^\perp} \mathcal{F}(D_\beta f) \ .$$

Proof: We deduce from Lemma 1:

$$\begin{aligned} \sum_{\alpha \in V} \mathcal{F}^2(f + \phi_\alpha) &= \sum_{\alpha \in V} \sum_{\beta \in \mathbf{F}_2^n} (-1)^{\alpha \cdot \beta} \mathcal{F}(D_\beta f) \\ &= \sum_{\beta \in \mathbf{F}_2^n} \mathcal{F}(D_\beta f) \left(\sum_{\alpha \in V} (-1)^{\alpha \cdot \beta} \right) = 2^k \sum_{\beta \in V^\perp} \mathcal{F}(D_\beta f) \end{aligned}$$

since $\sum_{\alpha \in V}(-1)^{\alpha \cdot \beta}$ equals 2^k if $\beta \in V^\perp$ and it equals 0 otherwise. □

We first consider the case where a function f with n variables satisfies the propagation criterion with respect to any $\beta \neq 0$ belonging to an hyperplane. We will use the following well-known lemma due to Jacobi (see [8, Ch. VI]):

Lemma 3. *Let n be an integer, $n > 2$, and let X and Y be two even integers. Then the condition $X^2 + Y^2 = 2^{n+1}$ implies*

- *if n is even, then $X^2 = Y^2 = 2^n$;*
- *if n is odd, then $X^2 = 2^{n+1}$ and $Y = 0$ – or vice-versa.*

For odd values of n, the functions with n variables having balanced derivatives $D_\beta f$ for every nonzero β in an hyperplane can be characterized as follows:

Theorem 4. *Let n be an odd integer, $n > 2$, and f be a Boolean function with n variables. Then the following properties are equivalent.*

(i) *There is an hyperplane $H \subset \mathbf{F}_2^n$ such that f satisfies the propagation criterion with respect to $H \setminus \{0\}$.*

(ii) *f has a three-valued extended Walsh spectrum, $\mathcal{L}(f)$ equals $2^{(n+1)/2}$ and there is some $a \in \mathbf{F}_2^n$ such that*

$$\forall \beta \in \mathbf{F}_2^n, \quad \mathcal{F}^2(f + \phi_\beta) \neq \mathcal{F}^2(f + \phi_{\beta+a}) \ .$$

(iii) *There is a linear permutation π of $\mathbf{F}_2^n$ such that $f \circ \pi(x_1, \ldots, x_n) = (1 + x_n)g + x_n h$ where both g and h are bent functions with $(n-1)$ variables.*

Proof: (i) ⇒ (ii) . Let $a \in \mathbf{F}_2^n$ be such that $H = \{x \in \mathbf{F}_2^n, a \cdot x = 0\}$. Proposition 5 gives for any $\beta \in H$

$$\mathcal{F}^2(f + \phi_\beta) + \mathcal{F}^2(f + \phi_{\beta+a}) = 2 \sum_{\alpha \in H} \mathcal{F}(D_\alpha(f + \phi_\beta)) \ .$$

Since $D_\alpha(f + \phi_\beta) = D_\alpha(f) + \alpha \cdot \beta$, we have

$$\mathcal{F}^2(f + \phi_\beta) + \mathcal{F}^2(f + \phi_{\beta+a}) = 2 \sum_{\alpha \in H} (-1)^{\alpha \cdot \beta} \mathcal{F}(D_\alpha f) = 2\mathcal{F}(D_0 f) = 2^{n+1} \ .$$

From Lemma 3, we deduce that, for any $\beta \in H$, $\mathcal{F}^2(f + \phi_\beta) = 2^{n+1}$ and $\mathcal{F}^2(f + \phi_{\beta+a}) = 0$, or vice-versa. It then follows that, for any $\beta \in \mathbf{F}_2^n$, $\mathcal{F}(f + \phi_\beta)$ belongs to $\{0, \pm 2^{(n+1)/2}\}$ and that $\mathcal{F}^2(f + \phi_\beta) \neq \mathcal{F}^2(f + \phi_{\beta+a})$.

(ii) ⇒ (iii). Let $(e_1, \ldots, e_n)$ denote the canonical basis of $\mathbf{F}_2^n$. Let π be a linear permutation of $\mathbf{F}_2^n$ such that $\pi^{-1}(a) = e_n$. Assertion **(ii)** gives for any $\beta \in \mathbf{F}_2^n$,

$$\mathcal{F}^2(f \circ \pi + \phi_\beta) + \mathcal{F}^2(f \circ \pi + \phi_{\beta+e_n}) = 2^{n+1} \ . \tag{1}$$

For any β in the hyperplane spanned by $e_1, \ldots, e_{n-1}$, ϕ_β does not depend on x_n. We then have $\phi_\beta(x_1, \ldots, x_n) = \phi(x_1, \ldots, x_{n-1})$ where ϕ describes the set of all

linear functions with $(n-1)$ variables when β varies. Using the decomposition $f \circ \pi(x_1, \ldots, x_n) = (1 + x_n)g + x_n h$, we obtain

$$\mathcal{F}(f \circ \pi + \phi_\beta) = \mathcal{F}(g + \phi) + \mathcal{F}(h + \phi) \text{ and}$$
$$\mathcal{F}(f \circ \pi + \phi_{\beta + e_n}) = \mathcal{F}(g + \phi) - \mathcal{F}(h + \phi) \ .$$

Equation (1) now gives

$$\mathcal{F}^2(g + \phi) + \mathcal{F}^2(h + \phi) = \frac{1}{2}\left(\mathcal{F}^2(f \circ \pi + \phi_\beta) + \mathcal{F}^2(f \circ \pi + \phi_{\beta + e_n})\right) = 2^n \ .$$

We deduce from Lemma 3 that, for any linear function ϕ, both $\mathcal{F}^2(g + \phi)$ and $\mathcal{F}^2(h + \phi)$ equal 2^{n-1}, and thus that g and h are bent.

(iii) $\Rightarrow$ (i). Let H' be the hyperplane spanned by $e_1, \ldots, e_{n-1}$. For any $\alpha \in H'$, $D_\alpha(f \circ \pi)$ can be decomposed as

$$D_\alpha(f \circ \pi)(x_1, \ldots, x_n) = (1 + x_n) D_\alpha g(x_1, \ldots, x_{n-1}) + x_n D_\alpha h(x_1, \ldots, x_{n-1}) \ .$$

If g and h are bent, the derivatives $D_\alpha g$ and $D_\alpha h$ are balanced for any $\alpha \in H'$, $\alpha \neq 0$. It follows that $D_\alpha(f \circ \pi)$ is balanced and thus $D_\alpha f$ is balanced for any nonzero α in $\pi(H')$. □

Remark 1. Assertion **(iii)** can actually be generalized. For any vector $\alpha \in \mathbf{F}_2^n$, the restrictions of a Boolean function with n variables to $H_\alpha = \{x \in \mathbf{F}_2^n,\ \alpha \cdot x = 0\}$ and to its complementary set can be identified with Boolean functions with $(n-1)$ variables. Moreover, $\alpha \notin H_\alpha$ if and only if $\sum_{i=1}^n \alpha_i$ is odd. In this case, $\mathbf{F}_2^n$ is the direct sum of H_α and $H_\alpha^\perp$. Exactly as in the previous theorem, we can prove that if f satisfies **(i)** then for any $\alpha \in \mathbf{F}_2^n$ such that $\sum_{i=1}^n \alpha_i$ is odd, there exists a linear permutation π of $\mathbf{F}_2^n$ such that both restrictions of f to H_α and to its complementary set are bent.

When the number of variables is even, we obtain a similar result which provides new characterizations of bent functions. The detailed proof, which relies on the same arguments as the previous one, can be found in [4].

Theorem 5. *Let n be an even integer, $n > 2$, and f be a Boolean function with n variables. Then the following properties are equivalent.*

- **(i)** *There is an hyperplane $H \subset \mathbf{F}_2^n$ such that f satisfies the propagation criterion with respect to $H \setminus \{0\}$.*
- **(ii)** *For any hyperplane $H \subset \mathbf{F}_2^n$, f satisfies the propagation criterion with respect to $H \setminus \{0\}$.*
- **(iii)** *f is bent.*
- **(iv)** *$f(x_1, \ldots, x_n) = (1 + x_n)g + x_n h$ where both g and h are almost optimal functions with $(n-1)$ variables having a three-valued extended Walsh spectrum and, for any linear function ϕ with $(n-1)$ variables, we have*

$$\mathcal{F}^2(g + \phi) \neq \mathcal{F}^2(h + \phi) \ .$$

As pointed out in the remark following Theorem 4, Property **(iv)** also holds if we consider the decomposition of a bent function with respect to any vector α such that $\sum_{i=1}^{n} \alpha_i$ is odd. Note that this theorem is of interest for effective purposes: for checking that a function f is bent it is sufficient to compute the $\mathcal{F}(D_\alpha f)$ for α in some hyperplane.

Similar techniques provide the following result for functions satisfying the propagation criterion with respect to a linear subspace of codimension 2.

Theorem 6. *Let f be a Boolean function with n variables, $n > 2$. Assume that there exists a linear subspace $V \subset \mathbf{F}_2^n$ of codimension 2 such that f satisfies the propagation criterion with respect to $V \setminus \{0\}$.*

- *If n is odd, then f is an almost optimal function with a three-valued extended Walsh spectrum and there is a linear permutation π of $\mathbf{F}_2^n$ such that*

$$f \circ \pi(x_1, \ldots, x_n) = (1 + x_{n-1})(1 + x_n)g_{00} + x_{n-1}(1 + x_n)g_{10} + (1 + x_{n-1})x_n g_{01} + x_{n-1}x_n g_{11}$$

  where all g_{ij} are almost optimal functions with $(n-2)$ variables having a three-valued extended Walsh spectrum.
- *If n is even, then f is either bent or it satisfies $\mathcal{L}(f) = 2^{(n+2)/2}$ and its Walsh coefficients belong to $\{0, \pm 2^{n/2}, \pm 2^{(n+2)/2}\}$. Moreover, there is a linear permutation π of $\mathbf{F}_2^n$ such that*

$$f \circ \pi(x_1, \ldots, x_n) = (1 + x_{n-1})(1 + x_n)g_{00} + x_{n-1}(1 + x_n)g_{10} + (1 + x_{n-1})x_n g_{01} + x_{n-1}x_n g_{11}$$

  where the Walsh coefficients of all g_{ij} belong to $\{0, \pm 2^{(n-2)/2}, \pm 2^{n/2}\}$.

Converses are not valid in Theorem 6: for odd n, there exist some functions which are not almost optimal and whose restrictions are almost optimal and have a three-valued extended Walsh spectrum. Moreover, the set of all functions satisfying the propagation criterion with respect to a subspace of codimension 2 does not contain all almost optimal functions with a three-valued extended Walsh spectrum.

Example 3. Let $f(x_1, \ldots, x_7) = x_1x_2x_3x_4 + x_1x_3x_5x_6 + x_1x_2x_3 + x_1x_3x_7 + x_1x_2 + x_3x_4 + x_5x_6$. This almost optimal function has a three-valued extended Walsh spectrum but the set $\{\alpha \in F_2^7, \mathcal{F}(D_\alpha f) = 0\} \cup \{0\}$ does not contain any linear space of dimension 5.

Theorems 4 and 6 can be used for generalizing some results given in [31]: any Boolean function with an odd number of variables which has at most 7 nonzero auto-correlation coefficients is almost optimal and it has a three-valued extended Walsh spectrum. This result does not hold anymore when f has 8 nonzero auto-correlation coefficients:

Example 4. For any odd $n \geq 5$, the function

$$f(x_1, \ldots, x_n) = x_2x_3x_4x_5 + x_1x_4x_5 + x_3x_5 + x_2x_4 + g(x_6, \ldots, x_n) \qquad (2)$$

where g is any bent function with $(n–5)$ variables, is such that $\{\alpha \in \mathbf{F}_2^n, \mathcal{F}(D_\alpha f) \neq 0\} = Span(e_1, e_2, e_3)$. This function satisfies $\mathcal{L}(f) = 2^{(n+1)/2}$ but its extended Walsh spectrum has exactly 5 values, $0, \pm 2^{(n-1)/2}, \pm 2^{(n+1)/2}$. Moreover, its sum-of-squares indicator is $\mathcal{V}(f) = 2^{2m-3}$ [1]. Since the bent function g can take any degree less than or equal to $(n-5)/2$, the function defined in (2) can be obtained for any degree d, $4 \leq d \leq (n-5)/2$. Other almost optimal functions whose extended Walsh spectra have more than 3 values can be found in [10, 11].

7 Correlation-immunity of Boolean functions with a three-valued extended Walsh spectrum

We now show that most functions with a three-valued extended Walsh spectrum can be easily transformed into a 1-resilient function, *i.e.* into a function which is balanced and first-order correlation-immune. Since the values of the extended Walsh spectrum are symmetric with respect to 0, if the extended Walsh spectrum of a function has exactly three values, then one of these values is 0. Such a function can therefore be transformed (by addition of a linear function) into a balanced function which have the same extended Walsh spectrum.

Theorem 7. *Let f be balanced Boolean function with n variables. Assume that its extended Walsh spectrum takes three values. Then there exists a linear permutation of $\mathbf{F}_2^n$ such that $f \circ \pi$ is 1-resilient if and only if there is no linear permutation π' of $\mathbf{F}_2^n$ such that $f \circ \pi'$ satisfies PC$(n-1)$.*

Proof: Recall that Proposition 3 asserts that f can be transformed into a 1-resilient function if and only if $E_{CI}(f)$ has rank n. We know from Theorem 2 that $\mathcal{L}(f) = 2^i$ for some $i \geq n/2$ and that the number of zero Walsh coefficients of f is $|E_{CI}(f)| = 2^n - 2^{2n-2i}$. Since f is balanced, it can not be bent and thus $i \geq (n+1)/2$. It follows that $|E_{CI}(f)| \geq 2^{n-1}$ with equality if and only if $i = (n+1)/2$. We obviously deduce that $E_{CI}(f)$ has full rank when $\mathcal{L}(f) > 2^{(n+1)/2}$. Let us now assume that n is odd and $\mathcal{L}(f) = 2^{(n+1)/2}$. The only case where $E_{CI}(f)$ does not have full rank is when it is an hyperplane of $\mathbf{F}_2^n$. Let $\{0, a\} = E_{CI}(f)^\perp$. Proposition 5 applied to $E_{CI}(f)$ leads to

$$0 = \sum_{\alpha \in E_{CI}(f)} \mathcal{F}^2(f + \phi_\alpha) = 2^{n-1}\left(\mathcal{F}(D_0 f) + \mathcal{F}(D_a f)\right) = 2^{n-1}(2^n + \mathcal{F}(D_a f)) \ .$$

Thus $\mathcal{F}(D_a f) = -2^n$; f is then an almost optimal function which has a linear structure. From Corollary 2 we deduce that f can be transformed into a function satisfying PC$(n-1)$.

Conversely, if there is a linear permutation π such $f \circ \pi$ satisfies PC$(n-1)$ then $\mathcal{L}(f) = 2^{(n+1)/2}$ and f has a linear structure a. We now apply Proposition 5

to $H = \{0, a\}^\perp$:

$$\sum_{\alpha \in H} \mathcal{F}^2(f + \phi_\alpha) = 2^{n-1}(2^n + \mathcal{F}(D_a f)) = 0 \text{ or } 2^{2n} \ .$$

Since by hypothesis $f \circ \pi$ is balanced, we have that

$$\sum_{\alpha \in H} \mathcal{F}^2(f + \phi_\alpha) \leq \mathcal{L}(f)^2(2^{n-1} - 1) < 2^{2n} \ .$$

Thus $\sum_{\alpha \in H} \mathcal{F}^2(f + \phi_\alpha) = 0$. It follows that $\mathcal{F}(f + \phi_\alpha) = 0$ for all $\alpha \in H$. Since $|E_{CI}(f)| = 2^{n-1}$, we deduce that $E_{CI}(f) = H$ and thus it has rank $n - 1$. □

For any odd n, 1-resilient functions with n variables having nonlinearity $2^{n-1} - 2^{(n-1)/2}$ can then be easily constructed. According to Theorem 5 it is sufficient to consider the restriction of a bent function with $(n + 1)$ variables to any hyperplane $\{x \in \mathbf{F}_2^{n+1},\ \alpha \cdot x\}$ where $\sum_{i=1}^{n+1} \alpha_i$ is odd. We then only have to check that this function has no linear structure and we transform it by addition of an appropriate linear function and by composition with a linear permutation.

Corollary 3. *Let n be an odd integer. For any integer d, $2 \leq d \leq (n+1)/2$, there exists a 1-resilient function with n variables having degree d and nonlinearity $2^{n-1} - 2^{(n-1)/2}$.*

Proof: We consider the following bent function with $(n + 1)$ variables which belongs to the Maiorana-McFarland class [9]:

$$\forall (x, y) \in \mathbf{F}_2^{\frac{n+1}{2}} \times \mathbf{F}_2^{\frac{n+1}{2}}, \quad f(x, y) = x \cdot \pi(y) + h(y)$$

where h is any Boolean function with $(n+1)/2$ variables and π is the permutation of $\mathbf{F}_2^{\frac{n+1}{2}}$ identified with the power function $x \mapsto x^s$ over $\mathbf{F}_{2^{\frac{n+1}{2}}}$. We choose for example $s = 2^k + 1$ with $k < (n + 1)/2$ and $\frac{n+1}{2\gcd(k,(n+1)/2)}$ odd, or $s = 7$ when $(n + 1)$ is power of 2. Let g be the restriction of f to the hyperplane $\{x \in \mathbf{F}_2^{n+1}, x_1 = 0\}$. The restriction of f has no linear structure when all derivatives of f have degree at least 2. Here we have for any (α, β),

$$D_{(\alpha,\beta)} f(x, y) = \alpha \cdot \pi(y + \beta) + x \cdot (\pi(y + \beta) + \pi(y)) + D_\beta g(y) \ .$$

Our choice for permutation π implies that the degree of $D_{(\alpha,\beta)} f$ is at least 2 when $(\alpha, \beta) \neq (0, 0)$ (see e.g. [19]). It follows that g has no linear structure; it can therefore be transformed into a 1-resilient almost optimal function. Since there is no restriction on h, h can be chosen of any degree less than or equal to $(n + 1)/2$. Thus g can take any degree d, $4 \leq d \leq (n + 1)/2$. Note that such almost optimal functions of degree 2 and 3 can easily be constructed from the functions with 5 variables given in [1]. □

Note that Sarkar and Maitra [24] provide a construction method for 1-resilient functions with n variables having nonlinearity $2^{n-1} - 2^{(n-1)/2}$ and degree $(n-2)$, for any odd $n \geq 5$.

References

1. E.R. Berlekamp and L.R. Welch. Weight distributions of the cosets of the (32,6) Reed-Muller code. *IEEE Trans. Inform. Theory*, 18(1):203–207, 1972.
2. E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In *Advances in Cryptology - EUROCRYPT'99*, number 1592 in Lecture Notes in Computer Science, pages 12–23. Springer-Verlag, 1999.
3. E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
4. A. Canteaut, C. Carlet, P. Charpin, and C. Fontaine. On cryptographic properties of the cosets of $R(1, m)$. Submitted.
5. C. Carlet. Partially-bent functions. In *Advances in Cryptology - CRYPTO'92*, number 740 in Lecture Notes in Computer Science, pages 280–291. Springer-Verlag, 1992.
6. C. Carlet. Two new classes of bent functions. In *Advances in Cryptology - EUROCRYPT'93*, number 765 in Lecture Notes in Computer Science, pages 77–101. Springer-Verlag, 1994.
7. C. Carlet. On the propagation criterion of degree ℓ and order k. In *Advances in Cryptology - EUROCRYPT'98*, number 1403 in Lecture Notes in Computer Science, pages 462–474. Springer-Verlag, 1998.
8. L. E. Dickson. *History of the Theory of Numbers*, volume II. Chelsea Publishing Compagny, 1919.
9. J.F. Dillon. *Elementary Hadamard Difference sets.* PhD thesis, University of Maryland, 1974.
10. E. Filiol and C. Fontaine. Highly nonlinear balanced Boolean functions with a good correlation-immunity. In *Advances in Cryptology - EUROCRYPT'98*, number 1403 in Lecture Notes in Computer Science, pages 475–488. Springer-Verlag, 1998.
11. C. Fontaine. On some cosets of the First-Order Reed-Muller code with high minimum weight. *IEEE Trans. Inform. Theory*, 45(4):1237–1243, 1999.
12. T. Johansson and F. Jönsson. Fast correlation attacks based on turbo code techniques. In *Advances in Cryptology - CRYPTO'99*, number 1666 in Lecture Notes in Computer Science, pages 181–197. Springer-Verlag, 1999.
13. T. Johansson and F. Jönsson. Improved fast correlation attack on stream ciphers via convolutional codes. In *Advances in Cryptology - EUROCRYPT'99*, number 1592 in Lecture Notes in Computer Science, pages 347–362. Springer-Verlag, 1999.
14. F.J. MacWilliams and N.J.A. Sloane. *The theory of error-correcting codes.* North-Holland, 1977.
15. M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - EUROCRYPT'93*, number 765 in Lecture Notes in Computer Science. Springer-Verlag, 1994.
16. R.J. McEliece. Weight congruence for p-ary cyclic codes. *Discrete Mathematics*, 3:177–192, 1972.
17. W. Meier and O. Staffelbach. Fast correlation attack on certain stream ciphers. *Jounal of Cryptology*, pages 159–176, 1989.
18. W. Meier and O. Staffelbach. Nonlinearity criteria for cryptographic functions. In *Advances in Cryptology - EUROCRYPT'89*, number 434 in Lecture Notes in Computer Science, pages 549–562. Springer-Verlag, 1990.
19. K. Nyberg. Differentially uniform mappings for cryptography. In *Advances in Cryptology - EUROCRYPT'93*, number 765 in Lecture Notes in Computer Science, pages 55–64. Springer-Verlag, 1993.

20. N.J. Patterson and D.H. Wiedemann. The covering radius of the $[2^{15}, 16]$ Reed-Muller code is at least 16276. *IEEE Trans. Inform. Theory*, IT-36(2):443, 1983.
21. N.J. Patterson and D.H. Wiedemann. Correction to [20]. *IEEE Trans. Inform. Theory*, IT-36(2):443, 1990.
22. B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle. Propagation characteristics of Boolean functions. In *Advances in Cryptology - EUROCRYPT'90*, number 437 in Lecture Notes in Computer Science, pages 155–165, Springer-Verlag, 1990. Springer-Verlag.
23. O.S. Rothaus. On bent functions. *J. Combin. Theory Ser. A*, 20:300–305, 1976.
24. P. Sarkar and S. Maitra. Construction of nonlinear boolean functions with important cryptographic properties. In *Advances in Cryptology - EUROCRYPT 2000*, number 1807 in Lecture Notes in Computer Science, pages 491–512, Springer-Verlag, 2000 (this volume).
25. C.E. Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28:656–715, 1949.
26. T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Trans. Inform. Theory*, IT-30(5):776–780, 1984.
27. T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Computers*, C-34(1):81–84, 1985.
28. A.F. Webster and S.E. Tavares. On the design of S-boxes. In *Advances in Cryptology - CRYPTO'85*, number 219 in Lecture Notes in Computer Science, pages 523–534. Springer-Verlag, 1985.
29. G. Xiao and J.L. Massey. A spectral characterization of correlation-immune combining functions. *IEEE Trans. Inform. Theory*, IT-34(3):569–571, 1988.
30. X.-M. Zhang and Y. Zheng. GAC - the criterion for global avalanche characterics of cryptographic functions. *Journal of Universal Computer Science*, 1(5):320–337, 1995.
31. X.-M. Zhang and Y. Zheng. Characterizing the structures of cryptographic functions satisfying the propagation criterion for almost all vectors. *Designs, Codes and Cryptography*, 7(1):11–134, 1996.
32. Y. Zheng and X.-M. Zhang. Plateaued functions. In *Information and Communication Security, ICICS'99*, number 1726 in Lecture Notes in Computer Science, pages 284–300. Springer-Verlag, 1999.

Cox-Rower Architecture for Fast Parallel Montgomery Multiplication

Shinichi Kawamura[1], Masanobu Koike[2], Fumihiko Sano[2], and Atsushi Shimbo[1]

[1] Toshiba Research and Development Center
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki, 212-8582, Japan
[2] Toshiba System Integration Technology Center
3-22, Katamachi Fuchu-shi, Tokyo, 183-8512, Japan
{shinichi2.kawamura, masanobu2.koike, fumihiko.sano, atsushi.shimbo}
@toshiba.co.jp

Abstract. This paper proposes a fast parallel Montgomery multiplication algorithm based on Residue Number Systems (RNS). It is easy to construct a fast modular exponentiation by applying the algorithm repeatedly. To realize an efficient RNS Montgomery multiplication, the main contribution of this paper is to provide a new RNS base extension algorithm. Cox-Rower Architecture described in this paper is a hardware suitable for the RNS Montgomery multiplication. In this architecture, a base extension algorithm is executed in parallel by plural Rower units controlled by a Cox unit. Each Rower unit is a single-precision modular multiplier-and-accumulator, whereas Cox unit is typically a 7 bit adder. Although the main body of the algorithm processes numbers in an RNS form, efficient procedures to transform RNS to or from a radix representation are also provided. The exponentiation algorithm can, thus, be adapted to an existing standard radix interface of RSA cryptosystem.

1 Introduction

Many researchers have been working on how to implement public key cryptography faster. A fast modular multiplication for large integers is of special interest because it gives a basis for a fast modular exponentiation which is used for many cryptosystems such as, RSA, Rabin, Diffie-Hellman and ElGmal. Recent improvement of factoring an integer leads to a recommendation that one should use a longer key size. So, even faster algorithms are required. A lot of work has been done with a view to realizing a fast computation in a radix representation. It might seem that in a radix representation, all the major performance improvements have been achieved. Nevertheless, use of the Residue Number Systems (RNS) appears to be a promising approach for achieving a breakthrough.

RNS is a method of representing an integer with a set of its residues in terms of a given base which is a set of relatively prime moduli. A well-known advantage of RNS is that if addition, subtraction, or multiplication are to be done, the computation for each RNS element can be carried out independently. If n processing units perform the computation, the processing speed will be n times

faster. So, RNS is studied with a view to its application in the areas where fast and parallel processing methods are required[1–4]. Digital signal processing is one such area. As for cryptographic applications, a paper by Quisquater, et al.[5] was the first report on the application of RNS to RSA cryptosystem[6]. With respect to RNS, however, it deals with a limited case since one cannot choose an arbitrary base, rather one has to choose secret keys p and q as the RNS base. Thus, it can be applied only to decryption. The disadvantages of RNS are that division and comparison are not efficiently implemented. Therefore, although RNS is considered to be a good candidate for a fast and parallel computation for public key cryptography, it was not until the early 90's that RNS was shown to be really applicable for that purpose.

To overcome the disadvantages of RNS, a novel approach to combine RNS with Montgomery multiplication was proposed. The idea behind this is that since Montgomery multiplication effectively avoids the division in a radix representation, it is expected to be effective for avoiding difficulties in implementing division in RNS as well. To the best of our knowledge, Posch, et al. are the first who invented an RNS Montgomery multiplication[7]. Other works [9] and [11] also discuss RNS Montgomery multiplications. These works deal with methods where RNS base can be chosen almost independently of secret keys p and q. So, these algorithms can be applied to RSA encryption as well as decryption. Note that Paillier's algorithm in [11] is aimed at a special case where the base size is limited to 2. The latter two systems,[9] and [11], are partly based on a mixed radix representation. It seems to us that a fully parallel computation cannot be realized in this setting and thus the methods are slower. So far, Posch, et al.'s method seems the fastest for a parallel hardware and general parameters.

According to three forerunners above, most of the processing time for RNS Montgomery multiplication is devoted to base extensions. A base extension is a procedure to transform a number represented in an RNS base into that in another base, the subset of which is the original base. So, the main contribution of this paper is to provide a new base extension algorithm. This results in a new RNS Montgomery multiplication algorithm which requires less hardware and is more sophisticated than Posch, et al.'s. It is easy to realize modular exponentiation algorithm by applying the RNS Montgomery multiplication repeatedly. In addition, it is important that the algorithm can be adapted to an existing standard RSA interface, i.e., usually, a radix representation. Therefore, another purpose of this paper is to provide efficient ways to transform RNS to or from a radix representation.

This paper is organized as follows: Section 2 briefly describes basic notions such as an RNS representation, a Montgomery multiplication, and an RNS Montgomery multiplication. In section 3, a new base extension algorithm is proposed, which plays an important role in an RNS Montgomery multiplication. Section 4 presents Cox-Rower Architecture and the RNS Montgomery multiplication algorithm, which is applied to construct an exponentiation algorithm. Transformations between RNS and a radix representation are shown as well. Section

5 deals with implementation issues such as parameter design and performance. Section 6 concludes the paper.

2 Preliminaries

2.1 Residue Number Systems

Usually, a number is expressed in a radix representation. A radix 2^r representation of x is n-tuple $(x_{(n-1)}, \cdots, x_{(0)})$ which satisfies

$$x = \sum_{i=0}^{n-1} x_{(i)} 2^{ri} = (2^{r(n-1)}, \ldots, 2^r, 1) \begin{pmatrix} x_{(n-1)} \\ \vdots \\ x_{(1)} \\ x_{(0)} \end{pmatrix} \quad (1)$$

where, $0 \leq x_{(i)} \leq 2^r - 1$.

Residue Number Systems (RNS) are also a method for representing a number. Let $< x >_a$ denote an RNS representation of x, then

$$< x >_a = (x[a_1], x[a_2], \cdots, x[a_n])$$

where, $x[a_i] = x \bmod a_i$. The set $a = \{a_1, a_2, \cdots, a_n\}$ is called a base whose number of elements is called a base size. In this example, a base size is n. We require here that $\gcd(a_i, a_j) = 1$ (if $i \neq j$).

According to the Chinese remainder theorem, x can be computed from $< x >_a$ as

$$x = \left(\sum_{i=1}^{n} x[a_i] A_i^{-1}[a_i] A_i \right) \bmod A = \left(\sum_{i=1}^{n} (x[a_i] A_i^{-1}[a_i] \bmod a_i) A_i \right) \bmod A \quad (2)$$

where, $A = \prod_{i=1}^{n} a_i$, $A_i = A/a_i$, and $A_i^{-1}[a_i]$ is a multiplicative inverse of A_i modulo a_i. In Equation (2), the expression in the middle is a general form, whereas our base extension is based on the last one.

In the following section, we use two different bases, a and b, to realize an RNS modular multiplication. They are assumed to satisfy $\gcd(A, B) = 1$. A symbol m is sometimes used instead of a or b when the symbol can be replaced by either a or b. We also use a convention that $< z >_{a \cup b} = (< x >_a, < y >_b)$ which means that z is a number that satisfies $z \equiv x (\bmod A)$, $z \equiv y (\bmod B)$, and $z < AB$.

The advantages of an RNS representation are that addition, subtraction, and multiplication are simply realized by modular addition, subtraction, and multiplication of each element:

$$< x >_a \pm < y >_a = ((x[a_1] \pm y[a_1])[a_1], \cdots, (x[a_n] \pm y[a_n])[a_n])$$
$$< x >_a \cdot < y >_a = ((x[a_1] y[a_1])[a_1], \cdots, (x[a_n] y[a_n])[a_n]) .$$

Since each element is independently computed, if n computation units run in parallel, this computation finishes within a time required for a single operation of the unit. The disadvantages of an RNS representation are that it is comparatively difficult to perform comparison and division[12].

2.2 Montgomery Multiplication

Montgomery's modular multiplication method without division is a standard method in a radix representation to implement a public key cryptography which requires modular reduction[13]. The algorithm is presented in five steps below whose inputs are x, y, and N ($x, y < N$), and the output is $w \equiv xyR^{-1} (\bmod N)$, where $w < 2N$.

1: $s \leftarrow xy$
2: $t \leftarrow s \cdot (-N^{-1}) \bmod R$
3: $u \leftarrow t \cdot N$
4: $v \leftarrow s + u$
5: $w \leftarrow v/R$

where, $\gcd(R, N) = 1$ and $N < R$. In step 2, t is computed so that v is a multiple of R. Actually, assume that v is a multiple of R, i.e., $v \bmod R = 0$, then $(s + tN) \bmod R = 0$. This equation is solved as $t \equiv -sN^{-1} (\bmod R)$, which is equivalent to the computation in step 2. R must be chosen so that steps 2 and 5 are efficiently computed. It is usually chosen to be 2's power in a radix 2 representation. $\gcd(R, N) = 1$ ensures existence of $N^{-1} \bmod R$. Condition $N < R$ is sufficient for $w < 2N$ because $w = (xy + tN)/R < (N^2 + RN)/R = (N/R + 1)N < 2N$. Since $wR = xy + tN$, $wR \equiv xy \ (\bmod N)$ holds. By multiplying $R^{-1} \bmod N$ on both sides, $w \equiv xyR^{-1} (\bmod N)$ is obtained. The Montgomery multiplication is also useful for avoiding inefficient divisions in RNS.

2.3 Montgomery Multiplication in RNS

To derive an RNS Montgomery multiplication algorithm, we introduce two RNS bases a and b, and translate 5 steps in the previous section into the RNS computation in base $a \cup b$. It is assumed that A and B is chosen sufficiently large, so that all intermediate values are less than AB. Under this assumption, steps 1, 3, and 4 in the previous section is easily transformed into RNS form. For instance, step 1 will be performed by $< s >_{a \cup b} = < x >_{a \cup b} \cdot < y >_{a \cup b}$.

As for step 2, a constant R is set to $B = \prod_{i=1}^{n} b_i$. Then, t can be computed simply by $< t >_b = < s >_b < -N^{-1} >_b$. It is necessary, however, that $< t >_{a \cup b}$ is derived from $< t >_b$ so that the computation in base $a \cup b$ is continued. In this paper, such a procedure is called a base extension, where a number represented in either base a or base b is transformed into that in base $a \cup b$.

The remaining step is 5. Since v is a multiple of B, w is an integer which satisfies $v = wB$. So, if A is larger than w, w can be computed by $< w >_a = < v >_a \cdot < B^{-1} >_a$. Note that base b representation is unnecessary to realize step 5 in RNS. In addition, base b representation in step 4 is always $< v >_b = < 0 >_b$, because v is a multiple of B. So, the computation in base b at steps 3 and 4 can be skipped as well.

Figure 1 shows an overview of the RNS Montgomery multiplication algorithm. In this Figure, operations in base a and base b are shown separately. Each step corresponds to the step of the same number in the previous section.

Input: $< x >_{a\cup b}, < y >_{a\cup b}$, (where $x,\ y < 2N$)		
Output: $< w >_{a\cup b}$ (where $w \equiv xyB^{-1} (\mathrm{mod} N), w < 2N$)		
	Base a Operation	Base b Operation
1:	$< s >_a \leftarrow < x >_a \cdot < y >_a$	$< s >_b \leftarrow < x >_b \cdot < y >_b$
2a:	—	$< t >_b \leftarrow < s >_b \cdot < -N^{-1} >_b$
2b:	$< t >_{a\cup b} \Leftarrow < t >_b$	
3:	$< u >_a \leftarrow < t >_a \cdot < N >_a$	—
4:	$< v >_a \leftarrow < s >_a + < u >_a$	—
5a:	$< w >_a \leftarrow < v >_a \cdot < B^{-1} >_a$	—
5b:	$< w >_a \Longrightarrow < w >_{a\cup b}$	

Fig. 1. Overview of the Montgomery Multiplication in RNS

Almost the same procedure is provided by Posch, et al[7]. Note that the range of input is changed from less than N to less than $2N$. The purpose of it is to make the range of input and output compatible with each other, so that it becomes possible to construct a modular exponentiation algorithm by repeating the Montgomery multiplication. Base extension at step 5b is necessary for the same reason.

If the two base-extension steps in Fig.1 are error-free, we can specify the condition that A and B should satisfy for a given N. Condition that $\gcd(B, N) = 1$ and $\gcd(A, B) = 1$ is sufficient for the existence of N^{-1} mod B and B^{-1} mod A, respectively. $4N \leq B$ is also sufficient for $w < 2N$ to hold when $x, y < 2N$. Actually,

$$w = \frac{v}{B} = \frac{xy + tN}{B} < \frac{(2N)^2 + BN}{B} = \left(\frac{4N}{B} + 1\right) N \leq 2N.$$

This equation also shows that condition $2N \leq A$ is sufficient for $w < A$ and $v < AB$. Since v is the maximum intermediate value, all values are less than AB. In summary, the following four conditions are sufficient:

- $\gcd(B, N) = 1$,
- $\gcd(A, B) = 1$,
- $4N \leq B$, and
- $2N \leq A$.

Since the base extension algorithm proposed later introduces approximations, the last two conditions will be modified in section 4.1 by Theorem 3.

In Fig.1, if n processing units perform in parallel, the processing time is roughly estimated as the time for 5 single-precision modular multiplications plus two base extensions. Therefore the devising of a fast base extension algorithm is crucial for realizing a fast RNS Montgomery multiplication.

3 New Approach for Base Extension

3.1 Reduction Factor k

One might transform an RNS expression to another via a radix representation, i.e., $< x >_m \rightarrow x \rightarrow < x >_{m'}$ and thus, obtain $< x >_{m \cup m'} = (< x >_m, < x >_{m'})$. However, such a naive approach usually requires multi-precision integer arithmetic which it is preferable to avoid. Nevertheless, considering how to represent x with $< x >_m$'s elements is a key approach in our work as well as in [7], [9], and [11]. From Equation (2), there exists a unique integer k that satisfies

$$x = \sum_{i=1}^{n} (x[m_i] M_i^{-1}[m_i] \bmod m_i) M_i - kM. \tag{3}$$

In this paper, k is called a reduction factor. Our objective here is to represent k with known variables. Let us define a value ξ_i as

$$\xi_i = x[m_i] M_i^{-1}[m_i] \bmod m_i.$$

Then, Equation (3) is simplified as

$$x = \sum_{i=1}^{n} \xi_i M_i - kM. \tag{4}$$

Here unknown parameters are k and x. If both sides are divided by M, it follows that

$$\sum_{i=1}^{n} \frac{\xi_i}{m_i} = k + \frac{x}{M}. \tag{5}$$

Since $0 \leq x/M < 1$, $k \leq \sum_{i=1}^{n} \frac{\xi_i}{m_i} < k + 1$ holds. Therefore,

$$k = \left\lfloor \sum_{i=1}^{n} \frac{\xi_i}{m_i} \right\rfloor.$$

Here, $0 \leq k < n$ holds, because $0 \leq \xi_i / m_i < 1$. It is important that k is upperbounded by n. Due to this property our algorithm is simpler than Posch, et al.'s algorithm.

3.2 Approximate Representation for Factor k

In the previous section a close estimate for k is derived. It requires, however, division by base values which is in general not easy. To facilitate the computation, two approximations are introduced here:

- a denominator m_i is replaced by 2^r, where $2^{r-1} < m_i \leq 2^r$
- a numerator ξ_i is approximated by its most significant q bits, where $q < r$

In this paper it is assumed that r is common to all base elements to realize modularity of hardware, whereas in general r may be different for each m_i. With these approximations, $\widehat{k}$ is given by

$$\widehat{k} = \left\lfloor \sum_{i=1}^{n} \frac{\mathrm{trunc}(\xi_i)}{2^r} + \alpha \right\rfloor \tag{6}$$

where, $\mathrm{trunc}(\xi_i) = \xi_i \bigwedge (\overbrace{1\ldots1}^{q}\overbrace{0\ldots0}^{(r-q)})_{(2)}$, and $\bigwedge$ means a bitwise AND operation. An offset value α is introduced to compensate errors caused by approximation. Suggested values of α will be derived later. Since division by 2's power can be realized by virtually shifting the fixed point, the approximate value $\widehat{k}$ is computed by addition alone. Further, $\widehat{k}$ can be computed recursively bit by bit using the following equations with an initial value $\sigma_0 = \alpha$:

$$\sigma_i = \sigma_{i-1} + \mathrm{trunc}(\xi_i)/2^r, \quad k_i = \lfloor \sigma_i \rfloor, \quad \sigma_i = \sigma_i - k_i \quad (\text{for } i = 1, \cdots, n). \tag{7}$$

It is easy to show that the sequence k_i satisfies $\widehat{k} = \sum_{i=1}^{n} k_i$, and $k_i \in \{0, 1\}$.

To evaluate the effect of the approximation later, ϵs' and δs' are defined as

$$\epsilon_{m_i} = (2^r - m_i)/2^r, \quad \delta_{m_i} = (\xi_i - \mathrm{trunc}(\xi_i))/m_i \tag{8}$$

$$\epsilon_m = \mathrm{Max}(\epsilon_{m_i}), \quad \delta_m = \mathrm{Max}(\delta_{m_i}) \tag{9}$$

$$\epsilon = \mathrm{Max}(\epsilon_a, \epsilon_b), \quad \delta = \mathrm{Max}(\delta_a, \delta_b). \tag{10}$$

ϵ is due to a denominator's approximation and δ is related to a numerator's.

3.3 Recursive Base Extension Algorithm

Integrating Equations (4), (6), and (7), a main formula for a base extension from base m to base $m \cup m'$ is derived as

$$x[m'_i] = \left(\sum_{j=1}^{n} \{\xi_j M_j[m'_i] + k_j (m'_i - M[m'_i])\} \right) \bmod m'_i \quad (\text{for } {}^{\forall}i). \tag{11}$$

Figure 2 shows the overall base extension procedure, where step 7 corresponds to Equation (11) and steps 2, 4, 5, and 6 to Equation (7). n processing units are assumed to run in parallel. Each unit is dedicated to some m_i or m'_i and independently computes

$$c_j = (c_{j-1} + f_j g_j + d_j) \bmod m_i.$$

Since the algorithm introduces approximation, the base extension algorithm does not always output a correct value. The following two theorems state how much error will occur under two different conditions. Refer to Appendix A for their proofs.

Input: $< x >_m;\ m,\ m';\ \alpha$
Output: $< z >_{m \cup m'} = (< x >_m, < y >_{m'})$
Precomputation: $< M_i^{-1} >_m, < M_i >_{m'}\ (\text{for}^{\forall} i), < -M >_{m'}$
1: $\xi_i = x[m_i] \cdot M_i^{-1}[m_i] \bmod m_i\ (\text{for}^{\forall} i)$
2: $\sigma_0 = \alpha,\ y_{i,0} = 0\ (\text{for}^{\forall} i)$
3: **For** $j = 1, \cdots, n$, **compute**.
4: $\sigma_j = \sigma_{(j-1)} + \text{trunc}(\xi_j)/2^r$
5: $k_j = \lfloor \sigma_j \rfloor$ /* Comment: $k_j \in \{0, 1\}$ */
6: $\sigma_j = \sigma_j - k_j$
7: $y_{ij} = y_{i,(j-1)} + \xi_j \cdot M_j[m_i'] + k_j \cdot (-M)[m_i']\ (\text{for}^{\forall} i)$
8: **End for**
9: $y[m_i'] = y_{i,n} \bmod m_i'\ (\text{for}^{\forall} i)$

Fig. 2. Base Extension Algorithm (BE)

Theorem 1. *If* $0 \leq n(\epsilon_m + \delta_m) \leq \alpha < 1$ *and* $0 \leq x < (1 - \alpha)M$, *then* $\widehat{k} = k$ *and the algorithm* BE *(in Fig.2) extends the base without error, i.e.,* $z = x$ *holds with respect to output* $< z >_{m \cup m'}$.

Theorem 2. *If* $\alpha = 0$, $0 \leq n(\epsilon_m + \delta_m) < 1$ *and* $0 \leq x < M$, *then* $\widehat{k} = k$ *or* $k - 1$ *and the algorithm* BE *(in Fig.2) outputs* $< z >_{m \cup m'}$ *which satisfies* $z \equiv x (\text{mod} M)$ *and* $z < \{1 + n(\epsilon_m + \delta_m)\}M$.

Theorem 1 means that if an offset α is properly chosen, the algorithm BE is error-free so long as the input x is not too close to M. Note that x is not lowerbounded. On the other hand, Theorem 2 means that without an offset α, for any input x, the algorithm BE outputs a correct value or correct value plus M. As for Theorem 2, in [7], Posch, et al., observed a similar fact with respect to their own base extension algorithm.

4 Cox-Rower Architecture

4.1 RNS Montgomery Multiplication Algorithm

The Montgomery multiplication algorithm in Fig.3 is derived by integrating base extension algorithm in Fig.2 into the flow in Fig.1. At the base extension in step 4, an offset value is 0 and the extension error upperbounded by Theorem 2 will occur. In Fig.3, a symbol $\widehat{t}$ is used in place of t to imply extension error. In step 8, on the other hand, the offset value α is chosen so that the extension is error-free by Theorem 1. As will be shown later, typical offset value is 1/2.

By defining $\Delta = n(\epsilon + \delta)$, the theorem below ensures correctness of the algorithm. Refer to Appendix B for the proof.

Theorem 3. *If* (1) $gcd(N, B) = 1$, (2) $gcd(A, B) = 1$, (3) $0 \leq \Delta \leq \alpha < 1$, (4) $4N/(1 - \Delta) \leq B$, *and* (5) $2N/(1 - \alpha) \leq A$, *then for any input* $x, y < 2N$, *the algorithm* MM *(in Fig.3) outputs* $< w >_{a \cup b}$ *which satisfies* $w \equiv xyB^{-1} (\text{mod} N)$, $w < 2N$.

Condition (4) is derived to satisfy $w < 2N$. Conditions (3) and (5) are necessary in order that the base extension at step 8 is error-free. Conditions (4) and (5) are sufficient for the largest intermediate value v to be less than AB. Theorem 3 ensures that the range of an output w is compatible with that of inputs x and y. This allows us to use the algorithm repeatedly to construct a modular exponentiation.

Input:	$< x >_{a\cup b},\ < y >_{a\cup b}$ (where $x, y < 2N$)	
Output:	$< w >_{a\cup b}$ (where $w \equiv xyB^{-1} (\mathrm{mod} N),\ w < 2N$)	
Precomputation:	$< -N^{-1} >_b, < N >_a, < B^{-1} >_a$	
1:	$s[a_i] = x[a_i] \cdot y[a_i] \bmod a_i$	(for$^{\forall} i$)
2:	$s[b_i] = x[b_i] \cdot y[b_i] \bmod b_i$	(for$^{\forall} i$)
3:	$t[b_i] = s[b_i] \cdot (-N^{-1})[b_i] \bmod b_i$	(for$^{\forall} i$)
4:	$< \widehat{t} >_{a\cup b} \leftarrow BE(< t >_b;\ b,\ a;\ \ 0)$	
5:	$u[a_i] = \widehat{t}[a_i] \cdot N[a_i] \bmod a_i$	(for$^{\forall} i$)
6:	$v[a_i] = (s[a_i] + u[a_i]) \bmod a_i$	(for$^{\forall} i$)
7:	$w[a_i] = v[a_i] \cdot B^{-1}[a_i] \bmod a_i$	(for$^{\forall} i$)
8:	$< w >_{a\cup b} \leftarrow BE(< w >_a;\ a,\ b;\ \alpha > 0)$	

(Note: BE is the algorithm shown in Fig.2.)

Fig. 3. RNS Montgomery Multiplication Algorithm (MM)

Figure 4 shows a typical hardware structure suitable for the RNS Montgomery multiplication. There are n sets of Rower units and a Cox unit. Each Rower unit has a multiplier-and-accumulator with modular reduction by a_i or b_i. Cox unit consists of truncation unit, q-bit adder, and its output register. It computes $\widehat{k}$ bit by bit. Cox unit acts as if it directs the Rower units which compute the main part of a Montgomery multiplication.

Our proposal has an advantage over the Posch, et al.'s [7][8] in that the base extension in step 8 is error-free. This makes extra steps for error correction unnecessary in our algorithm. In addition, in our algorithm, the reduction factor k can be computed by addition alone, whereas a multiplier-and-accumulator similar to a Rower unit is required in their algorithm. Unlike Posch, et al.'s, there is no lower bound for N in our algorithm. This means an LSI which can execute 1024 bit RSA cryptosystem can also deal with 768 bit, 512 bit, and so on.

4.2 Exponentiation with RNS Montgomery Multiplication

Figure 5 shows an exponentiation algorithm based on the binary method, otherwise known as square-and-multiply method. The main loop of the algorithm is realized by the repetition of Montgomery multiplications in Fig.3. The first step of the algorithm transforms an input integer x into $x' = xB \bmod N$. The last

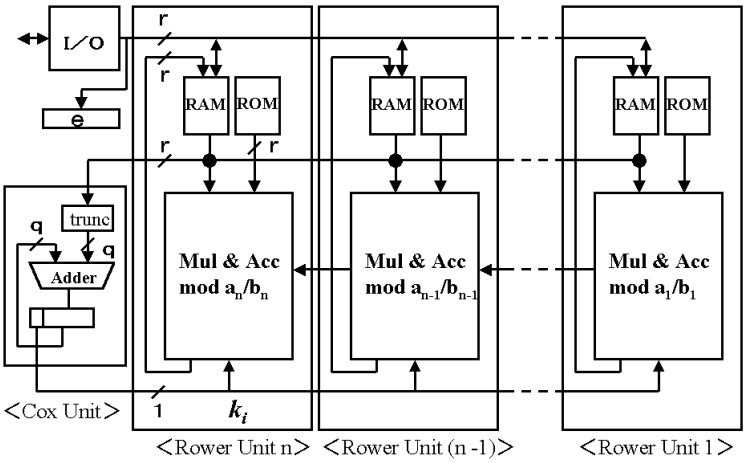


Fig. 4. The Cox-Rower Architecture

step is the inverse of the first step. It is possible to replace a binary exponentiation method by other more efficient methods such as a window method.

In [7] and [10], it was proposed that RNS should be used as the input and output representation of the algorithm, presumably to avoid further steps necessary for Radix-to-RNS and RNS-to-Radix transformations. Actually, they did not provide any Radix to or from RNS transformations. In order to adapt the architecture to an existing interface of the RSA cryptosystem, it seems important to provide Radix to or from RNS transformations suitable for the Cox-Rower Architecture. Such transformations will be provided in the following two sections.

Input: $< x >_{a\cup b}$, $e = (e_k, \cdots, e_1)_{(2)}$ (where $e_k = 1, k \geq 2$)
Output: $< y >_{a\cup b}$ (where $y \equiv x^e (\mathrm{mod} N)$, $y < 2N$)
Precomputation: $< B^2 \bmod N >_{a\cup b}$
1: $< x' >_{a\cup b} \leftarrow MM(< x >_{a\cup b}, < B^2 \bmod N >_{a\cup b})$
2: $< y >_{a\cup b} \leftarrow < x' >_{a\cup b}$
3: **For** $i = k-1, \cdots, 1$, **compute**.
4: $\quad < y >_{a\cup b} \leftarrow MM(< y >_{a\cup b}, < y >_{a\cup b})$
5: $\quad$ **If** $e_i = 1$, **then** $< y >_{a\cup b} \leftarrow MM(< y >_{a\cup b}, < x' >_{a\cup b})$
6: **End for**
7: $< y >_{a\cup b} \leftarrow MM(< y >_{a\cup b}, < 1 >_{a\cup b})$
(Note: MM is the algorithm shown in Fig.3.)

Fig. 5. RNS Modular Exponentiation Algorithm (EXP)

4.3 RNS-Radix Conversion

As Equation (4) is the basis for the whole algorithm, the equation is used as the basis for the RNS-to-Radix conversion. Radix-2^r representations for A_i and A are derived below.

$$A_i = (2^{r(n-1)}, \ldots, 2^r, 1) \begin{pmatrix} A_{i(n-1)} \\ \vdots \\ A_{i(1)} \\ A_{i(0)} \end{pmatrix}, \quad A = (2^{r(n-1)}, \ldots, 2^r, 1) \begin{pmatrix} A_{(n-1)} \\ \vdots \\ A_{(1)} \\ A_{(0)} \end{pmatrix}$$

By substituting these into Equation (4) and rearranging the equation, we obtain,

$$x = (2^{r(n-1)}, \ldots, 2^r, 1) \sum_{i=1}^{n} \left\{ \xi_i \begin{pmatrix} A_{i(n-1)} \\ \vdots \\ A_{i(1)} \\ A_{i(0)} \end{pmatrix} - k_i \begin{pmatrix} A_{(n-1)} \\ \vdots \\ A_{(1)} \\ A_{(0)} \end{pmatrix} \right\}. \tag{12}$$

Each row in Equation (12) can be computed in parallel by using the Cox-Rower Architecture. Note that in this case, carry should be accumulated in each unit while the n steps of summation are being continued. After the summation is finished, the saved carry is propagated from Rower unit 1 up to Rower unit n. The carry propagation circuit is shown in Fig.4 with arrows from Rower unit $(i-1)$ to i. This carry propagation requires n steps. The transformation is error-free if Conditions in Theorem 1 is satisfied.

Although the transformed value is error-free, the output value of the Montgomery multiplication itself may be larger than modulus N. Therefore it is necessary that N is subtracted from the transformed radix representation if it is larger than N. This is called a (final) correction, and is carried out in n steps on the same hardware.

4.4 Radix-RNS Conversion

Given a radix-2^r representation of x as $(x_{(n-1)}, \cdots, x_{(0)})$, we have to derive a method to compute $< x >_m$, that matches the Cox-Rower Architecture. By applying mod m_i operation to Equation (1), we obtain

$$x[m_i] = \left(\sum_{j=0}^{n-1} x_{(j)} \cdot (2^{rj}[m_i]) \right) \bmod m_i \ \ (\text{for } {}^{\forall}i).$$

If constant $2^{rj}[m_i]$ is precomputed, this computation is well suited to the Cox-Rower Architecture. The computation finishes in n steps when executed by n units in parallel.

5 Implementation

5.1 Parameter Design

This section describes a procedure to determine parameters r, n, ϵ, δ, α, and q, for a given modulus N to satisfy five Conditions in Theorem 3. First we assume N is 1024 bit number and all base elements a_i and b_i are 32 bit, i.e., $r = 32$. This requires $nr > 1024$ and thus $n \geq 33$.

Since $\epsilon = \mathrm{Max}(2^r - m_i)/2^r$, if a_i and b_i are taken sufficiently close to 2^r, ϵ can be small. Actually, by computer search, for $n = 33$, we can find a and b with $\epsilon < 2^{-22}$, which satisfy Conditions (1) and (2) in Theorem 3.

δ's upper bound is mainly restricted by q, namely, the precision of the adder in Cox unit. We can derive the following inequality (See Appendix C).

$$\delta \leq \frac{1}{2^q} \cdot \frac{1 - 2^{-(r-q)}}{1 - \epsilon} \simeq \frac{1}{2^q}$$

The last approximation is correct if $2^{-(r-q)} \ll 1$ and $\epsilon \ll 1$. On the other hand, Condition (3) $\Delta = n(\epsilon + \delta) \leq \alpha$ is rearranged to $\delta \leq \alpha/n - \epsilon$. Therefore, the following condition is sufficient for Δ to be less than α.

$$\frac{1}{2^q} < \frac{\alpha}{n}\left(1 - \frac{\epsilon n}{\alpha}\right)$$

If we choose $\alpha = 1/2$, $n = 33$, and $\epsilon < 2^{-22}$, the minimum acceptable value for q is 7. This means Cox unit should have a 7 bit adder to satisfy Condition (3) and the initial value α of its output register can be $1/2$.

Finally, by the definition of ϵ, $A, B \geq 2^{nr}(1-\epsilon)$ can be shown. Comparing this value with $4N/(1-\Delta)$ and $2N/(1-\alpha)$, it is shown that for $n = 33$, Conditions (4) $4N/(1-\Delta) \leq B$ and (5) $2N/(1-\alpha) \leq A$ are satisfied.

5.2 Performance

Table 1 summarizes number of operations necessary to estimate the modular exponentiation time. Columns (1), (2), and (3) of the table correspond to a Montgomery multiplication, an exponentiation, and other functions, respectively. Let L, f, and R denote the total number of operations, a frequency of operation, and a throughput of exponentiation, respectively. L is then roughly estimated by $L = (1) \times (2) + (3)$ and $R = f \cdot nr/(L/n)$. Here, L is divided by n because n processing units operate at a time. The throughput R is then approximated by

$$R \approx \frac{f}{3n + 27/2}.$$

For 1024-bit full exponentiation, R is about 890 [kbit/sec] if $r = 32$, $n = 33$, and $f = 100$MHz are chosen. According to [8], these are a reasonable choice for deep sub-micron CMOS technologies such as $0.35 - 0.18\ \mu m$. If a binary exponentiation is replaced by a 4-bit window method, R is improved to 1.1 [Mbps] with a penalty of approximately 4 kByte RAM increase. Table 2 shows the required memory size for a binary exponentiation.

Table 1. Number of Operations in algorithm EXP

	(1)		(2)	(3)		
	Alg. MM		Alg. EXP	Others		
	Alg. BE	Others	No. of MM	Radix-RNS	RNS-Radix	Correction
Operation	mod-mul	mod-mul	–	mod-mul	mod-mul	Subtraction
No. of Operations	$2n(n+2)$	$5n$	$\frac{3nr}{2}+2$	n^2	$2n^2+n$	n^2

Table 2. Memory Size ($r = 32, n = 33$)

	RAM (Byte)	ROM (Byte)
Symbol	nr	$nr(7n+11)/8$
Total	1k	32k
Per Rower Unit	32	970

6 Conclusion

A new RNS Montgomery multiplication algorithm has been presented. Our algorithm together with representation transformations can be implemented on the Cox-Rower Architecture proposed in this paper. The performance is roughly estimated and turns out to be quite high because of the inherent parallelism of RNS. This paper contains no explanation about the fact that a modular reduction operation $y = x \bmod m_i$ which is used in Equation (11) etc. can be relaxed to $y \equiv x (\bmod m_i)$ and $y < 2^r$. In this case as well, theorems similar to Theorem 1, 2, and 3 can be proven. The relaxed modular reduction will result in simpler hardware. In addition, for moduli $m_i = 2^r - \mu_i$, μ_i can be chosen so that the modular reduction is fast, and $\mu_i \ll 2^r$ is one such criteria. A VLSI design and a detailed performance estimation remains to be studied.

References

1. A. P. Shenoy, R. Kumaresan, *"Fast Base Extension Using a Redundant Modulus in RNS,"* IEEE Trans. on Computers, Vol.38, No.2, pp.292–297, Feb. 1989.
2. A. P. Shenoy, R. Kumaresan, *"Residue to Binary Conversion for RNS Arithmetic Using Only Modular Look-up Tables,"* IEEE Trans. on Circuit and Systems, Vol.35, No.9, pp.1158–1162, Sep. 1988.
3. M. A. Soderstrand, C. Vernia, Jui-Hua Chang *"An Improved Residue Number System Digital-to-Analog Converter,"* IEEE Trans. on Circuit and Systems, Vol.30, No.12, pp.903–907, Dec. 1983.
4. C. H. Huang, *"A Fully Parallel Mixed-Radix Conversion Algorithm for Residue Number Applications,"* IEEE Trans. on Computers, Vol.32, No.4, pp.398–402, April, 1983.

5. J.-J. Quisquater, C. Couvreur, *"Fast Decipherment Algorithm for RSA Public-Key Cryptosystem,"* Electronics Letters, Vol.18, pp.905–907, Oct., 1982.
6. R. Rivest, A. Shamir, L. Adleman, *"A Method for Obtaining Digital Signatures and Public Key Cryptosystems,"* Communications of the ACM, Vol.21, No.2, pp.120–126, Feb., 1978.
7. K. C. Posch, R. Posch, *"Modulo Reduction in Residue Number Systems,"* IEEE Trans. on Parallel and Distributed Systems, Vol.6, No.5, pp.449–454, May 1995.
8. J. Schwemmlein, K. C. Posch, R. Posch, *"RNS-Modulo Reduction Upon a Restricted Base Value Set and its Applicability to RSA Cryptography,"* Computer & Security, Vol.17, No.7, pp.637–650, 1998.
9. Jean-Claud Bajard, Laurent-Stephane Didier, Peter Kornerup, *"An RNS Montgomery Multiplication Algorithm,"* Proceedings of ARITH13, IEEE Computer Society, pp.234–239, July 1997.
10. Jean-Claud Bajard, Laurent-Stephane Didier, Peter Kornerup, *"An RNS Montgomery Multiplication Algorithm,"* IEEE Trans. on Computers, Vol.47, No.7, pp.766–776, 1998.
11. Pascal Paillier, *"Low-Cost Double-Size Modular Exponentiation or How to Stretch Your Cryptoprocessor,"* Proc. of PKC'99, pp.223–234, 1999.
12. D. E. Knuth, *The Art of Computer Programming,* Vol.2, Seminumerical Algorithms, Second Edition, pp.268–276, Addison-Wesley, 1981.
13. P. L. Montgomery, *"Modular Multiplication without Trial Division,"* Mathematics of Computation, Vol.44, No.170, pp.519–521, April, 1985.

A Proof of Theorem 1 and 2

From Equation (8), $\delta_{m_i} = (\xi_i - \mathrm{trunc}(\xi_i))/m_i$. This leads to $\mathrm{trunc}(\xi_i) = \xi_i - m_i\delta_{m_i}$. Similarly, since $\epsilon_{m_i} = (2^r - m_i)/2^r$, $2^r = m_i/(1-\epsilon_{m_i})$ holds. Taking these into account, the following equation can be derived.

$$\sum_{i=1}^{n} \frac{\mathrm{trunc}(\xi_i)}{2^r} = \sum_{i=1}^{n} \frac{(\xi_i - m_i\delta_{m_i})(1-\epsilon_{m_i})}{m_i} = \sum_{i=1}^{n} \frac{(1-\epsilon_{m_i})\xi_i}{m_i} - \sum_{i=1}^{n}(1-\epsilon_{m_i})\delta_{m_i}$$
$$\geq (1-\epsilon_m)\sum_{i=1}^{n}\frac{\xi_i}{m_i} - n\delta_m > \sum_{i=1}^{n}\frac{\xi_i}{m_i} - n(\epsilon_m + \delta_m)$$

Apparently,

$$\sum_{i=1}^{n} \frac{\mathrm{trunc}(\xi_i)}{2^r} \leq \sum_{i=1}^{n} \frac{\xi_i}{m_i}.$$

Now it follows that

$$\sum_{i=1}^{n} \frac{\xi_i}{m_i} - n(\epsilon_m + \delta_m) < \sum_{i=1}^{n} \frac{\mathrm{trunc}(\xi_i)}{2^r} \leq \sum_{i=1}^{n} \frac{\xi_i}{m_i}.$$

By adding α on each sides and substituting Equation (5), the following equation is obtained.

$$\left(k + \frac{x}{M}\right) - n(\epsilon_m + \delta_m) + \alpha < \sum_{i=1}^{n} \frac{\mathrm{trunc}(\xi_i)}{2^r} + \alpha \leq \left(k + \frac{x}{M}\right) + \alpha \qquad (13)$$

Case 1: If $0 \leq n(\epsilon_m + \delta_m) \leq \alpha < 1$ and $0 \leq x < (1-\alpha)M$: Equation (13) leads to

$$k < \sum_{i=1}^{n} \frac{\mathrm{trunc}(\xi_i)}{2^r} + \alpha < k+1.$$

Therefore,

$$\widehat{k} = \left\lfloor \sum_{i=1}^{n} \frac{\mathrm{trunc}(\xi_i)}{2^r} + \alpha \right\rfloor = k$$

holds. This proves Theorem 1.

Case 2: If $\alpha = 0$, $0 \leq n(\epsilon_m + \delta_m) < 1$, and $0 \leq x < M$: From Equation (13)

$$k - 1 < \sum_{i=1}^{n} \frac{\mathrm{trunc}(\xi_i)}{2^r} < k+1.$$

Then,

$$\widehat{k} = \left\lfloor \sum_{i=1}^{n} \frac{\mathrm{trunc}(\xi_i)}{2^r} \right\rfloor = k \text{ or } k-1.$$

It is easy to see that, if $x/M - n(\epsilon_m + \delta_m) \geq 0$, then $\widehat{k} = k$. Contraposition leads to that if $\widehat{k} = k-1$, then $x/M - n(\epsilon_m + \delta_m) < 0$. Therefore, if $\widehat{k} = k-1$,

$$z = \sum_{i=1}^{n} \xi_i M_i - \widehat{k}M = x + M < \{n(\epsilon_m + \delta_m) + 1\}M.$$

Of course, if $\widehat{k} = k$, then $z = x$ and $z < M$. This proves Theorem 2.

B Proof of Theorem 3

The following requirements should be considered:

- Both $N^{-1} \bmod B$ and $B^{-1} \bmod A$ exists,
- All intermediate values are less than AB,
- For inputs less than $2N$, the algorithm outputs w which is less than $2N$,
- Base extension error at step 4 does not cause any trouble,
- w is computed correctly at step 7 and base extension at step 8 is error-free.

First requirement is satisfied by Conditions (1) and (2) in Theorem 3.

Here we define $\widehat{t}$ as a result of base extension at step 4. We also define the correct value as $t = s(-N^{-1}) \bmod B$. Due to Theorem 2, $\widehat{t} = t$ or $t + B$, and

$$\widehat{t} < \{1 + n(\delta_b + \epsilon_b)\}B \leq (1 + \Delta)B.$$

With this inequality, the largest intermediate value v is evaluated as follows:

$$\begin{aligned} v &= xy + \widehat{t}N < 4N^2 + (1+\Delta)BN \\ &\leq (1-\Delta)BN + (1+\Delta)BN \quad \text{(by Condition (4))} \\ &= 2BN \\ &\leq (1-\alpha)AB \quad \text{(by Condition (5))} \\ &\leq AB \quad \text{(by Condition (3))}. \end{aligned} \tag{14}$$

This satisfies the second requirement above. Further, by dividing each term of Equation (14) by B, we also obtain $w = v/B < 2N \leq (1-\alpha)A$. Thus, third requirement above is satisfied and the value $< w >_a$ is extended to $< w >_{a \cup b}$ without error if α is chosen according to Condition (3) and w is an integer.

We still have to confirm that v computed with $\widehat{t}$ is a multiple of B, and whether w is correctly computed by $< v >_a < B^{-1} >_a$. Since v is either $xy + tN$ or $xy + (t+B)N$ and $xy + tN \equiv 0 \pmod{B}$, we obtain $v \equiv 0 \pmod{B}$. So v is a multiple of B and $w = v/B$ is an integer, which is less than A. Taking these into account, w can be computed by $< v >_a < B^{-1} >_a$, because $vB^{-1} \bmod A = (wB)B^{-1} \bmod A = w \bmod A = w$ (last equation is due to $w < A$). On the other hand,

$$w = \frac{v}{B} = \frac{xy + tN}{B} \text{ or } \frac{xy + (t+B)N}{B}. \tag{15}$$

In both cases, it is easy to confirm $w \equiv xyB^{-1} (\mathrm{mod} N)$. This proves Theorem 3.

C δ's upper Bound

From Equation (8), $\delta_{m_i} = (\xi_i - \mathrm{trunc}(\xi_i))/m_i$. So,

$$\delta = \mathrm{Max}\left(\frac{\xi_i - \mathrm{trunc}(\xi_i)}{m_i}\right) \leq \frac{\mathrm{Max}(\xi_i - \mathrm{trunc}(\xi_i))}{\mathrm{Min}(m_i)}. \tag{16}$$

On the other hand,

$$\epsilon = \mathrm{Max}\left(\frac{2^r - m_i}{2^r}\right) = \frac{2^r - \mathrm{Min}(m_i)}{2^r}.$$

This leads to

$$\mathrm{Min}(m_i) = 2^r(1-\epsilon). \tag{17}$$

Also,

$$\xi_i - \mathrm{trunc}(\xi_i) = \xi_i - \xi_i \bigwedge (\overbrace{1 \ldots 1}^{q} \overbrace{0 \ldots 0}^{(r-q)})_{(2)} \leq (\overbrace{1 \ldots 1}^{r-q})_{(2)} = 2^{r-q} - 1. \tag{18}$$

Substituting (17) and (18) to (16) results in

$$\delta \leq \frac{2^{r-q} - 1}{2^r(1-\epsilon)} = \frac{1}{2^q} \cdot \frac{1 - 2^{-(r-q)}}{1-\epsilon}.$$

Efficient Receipt-Free Voting Based on Homomorphic Encryption

Martin Hirt[1]* and Kazue Sako[2]

[1] ETH Zurich, Switzerland, hirt@inf.ethz.ch
[2] NEC Corporation, Japan, sako@ccm.cl.nec.co.jp

Abstract. Voting schemes that provide *receipt-freeness* prevent voters from proving their cast vote, and hence thwart vote-buying and coercion. We analyze the security of the multi-authority voting protocol of Benaloh and Tuinstra and demonstrate that this protocol is *not* receipt-free, opposed to what was claimed in the paper and was believed before. Furthermore, we propose the first practicable receipt-free voting scheme. Its only physical assumption is the existence of secret *one-way* communication channels from the authorities to the voters, and due to the public verifiability of the tally, voters only join a single stage of the protocol, realizing the "vote-and-go" concept. The protocol combines the advantages of the receipt-free protocol of Sako and Kilian and of the very efficient protocol of Cramer, Gennaro, and Schoenmakers, with help of designated-verifier proofs of Jakobsson, Sako, and Impagliazzo.
Compared to the receipt-free protocol of Sako and Kilian for security parameter ℓ (the number of repetitions in the non-interactive cut-and-choose proofs), the protocol described in this paper realizes an improvement of the total bit complexity by a factor ℓ.

1 Introduction

1.1 Background

Secret-ballot voting protocols are one of the most significant application of cryptographic protocols. The most efficient secret-ballot voting protocols can be categorized by their approaches into three types: Schemes using mix-nets [Cha81,PIK93,SK95,OKST97,Jak98,Abe99], schemes using homomorphic encryption [CF85,CY86,Ben87,BT94,SK94,CFSY96,CGS97], and schemes using blind signatures [FOO92,Sak94,Oka97]. The suitability of each of these three types varies with the conditions under which it is to be applied.

In a model with vote-buyers (or coercers), a voting scheme must ensure not only that a voter *can* keep his vote private, but also that he *must* keep it private. In other words, the voter should not be able to prove to a third party that he has cast a particular vote. He must neither obtain nor be able to construct a receipt proving the content of his vote. This property is referred to as *receipt-freeness*.

* Supported by the Swiss National Science Foundation, project no. SPP 5003-045293.

The concept of receipt-freeness was first introduced by Benaloh and Tuinstra [BT94]. Based on the assumption of a voting booth that physically guarantees secret communication between the authorities and each voter, they proposed two voting protocols using homomorphic encryptions. The first one is a single-authority voting protocol which, while being receipt-free, fails to maintain vote secrecy. Then they extend this protocol to the second protocol, which is a multi-authority scheme achieving vote secrecy. However, we show that this scheme is *not* receipt-free, as opposed to what is claimed in the paper.

Another receipt-free voting protocol based on a mix-net channel was proposed by Sako and Kilian [SK95]. In contrast to [BT94], it assumes only *one-way* secret communication from the authorities to the voters. The heavy processing load required for tallying in mix-net schemes, however, is a significant disadvantage of this protocol.

Finally, a receipt-free voting scheme using blind signatures was given by Okamoto [Oka97]. Here, the assumption was of *anonymous* one-way secret communication from each voter to each authority. Achieving communication that is both secret *and* anonymous would, however, be extremely difficult. Also, this scheme requires each voter to be active in *three rounds* (authorization stage, voting stage, and claiming stage), which is not acceptable in practice.

Another stream of research which relates to receipt-freeness is incoercible multi-party computation. Without any physical assumption, deniable encryption [CDNO97] allows an entity to lie *later* how the ciphertext decrypts, and this technique is used to achieve incoercible multi-party computation [CG96]. However, the concept of incoercibility is weaker than receipt-freeness. It would allow a voter to lie about his vote, but it cannot help against a voter who *wants* to make his encryption undeniable, and hence cannot prevent vote-buying.

1.2 Contributions

In this paper, we first demonstrate that the multi-authority protocol of Benaloh and Tuinstra [BT94] is not receipt-free, opposed to what was claimed in the paper and was believed before. We then present a novel generic construction for introducing receipt-freeness into a voting scheme based on homomorphic encryption by assuming some additional properties of the encryption function. This construction also includes a solution for the case that an authority does not send correct information through the untappable channel.[1] Moreover, as opposed to previous receipt-free protocols, we disable vote-buying even in cases where some authorities are colluding with the voter-buyer. The security of these protocols is specified with respect to a threshold t, where the correctness of the tally is guaranteed as long as at least t authorities remain honest during the whole protocol execution, and privacy is guaranteed as long as no t or more curious authorities pool their information.

[1] Due to the untappability of the channel the voter cannot prove that the received information is incorrect. In previous protocols, this problem was ignored, and the situation of a voter complaining about an authority would have lead to a deadlock.

Our construction gives a receipt-free voting protocol which runs as follows: For each voter the authorities jointly generate a randomly ordered list with an encryption of each valid vote, along the lines of [SK95]. The ordering of the list is secretly conveyed and proven to the voter by deploying the technique of designated-verifier proofs [JSI96], and the voter points to the encryption of his choice. Tallying of votes is performed using the homomorphic property of the encryption function.

By applying this generic construction to the voting protocol of Cramer, Gennaro, and Schoenmakers [CGS97], we obtain an efficient receipt-free voting protocol based on homomorphic encryption.

The efficiency achieved by our protocol compared to the protocol of Sako and Kilian [SK95] with security parameter ℓ in the case of 1-out-of-2 voting is as follows: The communication through the untappable channels and through the public channels are reduced by a factor of $\ell/4$ and $3\ell/2$, respectively. Altogether, this results in a speedup by a factor of ℓ. As an example, for $M = 1,000,000$ voters, $N = 10$ authorities, a $K = 1024$ bit group, and security parameter $\ell = 80$, the protocol of [SK95] communicates 102 GB (gigabyte) over the untappable channels and 924 GB over the public channels, whereas the protocol of this paper communicates 5 GB over untappable channels and 8 GB over public channels.

1.3 Organization of the Paper

The paper is organized as follows: In Sect. 2, we analyze the receipt-freeness of the protocol with multiple voting authorities of Benaloh and Tuinstra [BT94] and demonstrate its non receipt-freeness by showing how a voter can construct a receipt for the vote he casts. In Sect. 3, we present a generic receipt-free protocol for 1-out-of-L voting based on homomorphic encryptions, and in Sect. 4 we apply these techniques to the protocol of Cramer, Gennaro, and Schoenmakers [CGS97] and obtain an efficient receipt-free voting scheme. Finally, in Sect. 5, we even improve the efficiency of our protocol by tailoring it to 1-out-of-2 voting.

2 Analysis of the Benaloh-Tuinstra Protocol

The notion of receipt-freeness was first introduced by Benaloh and Tuinstra in [BT94]. They present two protocols that are claimed to be receipt-free. In the single-authority protocol, the authority learns how each vote was cast. This is of course far from satisfactory. In this section, we analyze the receipt-freeness of their protocol with multiple voting authorities and show how a voter can construct a receipt for the vote he casts.

2.1 Key Ideas of the Protocol

The basic idea of the multiple-authority protocol [BT94] is to have every voter secret-share his vote among the authorities (using Shamir's secret-sharing scheme [Sha79]), who then add up the shares and interpolate the tally. This idea works

due to the linearity of the secret-sharing scheme. There are two major tasks to solve: First, the voter must send one share to each authority in a receipt-free manner, and second, the voter must prove that the secret (the vote) is valid.

We concentrate on the second task: In order to secret-share the vote, the voter selects a random polynomial P of appropriate degree, such that $P(0) \in \{0,1\}$ is his vote. The share for the j-th authority is hence $P(j)$. Clearly, it is inherently important that the vote is valid, i.e. $P(0) \in \{0,1\}$, since otherwise the tally will be incorrect. Hence, the voter must provide a proof of validity for the cast vote.

For the sake of the proof of validity, the voter wishing to cast a vote v_0 submits a bunch of $n+1$ vote pairs, where n is a security parameter. That is, the voter submits the votes $(v_0, v_0'), \ldots, (v_n, v_n')$, and each pair (v_i, v_i') of votes must contain one 0-vote and one 1-vote in random order. For each pair (v_i, v_i') but the first, a coin is tossed and the voter is either asked to open the pair and show that indeed there is a 0-vote and a 1-vote, or he is asked to prove that either $v_i = v_0$ and $v_i' = v_0'$ is satisfied, or that $v_i = v_0'$ and $v_i' = v_0$ is satisfied. If the voter passes these tests, then with probability at least $1 - 2^{-n}$, v_0 is valid and is accepted as the voters vote.

2.2 How to Construct a Receipt

This cut-and-choose proof of validity offers an easy ability to prove a particular vote: In advance, the voter commits to the ordering of each pair of votes (i.e. he commits to the bit string $v_0, \ldots, v_n$). In each round of the cut-and-choose proof, one can verify whether the revealed data is consistent with this commitment. If no inconsistencies are detected while proving the validity of the vote, then with probability at least $1 - 2^{-n}$ the voter has chosen the ordering as committed, and also v_0 is as announced.

In order to obtain a receipt, the voter could select an arbitrary string s, and set the string $(v_0, \ldots, v_n)$ as the bitwise output of a known cryptographic hash function (e.g. MD5 or SHA) for that string s. Then, s is a receipt of the vote v_0.

3 Generic Receipt-Free Protocol

In this section, we present a novel and general construction for converting a voting protocol based on homomorphic encryption (with additional properties of the encryption function) into a receipt-free voting protocol. Receipt-freeness means that the voter cannot prove to a third party that he has cast a particular vote. The reason why most classical voting schemes are not receipt-free is simple: Each encrypted vote is published, and the voter himself can prove the content of his vote by revealing the randomness he used for encrypting. When a scheme requires the voter to choose randomness, then often the voter can exploit this to construct a receipt, for example by using the hash of a predetermined value (cf. Sect. 2). Therefore, in the protocol of this paper, the authorities jointly generate an encryption of each valid vote in random order, and each voter only *points* to the encrypted vote of his choice. The ordering of the encrypted valid

votes is proven to the voter in designated verifier manner through the untappable channel, so that the voter cannot transfer this proof to a vote-buyer.

3.1 Model and Definitions

ENTITIES. We consider a model with N authorities $A_1, \ldots, A_N$ and M voters. A threshold t denotes the lower bound on the number of authorities that is guaranteed to remain honest during the complete protocol execution.

COMMUNICATION. Communication takes place by means of a bulletin board which is publicly readable, and which every participant can write to (into his own section), but nobody can delete from. The bulletin board can be considered as public channels with memory. Furthermore, we assume the existence of untappable one-way channels from the authorities to the voters. The security of these channels must be physical, in such a way that even the voter cannot demonstrate what was sent over the channel (of course, the voter can record all received data, but he must not be able to *prove* to a third party that he received a particular string). Even a coercer who is physically present at the voter's place must not be able to eavesdrop the untappable channels. Note that some physical assumption seems to be inevitable for achieving receipt-freeness.[2] Indeed, untappable one-way channels from the authorities to the voters (as assumed in this paper and in [SK95]) are the weakest physical assumption for which receipt-free voting protocols are known to exist.

KEY INFRASTRUCTURE. To each voter, a secret key and a public key is associated, where it must be ensured that each voter knows the secret key according to his public key. This assumption is very natural and typically used for voter identification in any voting protocol. For the purpose of receipt-freeness, the knowledge of his own key is *essential*. If a voter can prove that he does not know his own secret key, then he can obtain a receipt (this holds for this protocol as well as for the protocol in [SK95]). We assume that the underlying public-key infrastructure guarantees that each voter knows his own key, but nevertheless we present a verification protocol (see Appendix A). Note that the receipt-free property is still achieved even if a voter discloses his secret key to the vote-buyer.

GENERALITY. The protocol is a 1-out-of-L voting scheme, where each entitled voter may submit one vote from the set $\mathcal{V}$ of valid votes, $|\mathcal{V}| = L$ (e.g. $L = 2$ and $\mathcal{V} = \{-1, 1\}$). The goal of the protocol is to securely compute the tally as the sum of the cast votes. Note that the restriction on a discrete set of valid votes is necessary in any receipt-free voting scheme. Voting schemes that allow the voter to cast an arbitrary string as his vote cannot ensure receipt-freeness, because they allow the voter to tag his vote.

[2] If the coercer is able to tap *all* communication channels between the voter and the authorities, then apparently the voter's private information (including secret key and randomness) is a receipt of the vote he has cast. The model for incoercible multi-party computation [CG96] does not assume physically secure channels, but participants who *want* to prove a certain behavior can do so in this setting, thus receipt-freeness is not achieved.

SECURITY. The security of the protocol comprises that the correctness of the computed tally is guaranteed as long as at least t authorities remain honest during the whole protocol execution (correctness); that any set of less than t authorities cannot decrypt any cast vote (privacy); and that a voter cannot prove to a third party which particular vote he has cast (receipt-freeness). In general, we assume that authorities do not collude with the vote-buyer, respectively the coercer (as assumed in all previous papers considering receipt-freeness). However, under certain circumstances some colluding can be tolerated. This is discussed in detail in Section 3.5.

3.2 Protocol Overview

The basic idea of the voting phase is illustrated in Fig. 1: First, each valid vote is encrypted in some deterministic way (e.g., by using the encryption function with "randomness" 0). This list of encrypted votes is publicly known (on the very left in the figure). Then, the first authority picks this list, shuffles it, and hands it to the next authority. To shuffle the list means to re-randomize each entry and to permute the order of the entries. In the figure, encryption is illustrated in terms of drawing a circle around the secret, and re-randomization is illustrated by rotating the secret. Then, the next authority picks the list, shuffles it, and so on. In addition to this shuffling, each authority must secretly reveal to the voter how the list was reordered yet in a *privately verifiable* manner through a secure untappable channel. This allows the voter to keep track of the ordering of the encrypted entries, and once each authority has shuffled the list, he can point to the encrypted vote of his choice. In order to prevent a voter who is colluding with an authority from casting an invalid vote, each authority must publicly prove that she shuffled correctly (without revealing the reordering, not shown in the figure). Votes cast this way are receipt-free: due to the *private* verifiability of how shuffling was performed, the voter has no way to convince a third party of the content of his vote. This property will be achieved by using designated-verifier proof technique [JSI96].

3.3 Requirements for the Basic Protocol

We assume a basic (non receipt-free) voting protocol based on homomorphic encryption for the stated model, and we require some extra properties of its encryption function. Let E be the (probabilistic) encryption function, and let $E(v)$ denote the set of encryptions for a vote v. An encryption e of vote v is one particular encryption of v, i.e. $e \in E(v)$. We require the following properties to be satisfied. The properties 1–3 are straightforward requirements for the encryption function of any voting scheme. The properties 4–6 are required exclusively in order to introduce the receipt-free property.

1. **encryption secrecy**
   For any group of less than t authorities it must be infeasible to decrypt any encryption e.

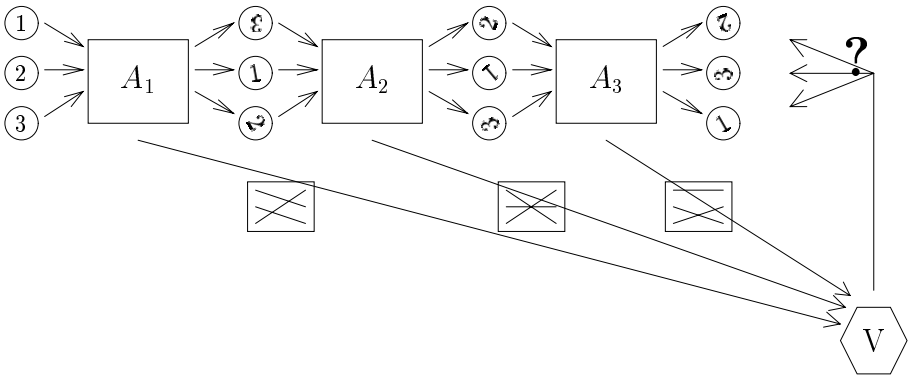


Fig. 1. Constructing a vote in $\mathcal{V} = \{1, 2, 3\}$ with 3 authorities.

2. **homomorphic property**
   We assume that the encryption function is homomorphic, that is, given the encryptions $e_1 \in E(v_1)$ and $e_2 \in E(v_2)$, the *addition* of these encryptions yields an encryption $e = e_1 \oplus e_2$ that encrypts the sum vote, i.e. $e \in E(v_1 + v_2)$. We require that this addition can be computed efficiently without any secrets.
3. **verifiable decryption**
   We require an efficient protocol for verifiably decrypting an encrypted sum vote, that is given any encryption $e \in E(T)$ the authorities can provide the sum of votes T and a proof that e indeed decrypts to T. This decryption and the proof must also work if up to $N - t$ authorities refuse cooperation or even misbehave maliciously. This protocol must not reveal any information that could weaken Property 1 (secrecy) of other encryptions.
4. **random re-encryptability**
   We require an algorithm for random re-encryption of any encryption e. Given $e \in E(v)$ (where typically v is unknown), there is a probabilistic re-encryption algorithm R that outputs $e' \in E(v)$, where e' is uniformly distributed over $E(v)$. We call the randomness used for generating e' the *witness*.
5. **existence of a 1-out-of-L re-encryption proof**
   Based on the random re-encryptability property, we assume the existence of an efficient protocol that given an encryption e, a list $e_1, \ldots, e_L$ of encryptions, and a witness that e_i is a re-encryption of e (for a given i), proves that indeed e_i is a re-encryption of e, without revealing i. This proof is called 1-out-of-L re-encryption proof.
6. **existence of a designated-verifier re-encryption proof**
   We assume the existence of an efficient protocol that given encryptions e and e' and a witness for e' being a re-encryption of e, proves the existence of such a witness in a manner that only the designated verifier can verify

its correctness [JSI96]. This proof is called designated-verifier re-encryption proof.

3.4 Introducing Receipt-Freeness

Given a voting protocol for the stated model which satisfies the requirements of the previous section, we can construct a receipt-free voting protocol. We first show how votes are generated (by the authorities) and how the voter casts his vote, then how tallying is performed.

VOTE GENERATION. Without loss of generality, assume that for each valid vote $v_i \in \mathcal{V}$, there exists a *standard encryption* $e_i^{(0)}$, where it is clear which v_i a given encryption $e_i^{(0)}$ belongs to.[3] Hence, $e_1^{(0)}, \ldots, e_L^{(0)}$ is a public list of all standard-encrypted valid votes.

In turn, for each authority A_k (where $k = 1, \ldots, N$):

1. A_k picks the list $e_1^{(k-1)}, \ldots, e_L^{(k-1)}$ of encrypted valid votes (for the first authority A_1, this is the public list of standard-encrypted valid votes, and for all succeeding authorities, this is the list of the previous authority). Then the authority shuffles this list randomly, and hands it to the next authority. To shuffle the list means to re-encrypt each encrypted vote (Property 4) and to permute the order of the list. More precisely, the authority randomly selects a permutation $\pi_k : \{1, \ldots, L\} \to \{1, \ldots, L\}$, computes a random re-encryption of $e_i^{(k-1)}$ and assigns it to $e_{\pi_k(i)}^{(k)}$ (for all $i = 1, \ldots, L$).
2. A_k publicly proves that she honestly shuffled, namely by proving for each i, there exists a re-encryption of $e_i^{(k-1)}$ in the list $e_1^{(k)}, \ldots, e_L^{(k)}$ without revealing which (1-out-of-L re-encryption proof, Property 5).
3. A_k secretly conveys to the voter the permutation π_k she used for reordering the encrypted votes and proves privately to him its correctness. More precisely, the permutation π_k and a designed-verifier proof for each $i = 1, \ldots, L$, that $e_{\pi_k(i)}^{(k)}$ is a re-encryption of $e_i^{(k-1)}$ (Property 6), is sent through the untappable channel to the voter.
4. If the voter does not accept the proof, he publicly complains about the authority. If the voter does so, then we set $e_1^{(k)} = e_1^{(k-1)}, \ldots, e_L^{(k)} = e_L^{(k-1)}$, i.e. the shuffling of this authority is ignored. The voter may complain against at most $N - t$ authorities.

CASTING A VOTE. The voter derives the position i of the encrypted vote $e_i^{(N)}$ of his choice, and publicly announces it.

TALLYING. The chosen encrypted votes of all voters are then summed for tallying. More precisely, they are added (using homomorphic addition $\oplus$, Property 2) to achieve an encryption $E(T)$ of the sum T of the votes. The authorities decrypt and output T and prove its correctness (Property 3).

[3] One technique to generate such encrypted votes is to use the probabilistic encryption algorithm E, and give as randomness the all-0 string. Such an encrypted vote $e_i^{(0)}$ can be decrypted by trying all valid votes $v \in \mathcal{V}$.

3.5 Security

CORRECTNESS. The correctness of the tally is guaranteed if all voters can cast the vote they wish (i.e. can trace the permutations of the authorities, Property 6), if they cannot cast invalid votes (Property 5), if the correct encrypted sum can be publicly computed (Property 2), and if the decryption of the sum is verifiable (Property 3).

PRIVACY. The privacy of each voter is guaranteed if an encrypted vote cannot be decrypted by an outstanding person or by any group of less than t authorities (Property 1). Also, given a list of encrypted votes and a shuffled list, it must be infeasible to find out which vote in the original list was permuted to which vote in the shuffled list (Property 4). Since at least t shufflings are performed correctly (at most $N-t$ shufflings can be skipped by a complaining voter), $t-1$ colluding authorities cannot find out the reordering of the list.

RECEIPT-FREENESS. The voter actively interacts at two points: First (in vote generation), the voter can disable the shuffling of up to $N-t$ authorities, and second (in vote casting), the voter points to the encrypted vote of his choice. Through the untappable channels, the voter receives the permutations π_k and the designated-verifier proofs for the correctness of each π_k. Due to the non-transferability of designated-verifier proofs (Property 6) and the untappability of the channels used he can lie for any of these permutations π_k, and this is sufficient for not being able to prove the cast vote. Note that although the proposed scheme is receipt-free, a coercer still can coerce a voter not to vote, or can coerce a voter to vote randomly.

In case that authorities collude with a vote-buyer or a coercer, then apparently receipt-freeness is still ensured as long as each voter knows at least one authority not colluding with the vote-buyer (then the voter can lie for the permutation π_k of this authority A_k). If a voter does not know such an authority, he can select one authority at random and lie for this permutation. In the context of vote-buying this means that the voter can forge a receipt for a vote he did not cast, and the vote-buyer accepts such a forged receipt with probability linear in the number of authorities not colluding with him, which seems to be unacceptable for the vote-buyer. However, in the context of coercion, this means that the probability of a lying voter to be caught is linear in the number of authorities colluding with the coercer, and this seems to be unacceptable for the voter.

4 [CGS97] Made Receipt-Free

In this section, we construct a receipt-free 1-out-of-L voting scheme based on the construction of Sect. 3 and on the protocol of Cramer, Gennaro, and Schoenmakers [CGS97].

4.1 Homomorphic ElGamal Encryption

The encryption scheme is exactly the same as used in [CGS97]. Here a very brief summary: The scheme is based on the ElGamal cryptosystem [E84]. Let

G be a commutative group of order $|G| = q$, where q is a large prime. G can be constructed as a subgroup of Z_p^*, where p is a large prime, but can also be obtained from elliptic curves. In the sequel, all operations are meant to be performed in G.

Let g be a generator of G, i.e. $G = \langle g \rangle$. The secret key z is chosen uniformly from Z_q, and the public key is $h = g^z$. The key pair (z, h) is constructed in a way that each authority receives a share z_i of z in a (t, N)-threshold secret-sharing scheme and is publicly committed to this share by $h_i = g^{z_i}$ [Ped91,CGS97]. Also, γ is another (independent) generator of G. The set $\mathcal{V}$ of valid votes contains L values in Z_q. An encryption of a vote $v \in \mathcal{V}$ is given by

$$E(v) = (g^\alpha, \gamma^v h^\alpha),$$

where $\alpha \in_R Z_q$ is a random number and γ^v is the "message" in the context of ElGamal.[4] We further let $e_v^{(0)} = (1, \gamma^v)$ be the standard encryption of v.

4.2 Encoding of Votes

There are several ways of encoding L votes in Z_q, such that the sum of several votes yields the sum of each type of vote. If for example $L = 2$, then one could set $\mathcal{V} = \{+1, -1\}$ and can derive how many 1-votes and how many (-1)-votes were cast from the sum and the number of cast votes.

For particular cases with $L > 2$, one can still use a similar approach. For example, if voters are allowed to cast "yes", "no", or "empty", and we are only interested in whether there are more "yes" or more "no" votes (disregarding the number of "empty" votes), one can use the encoding 1 for "yes", -1 for "no", and 0 for "empty".

However, if it must be possible to derive the exact number of cast votes for each choice, then more involved approaches are necessary. Along the ideas of [CFSY96], one can set $\mathcal{V} = \{1, M, M^2, \ldots, M^{L-1}\}$, where M denotes the number of voters. One can easily compute the number of cast votes for each choice, once the sum of the votes is computed.

We note that in any examples given in this subsection, decryption of the tally requires computing the discrete logarithm of γ^T, where T is the sum of all cast votes (as in [CGS97]). This can be done with complexity $O(\sqrt{M}^{L-1})$, see [CGS97] for more details.

4.3 Main Protocol

The main protocol is according to the generic protocol of Sect. 3. All we have to show is that the above encryption scheme satisfies the required properties of Sect. 3:

[4] The original ElGamal scheme is homomorphic with respect to multiplication. In order to achieve it to be homomorphic with respect to addition (Property 2), the message is chosen as γ^v. Multiplication of two messages corresponds to addition of the votes.

1. (secrecy) The secret key z is shared among the authorities such that any $t-1$ authorities cannot compute z. Violating the secrecy of the scheme would mean to either break ElGamal [E84] or the secret-sharing scheme [Ped91].
2. (homomorphic property) Addition of two encryptions $e_1 = (x_1, y_1)$ and $e_2 = (x_2, y_2)$ is defined as

$$e_1 \oplus e_2 = (x_1 x_2, y_1 y_2) \ .$$

   It is obvious that if $e_1 \in E(v_1)$ and $e_2 \in E(v_2)$, then $(e_1 \oplus e_2) \in E(v_1 + v_2)$.
3. (verifiable decryption) In order to decrypt T from $e = (x, y)$ the authorities first jointly compute, reveal and prove $\hat{x} = x^z$. This can be achieved by having every authority A_i compute $\hat{x}_i = x^{z_i}$, where z_i is A_i's share of the secret key z, and then compute $\hat{x}$ from $\hat{x}_i$. This is possible if at least t authorities reveal and prove $\hat{x}_i$. More details can be found in [Ped91,CGS97]. Once $\hat{x}$ is known, one can compute

$$\frac{y}{\hat{x}} = \frac{\gamma^T \cdot h^\alpha}{(g^\alpha)^z} = \gamma^T \ .$$

   Then, the authorities must find T. The computation complexity of this task is discussed in Sect. 4.2
4. (re-encryptability) The re-encryption $e' = (x', y')$ of an encrypted vote $e = (x, y)$ is given by

$$(x', y') = (g^\xi x, h^\xi y)$$

   for a random integer $\xi \in_R Z_q$. Clearly, if ξ is chosen uniformly in Z_q, then (x', y') is uniformly distributed. This ξ serves as a witness of re-encryption.
5. (1-out-of-L re-encryption proof) An efficient witness indistinguishable protocol with which an authority can prove that a re-encryption of a given encrypted vote e is contained in the list $e_1, \ldots, e_L$ will be given in Sect. 4.4.
6. (designated-verifier re-encryption proof) An efficient witness indistinguishable protocol with which an authority can prove privately that an encrypted vote e' is a re-encryption of e will be given in Sect. 4.5.

4.4 1-out-of-L Re-encryption Proof

We present a witness indistinguishable protocol with which a prover can prove that for an encrypted vote (x, y), there is a re-encryption in the L encrypted votes $(x_1, y_1), \ldots, (x_L, y_L)$ (1-out-of-L re-encryption proof). The protocol is based on techniques presented in [CDS94,CFSY96,CGS97]. For this protocol, assume that (x_t, y_t) is a re-encryption of (x, y), and the re-encryption randomness (the witness) is ξ, i.e. $(x_t, y_t) = (g^\xi x, h^\xi y)$.

1. The prover selects $d_1, \ldots, d_L$ and $r_1, \ldots, r_L$ at random, and computes

$$a_i = \left(\frac{x_i}{x}\right)^{d_i} \cdot g^{r_i} \quad \text{and} \quad b_i = \left(\frac{y_i}{y}\right)^{d_i} \cdot h^{r_i} \qquad (\text{for } i = 1, \ldots, L)$$

and sends it to the verifier. Note that these values commit the prover to d_i and r_i for all $i = 1, \ldots, L$ except for $i = t$. a_t and b_t only commit the prover to a value $w = \xi d_t + r_t$, since $a_t = g^{\xi d_t + r_t}$ and $b_t = h^{\xi d_t + r_t}$. This means that the prover still can change d_t and r_t *after* this round.

2. The verifier picks a random challenge $c \in_R Z_q$ and sends it to the prover.
3. The prover modifies d_t such that $c = d_1 + \ldots + d_L$, modifies r_t such that $w = \xi d_t + r_t$ (both mod q) and sends $d_1, \ldots, d_L$ and $r_1, \ldots, r_L$ (with d_t and r_t modified) to the verifier.
4. The verifier tests whether

$$c \stackrel{?}{=} d_1 + \ldots + d_L \pmod{q}$$
$$a_i \stackrel{?}{=} \left(\frac{x_i}{x}\right)^{d_i} \cdot g^{r_i} \quad (\text{for } i = 1, \ldots, L)$$
$$b_i \stackrel{?}{=} \left(\frac{y_i}{y}\right)^{d_i} \cdot h^{r_i} \quad (\text{for } i = 1, \ldots, L)$$

The proposed protocol is a 3-move witness-indistinguishable proof. Using the Fiat-Shamir-heuristic [FS86] the proof can be converted to be non-interactive. Using a technique of [CFSY96], we can even achieve a proof that only requires the prover to send $2L$ elements of G. Let $\mathcal{H}$ denote a cryptographic hash function, then

1. The prover computes a_i and b_i (for $i = 1, \ldots, L$) as in the interactive proof.
2. Then the prover computes the challenge $c = \mathcal{H}(E \| a_1 \| \ldots \| a_L \| b_1 \| \ldots \| b_L)$, where $a \| b$ is the concatenation of a and b, and $E = (x \| y \| x_1 \| x_2 \| \ldots \| x_L \| y_L)$ is the environment.
3. For this challenge, the prover computes d_i and r_i (for $i = 1, \ldots, L$). The proof is the $2L$-vector $(d_1, \ldots, d_L, r_1, \ldots, r_L)$.
4. A verifier examines whether

$$d_1 + \ldots + d_L \stackrel{?}{=} \mathcal{H}\left(E \left\| \left(\tfrac{x_1}{x}\right)^{d_1} g^{r_1} \right\| \ldots \left\| \left(\tfrac{x_L}{x}\right)^{d_L} g^{r_L} \right\| \left(\tfrac{y_1}{y}\right)^{d_1} h^{r_1} \right\| \ldots \left\| \left(\tfrac{y_L}{y}\right)^{d_L} h^{r_L}\right).$$

4.5 Designated-Verifier Re-encryption Proof

Each authority secretly conveys and proves to the voter how she reordered the list of encrypted votes. Therefore, for each $i = 1, \ldots, L$, the authority proves that $e^{(k)}_{\pi(i)}$ is a re-encryption of $e^{(k-1)}_i$. In the sequel, based on techniques from [JSI96], we show how the authority can privately prove that (x', y') is a re-encryption of (x, y), where ξ is the witness, i.e. $(x', y') = (g^\xi x, h^\xi y)$. The voter's secret key is denoted as z_v and the corresponding public key is given by $h_v = g^{z_v}$. This protocol relies on the voter's knowledge of his secret-key. If this property is not ensured by the underlying public-key infrastructure, a protocol for guaranteeing it must be employed (see Appendix A).

1. The prover selects d, w and r at random, computes

$$a = g^d \ , \ \ b = h^d \ , \text{ and } \ \ s = g^w h_v^r \ ,$$

   and sends it to the verifier. These values commit the prover to d, w and r. However, s is a chameleon commitment for w and r, and the verifier can use his knowledge of z_v to open s to arbitrary values w' and r' satisfying $w' + z_v r' = w + z_v r$.
2. The verifier picks a random challenge $c \in_R Z_q$ and sends it to the prover.
3. The prover computes $u = d + \xi(c + w)$ and sends w, r, u to the verifier.
4. The verifier tests whether

$$s \stackrel{?}{=} g^w h_v^r$$
$$g^u \stackrel{?}{=} \left(\frac{x'}{x}\right)^{c+w} \cdot a$$
$$h^u \stackrel{?}{=} \left(\frac{y'}{y}\right)^{c+w} \cdot b$$

This protocol can be made non-interactive using Fiat-Shamir-heuristic [FS86]:

1. The prover computes a, b and s as in the interactive proof.
2. Then the prover computes the challenge $c = \mathcal{H}(E \| a \| b \| s)$, where $a \| b \| s$ means the concatenation of a, b and s, and $E = (x \| y \| x' \| y')$ is the environment.
3. For this challenge, the prover computes u. The proof is the vector (c, w, r, u).[5]
4. A verifier tests whether

$$c \stackrel{?}{=} \mathcal{H}\left(E \left\| \frac{g^u}{\left(\frac{x'}{x}\right)^{c+w}} \right\| \frac{h^u}{\left(\frac{y'}{y}\right)^{c+w}} \right\| g^w h_v^r\Bigg) \ .$$

Now we show how that the verifier who knows the secret z_v such that $g^{z_v} = h_v$ can generate the above proof for any (x, y) and $(\tilde{x}, \tilde{y})$. The key is that the value s does *not* stick the verifier to w and r. The verifier selects α, β and $\tilde{u}$ at random, and computes

$$\tilde{c} = \mathcal{H}\left(\tilde{E} \left\| \frac{g^{\tilde{u}}}{\left(\frac{\tilde{x}}{x}\right)^{\alpha}} \right\| \frac{h^{\tilde{u}}}{\left(\frac{\tilde{y}}{y}\right)^{\alpha}} \right\| g^{\beta}\Bigg)$$
$$\tilde{w} = \alpha - \tilde{c} \pmod{q}$$
$$\tilde{r} = \frac{\beta - \tilde{w}}{z_v} \pmod{q}$$

and sets $(\tilde{c}, \tilde{w}, \tilde{r}, \tilde{u})$ as the proof. It is easy to see that this proof passes the above verification, i.e. for any $(\tilde{x}, \tilde{y})$, the voter can "prove" that it is a re-encryption of (x, y).

[5] We note that this construction is slightly more efficient than the one presented in [JSI96], where they require a 5-vector as proof.

4.6 Communication-Complexity Analysis

In this section we analyze the communication complexity of the 1-out-of-L voting scheme. We assume that there are N authorities and M active (participating) voters. Let K denote the number of bits that are used to store an element of the group G.

Both initialization of the ElGamal keys and revealment of the final result use constant (in M) many messages and are thus ignored. The relevant costs are related to shuffling and to the designated-verifier proofs. For each active voter, in turn every authority shuffles the list of encrypted votes, posts the new list to the bulletin board ($2L$ group elements), posts a proof for honest shuffling ($L \cdot 2L$ group elements), and secretly conveys and proves the reordering to the voter ($L \log_2 L$ bits for the permutation and $L \cdot 4$ group elements for the proofs). Finally, the voter posts the index of the encrypted vote of his choice to the bulletin board ($\log_2 L$ bits). In total, there are $2KLMN(L+1) + M \log_2 L$ bits posted to the bulletin board, and $LMN(4K + \log_2 L)$ bits transfered through the untappable channels.

When the protocol for ensuring that each voter knows his own secret key (cf. Appendix A) is considered as part of the protocol (and not as part of the public-key infrastructure), then the number of bits posted to the bulletin board is increased by $MK(N+t)$, and the number of bits transfered through the untappable channels is increased by MNK.

5 Efficient Receipt-Free 1-out-of-2 Voting

In this section we give a more efficient receipt-free protocol for 1-out-of-2 voting based on the scheme of [CGS97]. We take advantage of the encryption scheme that enables *flipping* of votes easily. That is, one can generate the opposite of an encrypted vote e without knowing the vote.

We define the set of valid votes $\mathcal{V} = \{-1, +1\}$ and we use the same encryption scheme as in Sect. 4. We define $e^{(0)} = (1, \gamma)$ be the standard encryption for the vote 1.

Consider an encrypted vote $(x, y) \in E(v)$ (where $v \in \mathcal{V}$ is unknown). One can easily *flip* the vote by $\overline{e} = (x^{-1}, y^{-1})$ which yields $\overline{e} \in E(-v)$. In other words, for every encrypted vote, the encrypted opposite vote is implicitly defined. Following we give an efficient receipt-free protocol for 1-out-of-2 elections that makes extensive use of this flipping property.

In turn, for each authority A_k (where $k = 1, \ldots, N$):

1. A_k picks $e^{(k-1)}$, an encrypted 1-vote or (-1)-vote (for the first authority A_1, this is the standard encryption of the 1-vote, and for all succeeding authorities, this is the encrypted vote of the previous authority). Then the authority computes a random re-encryption of $e^{(k-1)}$ and either flips it or not, and assigns the result to $e^{(k)}$.

2. A_k publicly proves that she honestly re-encrypted (and optionally flipped), namely by proving that either $e^{(k)}$ or $\overline{e^{(k)}}$ is a re-encryption of $e^{(k-1)}$. Therefore, the proof of Sect. 4.4 is used, where $L = 2$.
3. A_k secretly conveys and proves privately to the voter whether she flipped or not. This proof will be the same designated-verifier proof as given in Sect. 4.5, where $L = 2$.
4. At most $N-t$ times the voter may not accept the proof and publicly complain about the authority. Then $e^{(k)} = e^{(k-1)}$.

Finally the voter casts his vote by announcing whether his vote is $e^{(N)}$ or $\overline{e^{(N)}}$. This encrypted vote is then summed for tallying.

The analysis of this scheme gives that totally $6KMN+M$ bits are posted to the bulletin board, and $MN(4K+1)$ bits are sent over the untappable channels (both quantities are almost half of the costs with the protocol of Sect. 4, where $L = 2$).

A careful analysis of the receipt-free voting scheme of Sako and Kilian [SK95] for security parameter ℓ (the number of rounds in the non-interactive cut-and-choose proofs) reveals the complexity of that scheme: There are in total $(9K + \log_2 M)MN\ell$ bits sent over the public channels and $KMN\ell$ bits over the untappable channels. This is more than $3\ell/2$ times more on the public channels and $\ell/4$ times more on the untappable channels than the scheme of this paper. The costs of the protocol from Appendix A must be added to all quantities if required.

6 Concluding Remarks

We have presented a generic construction of a receipt-free protocol from a given basic voting scheme. By applying this generic construction to the voting protocol of [CGS97] we obtain an efficient receipt-free 1-out-of-L voting protocol, and by tailoring it to 1-out-of-2 voting this results in a protocol which is ℓ times more efficient than the protocol of [SK95] with security parameter ℓ. Due to the protocol failure in [BT94], the constructions in this paper give the first receipt-free voting scheme based on homomorphic encryptions.

7 Acknowledgments

Special thanks go to Herbert Schnider for his cooperation on analyzing the security of [BT94]. Furthermore, we would like to thank Daniel Bleichenbacher, Ronald Cramer, Rosario Gennaro, Markus Jakobsson, Ueli Maurer, Kouichi Sakurai, Berry Schoenmakers, and Stefan Wolf for many interesting discussions. Also, we would like to thank the anonymous referees for their useful comments on the paper.

References

[Abe99] Masayuki Abe. Mix-networks on permutation networks In *Advances in Cryptology — ASIACRYPT '99*, vol. 1716 of *LNCS*, pp. 258–273. Springer-Verlag, 1999.

[Ben87] Josh Cohen Benaloh: *Verifiable Secret-Ballot Elections.* Yale University PhD thesis, YALEU/DCS/TR-561, 1987.

[BT94] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th ACM Symposium on the Theory of Computing (STOC)*, pp. 544–553. ACM, 1994.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *Advances in Cryptology — CRYPTO '97*, vol. 1294 of *LNCS*, pp. 90–104. Springer-Verlag, 1997.

[CG96] Ran Canetti and Rosario Gennaro. Incoercible multiparty computation. In *Proc. 37th IEEE Symposium on the Foundations of Computer Science (FOCS)*, 1996.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94*, vol. 839 of *LNCS*. Springer-Verlag, 1994.

[CF85] Josh D. Cohen (Benaloh) and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proc. 26th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp. 372–382. IEEE, 1985.

[CY86] Josh Cohen (Benaloh) and Moti Yung: Distributing the Power of a Government to Enhance the Privacy of Voters. In *Proc. 5th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 52–62. ACM, 1986.

[CFSY96] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Advances in Cryptology — EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 72–83. Springer-Verlag, May 1996.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications*, 8:481–489, 1997. Preliminary version in *Advances in Cryptology — EUROCRYPT '97*.

[E84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — CRYPTO '84*, vol. 196 of *LNCS*, pp. 10–18. Springer-Verlag, 1984.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. 28th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp. 427–437, 1987.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO '86*, vol. 263 of *LNCS*, pp. 186–194. Springer-Verlag, 1986.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology — AUSCRYPT '92*, pp. 244–251, 1992.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated-verifier proofs and their applications. In *Advances in Cryptology — EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 143–154. Springer-Verlag, 1996.

[Jak98] Markus Jakobsson. A Practical Mix. In *Advances in Cryptology — EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 448–461, Springer-Verlag, 1998.

[MH96] Markus Michels and Patrick Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In *Advances in Cryptology — ASIACRYPT '96*, vol. 1163 of *LNCS*, pp. 125–132. Springer-Verlag, 1996.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. of Workshop on Security Protocols '97*, vol. 1361 of *LNCS*, pp. 25–35. Springer-Verlag, 1997.

[OKST97] Wakaha Ogata, Kaoru Kurosawa, Kazue Sako and Kazunori Takatani Fault Tolerant Anonymous Channel In *Information and Communications Security ICICS '97*, vol. 1334 of *LNCS*, pp. 440–444. Springer-Verlag, 1997.

[PIK93] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology — EUROCRYPT '93*, vol. 765 of *LNCS*, pp. 248–259. Springer-Verlag, 1993.

[Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *Advances in Cryptology — EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 522–526. Springer-Verlag, 1991.

[Sak94] Kazue Sako. Electronic voting schemes allowing open objection to the tally. In *Transactions of IEICE*, vol. E77-A No.1, Jan. 1994.

[SK94] Kazue Sako and Joe Kilian. Secure voting using partially compatible homomorphisms. In *Advances in Cryptology — CRYPTO '94*, vol. 839 of *LNCS*, pp. 411–424. Springer-Verlag, 1994.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme – A practical solution to the implementation of a voting booth. In *Advances in Cryptology — EUROCRYPT '95*, vol. 921 of *LNCS*, pp. 393–403. Springer-Verlag, 1995.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

A Ensuring Knowledge of the Secret-Key

In a model providing receipt-freeness, it is essential that each voter knows his own secret-key. We assume that this verification is part of the underlying public-key infrastructure, but nevertheless we provide a protocol that ensures a voter's knowledge of his secret-key. This protocol may be performed as part of the key registration (in the public-key infrastructure), or as part of the voting protocol if the key infrastructure does not provide this property. This protocol requires a secure one-way untappable channel as used in the vote generation phase.

The following protocol is based on Feldman's secret-sharing scheme [Fel87]. It establishes that a voter v knows the secret key z_v corresponding to his public key h_v (where $g^{z_v} = h_v$):

- The voter shares his secret key z_v among the authorities by using Feldman's secret-sharing scheme [Fel87]: The voter v chooses a uniformly distributed random polynomial $f_v(x) = z_v + a_1 x + \ldots + a_{t-1} x^{t-1}$ of degree $t-1$, and

secretly sends[6] the share $s_i = f_v(i)$ to authority A_i (for $i = 1, \ldots, N$). Furthermore, the voter commits to the coefficient of the polynomial by sending $c_i = g^{a_i}$ for $i = 1, \ldots, t-1$ to the bulletin board.

- Each authority A_i verifies with the following equation whether the received share s_i indeed lies on the committed polynomial $f_v(\cdot)$:

$$g^{s_i} \stackrel{?}{=} h_v \cdot c_1^i \cdot \ldots \cdot c_{t-1}^{i^{t-1}} \qquad \left(= g^{z_v} \cdot g^{a_1 i} \cdot \cdots \cdot \cdots \cdot g^{a_{t-1} i^{t-1}} = g^{f_v(i)} \right) .$$

  If an authority detects an error, she complains and the voter is requested to post her share to the bulletin board. If the posted share does not correspond to the commitments, the voter is disqualified.
- Finally, every authority (which did not complain in the previous stage) sends her share through the untappable channel to the voter.

In the above protocol, clearly after the second step, either the (honest) authorities will have consistent shares of the voter's secret key z_v, or the voter will be disqualified. However, so far it is not ensured that the voter indeed knows the secret key, as the shares could have been provided by the coercer. In any case, in the final step the voter learns z_v. There are at least t honest authority who either complained (and thus their share is published), or who sent their share to the voter, and hence the voter can interpolate the secret key z_v.

[6] Either the voter encrypts the share with the authority's public-key, or alternatively the authority first sends a one-time pad through the untappable channel, and the voter then encrypts with this pad.

How to Break a Practical MIX and Design a New One

Yvo Desmedt[1,2*] and Kaoru Kurosawa[3]

[1] Department of Computer Science, Florida State University
PO Box 4530, 206 Love Building
Tallahassee, FL 32306-4530, USA
desmedt@cs.uwm.edu
[2] Dept. of Mathematics, Royal Holloway,
University of London, UK
[3] Dept. of Electrical and Electronic Engineering,
Faculty of Engineering, Tokyo Institute of Technology
2–12–1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan
kurosawa@ss.titech.ac.jp

Abstract. A MIX net takes a list of ciphertexts $(c_1, \cdots, c_N)$ and outputs a permuted list of the plaintexts $(m_1, \cdots, m_N)$ without revealing the relationship between $(c_1, \cdots, c_N)$ and $(m_1, \cdots, m_N)$. This paper first shows that the Jakobsson's MIX net of Eurocrypt'98, which was believed to be resilient and very efficient, is broken. We next propose an efficient t-resilient MIX net with $O(t^2)$ servers in which the cost of each MIX server is $O(N)$. Two new concepts are introduced, existential-honesty and limited-open-verification. They will be useful for distributed computation in general.

1 Introduction

1.1 Background

In his extensive work to achieve anonymity, Chaum introduced the concept of a MIX net [6]. MIX nets have found many applications in anonymous communication [6], election schemes [6, 11, 19, 24] and payment systems [14]. A MIX net takes from each user a ciphertext and outputs a permuted list of the plaintexts without revealing who has sent which plaintext, i.e., which plaintext corresponds to which ciphertext. This aspect of a MIX net is also known as *privacy*. Although Pfitzmann-Pfitzmann [21] showed an attack against the RSA implementation of Chaum's MIX scheme, the concept itself was not broken but it was refined. The original MIX net given by Chaum [6] satisfies *privacy* only under the condition that all the senders are honest. To address this issue, one needs *robustness*. A topic that was studied prior to robustness is *verifiability*, to allow to *detect* that

* A part of this research was done while the author visited the Tokyo Institute of Technology, March 4-19, 1999. He was then at the University of Wisconsin – Milwaukee. A part of his research was funded by NSF CCR-9508528.

the output of the MIX net is incorrect. If an outsider can verify this, the scheme is called *universally verifiable*.

Before surveying robustness and verifiability, another problem of Chaum's MIX net based on RSA should be pointed out, which is that the size of each ciphertext c_i is long, i.e., proportional to the total number of MIX servers. Park et al. overcame this problem by using the ElGamal encryption scheme so that the size of each c_i became independent of the number of MIX servers [19]. Almost all MIX nets proposed from then on are based on the ElGamal encryption scheme.

A general method to achieving verifiability is to have each MIX server prove that it behaved correctly in zero knowledge. Sako and Kilian [24] showed such an efficient proof system for Park et al.'s MIX net. The above MIX nets are, however, not *robust*. If at least one MIX server stops, then the entire system stops. Ogata et al. showed the first MIX net satisfying privacy, verifiability and robustness [18]. We call a scheme satisfying all these properties *resilient*.

For comparison, we focus on the random permutation stage of the MIX net because almost all known resilient MIX nets consist of two stages, a random permutation stage and a threshold decryption stage. Also, there are some cases where we want a permutation, but not a decryption. If a MIX net does a permutation only, it is possible that the MIX servers do not need to know the decryption key, which is an advantage.

Now in Ogata et al.'s MIX net, the computational cost of each MIX server is $O(\kappa tN)$, where N is the number of users, κ is the security parameter and t stands for the threshold number of untrusted MIX servers. Subsequently, Abe showed a more efficient resilient MIX net which is also universally verifiable in which the external verifier's cost is reduced to $O(\kappa N)$ [1].

At the same time, Jakobsson showed a very efficient resilient MIX net at Eurocrypt '98 [13] (but not universally verifiable). Later, he showed a more efficient MIX net at PODC'99 [15]. In these schemes, the computational cost of each MIX server is $O(tN)$.

Recently Abe [2, 3] showed his second resilient MIX net which is efficient for a small number of users. In this MIX net, the complexity is $O(tN \log N)$. Jakobsson and Juels showed a MIX net which has the same advantage [16]. In their MIX net, the cost of each MIX server is $O(tN \log^2 N)$. Since these complexities grow faster in N than the other schemes, these schemes suit small N.

1.2 Our Contribution

This paper first shows that the Jakobsson's first MIX net (presented at Eurocrypt'98) [13], which was believed to be resilient and very efficient, is not robust. We present an attack such that at least one malicious MIX server can prevent computing the correct output. We exploit a homomorphic property of Jakobsson's Eurocrypt '98 scheme to attack it. Observe that we make no claims about other MIX networks, such as the PODC'99 Jakobsson paper [15].

We also propose a new and very efficient resilient MIX net (but it is not universally verifiable). To obtain this scheme, we introduce three new concepts:

Work-sharing-MIX in which (significantly) more MIX servers are being used than one trusts. In threshold schemes a tradeoff is used between reliability and privacy. The motivation of work-sharing-MIX is to have a tradeoff between the number of MIX severs and the computational effort per MIX sever. When N is large (as in national elections), and one wants a sufficiently high security (i.e., a large t), then the computational effort of existing schemes may be prohibitive. We share the computational effort over several machines while maintaining the requirements as privacy, robustness and verifiability.

Existential-honesty divides the MIX servers into blocks of which we can guarantee that one is free of dishonest MIX servers, assuming the number of dishonest MIX servers is bounded by t.

Limited-open-verification is the opposite of zero-knowledge. To prove that a computation has been done correctly the party that did the computation in a block will open the secret it used. However, she will only open this to the members in the same block.

More details are given later on. Those concepts may be useful in other contexts such as secure distributed computation. We achieve 100% robustness in contrast with prior schemes (i.e. the probability of the failure of robustness is 0).

Although the total computational cost of our scheme is comparable to the one of Jakobsson's MIX net of PODC'99 [15] (i.e. $O(t^2N)$), the computational cost of each MIX server is significantly smaller (i.e. $O(N)$) in ours versus the one in Jakobsson's scheme ($O(tN)$). To achieve this we need $O(t^2)$ MIX servers rather than the usual $O(t)$. This introduces several open problems, which we discuss in Sect. 6.

Other details, such as the computational complexity assumptions we need to prove privacy are discussed later on.

2 Model of MIX Net

2.1 Model and Definitions

In the model of MIX nets, there exist three types of participants: *users*, a *bulletin board*, and the *MIX servers*.

1. The *users* post encrypted messages $(c_1, \cdots, c_N)$ to the *bulletin board*.
2. After the bulletin board fills up, or after some other triggering event occurs, the *mix servers* compute a randomly permuted list of decryptions $(m_1, \cdots, m_N)$ of all valid encryptions posted on the bulletin board.

MIX nets must satisfy privacy, verifiability and robustness. Suppose that at most t among v MIX servers and at most $N-2$ among N senders are malicious. Then we say that a MIX net satisfies :

- *t-privacy* if the relationship between $(c_1, \cdots, c_N)$ and $(m_1, \cdots, m_N)$ is kept secret.

- *t-verifiability* if an incorrect output of the MIX net is detected with overwhelming probability.
- *t-robustness* if it can output $(m_1, \cdots, m_N)$ correctly with overwhelming probability.

We say that a MIX net is *t-resilient* if it satisfies t-privacy, t-verifiability and t-robustness.

2.2 ElGamal Based Encryption Scheme for Users

ElGamal based encryption scheme was commonly used in some of the previous robust MIX nets [18, 1, 13]. Let p be a safe prime, i.e., p, q be primes such that $p = 2q + 1$, and g be a generator of G_q. Let $y = g^x \bmod p$, where x is a secret key. The public key is (p, q, g, y).

The MIX servers share a secret key x using a $(t+1, v)$ threshold scheme [27], where v denotes the number of MIX servers.

To encrypt a value $m \in G_q$, a random number $\gamma \in_u Z_q$ is chosen and the ciphertext $(a, b) = (g^\gamma, my^\gamma)$ is calculated. For decryption, $m = b/a^x$ is calculated by a threshold decryption scheme [DF89,20, 12]

As pointed out by Jakobsson, to guarantee that $m \in G_q$, we should let $m = (M \mid p)M$ for an original message $M \in [1 \ldots (p-1)/2]$, where $(M \mid p)$ is the Jacobi symbol of M.

2.3 Non-malleable ElGamal

Malicious users may post copies or correlated ciphertexts of some encrypted messages of honest users (repeated ciphertext attack). They can then determine (with some probability) what the decryption of the attacked message was, by counting repeats or correlations in the output list. Therefore, it is necessary to use a non-malleable encryption scheme. A public key cryptosystem is said to be non-malleable [9] if there exists no probabilistic polynomial time (p.p.t.) adversary such that given a challenge ciphertext c, he can output a different ciphertext c' such that the plaintexts m, m' for c, c' are meaningfully related. (For example, $m' = m + 1$.)

Tsiounis and Yung [28], and independently Jakobsson [13], showed a non-malleable ElGamal encryption scheme by combining Schnorr's signature scheme [25] with ElGamal encryption scheme under some cryptographic assumption in the random oracle model. Jakobsson used the non-malleable ElGamal encryption scheme in his MIX net for users' encryption to prevent the repeated ciphertext attack [13]. (For a detailed study of the security consult [26].) We also use this scheme in our MIX net of Sect. 4.

3 An Attack for Jakobsson's Practical MIX

In this section, we show how to break the Jakobsson's MIX net of Eurocrypt'98 [13], which was believed to be t-resilient and very efficient.

Jakobsson first showed that a MIX net is obtained by using MIXEXP which takes a list of items $\bar{\mu} = (c_1, \ldots, c_N)$ and robustly computes a permutation $(c_1^{\delta}, \ldots, c_N^{\delta})$. To avoid cut and choose methods, Jakobsson [13] developed a subprotocol in which each MIX server proves that the product of his input elements and the product of his output elements satisfy a certain relation. However, this does not imply proving that each MIX server behaved correctly even if the subprotocol is combined with his other subprotocols. We show an attack such that all the output elements of a MIX server can be affected in a proper way. We also exploit a homomorphic property of his scheme to attack it.

His MIX net is not robust if the MIXEXP is not robust. Therefore, the details of MIXEXP are given in Sect. 3.1. Our attack is given in Sect. 3.2.

If the reader is not interested or is already familiar with his scheme, he can go directly to Sect. 3.2.

3.1 Structure of the Scheme Attacked [13]

Let

$$\Delta_j = g^{\delta_j} \tag{1}$$

be the public information of a MIX server j, where δ_j is his secret. Define

$$\delta \stackrel{\triangle}{=} \prod_{j \in Q} \delta_j, \tag{2}$$

where Q denotes a quorum. MIXEXP takes a list of items $\bar{\mu} = (c_1, \ldots, c_N)$ and robustly computes a permutation $(c_1^{\delta}, \ldots, c_N^{\delta})$.

Jakobsson then showed an efficient implementation of MIXEXP. It consists of four protocols, Blinding I, Blinding II, Unblinding I and Unblinding II. For simplicity, let $Q = \{1, 2, \cdots, t+1\}$. For a list $\bar{\xi} = (d_1, \ldots, d_N)$ and $e \in Z_q$, define

$$\bar{\xi}^e \stackrel{\triangle}{=} (d_1^e, \ldots, d_N^e).$$

Let κ be a security parameter.

Blinding I: (see Fig. 1) For $1 \leq \lambda \leq \kappa$,

1. MIX server 1 chooses a random number $\rho_{I\lambda_1}$ and a random permutation $\pi_{I\lambda_1}$. He then computes

$$\pi_{I\lambda_1}(c_1^{\rho_{I\lambda_1}}, c_2^{\rho_{I\lambda_1}}, \cdots, c_N^{\rho_{I\lambda_1}}).$$

2. MIX server 2 chooses a random number $\rho_{I\lambda_2}$ and a random permutation $\pi_{I\lambda_2}$. He then computes

$$\pi_{I\lambda_2} \circ \pi_{I\lambda_1}(c_1^{\rho_{I\lambda_1}\rho_{I\lambda_2}}, c_2^{\rho_{I\lambda_1}\rho_{I\lambda_2}}, \cdots, c_N^{\rho_{I\lambda_1}\rho_{I\lambda_2}}),$$

and so on.

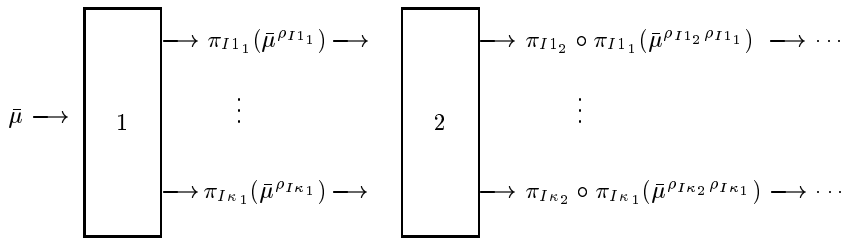


Fig. 1. Blinding I

The final output (in Blinding I) from MIX server $t+1$ is:

$$\bar{\mu}_{I\lambda} \stackrel{\triangle}{=} \pi_{I\lambda}(c_1^{\rho_{I\lambda}}, c_2^{\rho_{I\lambda}}, \cdots, c_N^{\rho_{I\lambda}}), \tag{3}$$

where

$$\pi_{I\lambda} \stackrel{\triangle}{=} \prod_{j \in Q} \pi_{I\lambda j}, \quad \rho_{I\lambda} \stackrel{\triangle}{=} \prod_{j \in Q} \rho_{I\lambda j}.$$

That is, MIXEXP outputs $\bar{\mu}_{I1}, \cdots, \bar{\mu}_{I\kappa}$ on input $\bar{\mu}$ in Blinding I.

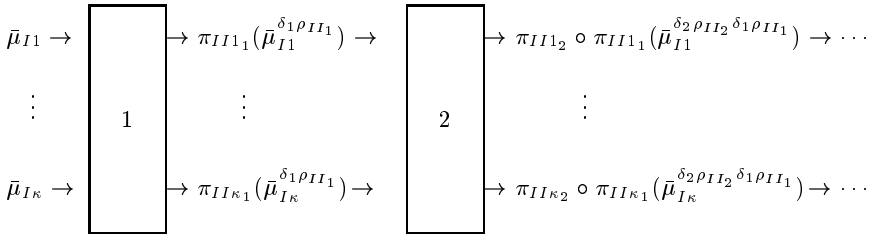


Fig. 2. Blinding II

Blinding II: (see Fig. 2) For $1 \leq \lambda \leq \kappa$,

1. MIX server 1 chooses a random number ρ_{II_1} and a random permutation $\pi_{II\lambda_1}$. He then computes

$$\pi_{II\lambda_1}(\bar{\mu}_{I\lambda}^{\delta_1\rho_{II1}}) = \pi_{II\lambda_1} \circ \pi_{I\lambda}((c_1^{\rho_{I\lambda}})^{\delta_1\rho_{II1}}, (c_2^{\rho_{I\lambda}})^{\delta_1\rho_{II1}},$$

$$\cdots, (c_N^{\rho_{I\lambda}})^{\delta_1\rho_{II1}})$$

from $\bar{\mu}_{I\lambda}$. Note that ρ_{II_1} is independent of λ while $\rho_{I\lambda_1}$ depends on λ.

2. MIX server 2 chooses a random number ρ_{II_2} and a random permutation $\pi_{II\lambda_2}$. He then computes

$$\pi_{II\lambda_2} \circ \pi_{II\lambda_1} \circ \pi_{I\lambda}((c_1^{\rho_{I\lambda}})^{\delta_1\rho_{II_1}\delta_2\rho_{II_2}}, (c_2^{\rho_{I\lambda}})^{\delta_1\rho_{II_1}\delta_2\rho_{II_2}},$$

$$\cdots, (c_N^{\rho_{I\lambda}})^{\delta_1\rho_{II_1}\delta_2\rho_{II_2}})$$

and so on.

The final output (in Blinding II) of MIX server $t+1$ is:

$$\bar{\sigma}_{II\lambda} \stackrel{\triangle}{=} \pi_{II\lambda} \circ \pi_{I\lambda}(c_1^{\rho_{I\lambda}\delta\rho_{II}}, c_2^{\rho_{I\lambda}\delta\rho_{II}}, \cdots, c_N^{\rho_{I\lambda}\delta\rho_{II}}), \tag{4}$$

where δ is defined in eq. (2) and

$$\pi_{II\lambda} \stackrel{\triangle}{=} \prod_{j\in Q} \pi_{II\lambda j}, \quad \rho_{II} \stackrel{\triangle}{=} \prod_{j\in Q} \rho_{IIj}.$$

That is, MIXEXP outputs $\bar{\sigma}_{II1}, \cdots, \bar{\sigma}_{II\kappa}$ on input $\bar{\mu}_{I1}, \cdots, \bar{\mu}_{I\kappa}$ in Blinding II.

From eq. (4), we see that

$$\bar{\sigma}_{II\lambda}^{1/\rho_{I\lambda}} = \pi_{II\lambda} \circ \pi_{I\lambda}(c_1^{\delta\rho_{II}}, \ldots, c_N^{\delta\rho_{II}}).$$

Note that $(c_1^{\delta\rho_{II}}, \ldots, c_N^{\delta\rho_{II}})$ of the right hand side is independent of λ. Therefore, $\bar{\sigma}_{II\lambda}^{1/\rho_{I\lambda}}$ must be equal for $1 \le \lambda \le \kappa$ if each list $\pi_{II\lambda} \circ \pi_{I\lambda}(c_1^{\delta\rho_{II}}, \ldots, c_N^{\delta\rho_{II}})$ is sorted. Unblinding I, based on this observation, is described as follows.

Unblinding I:

1. Each MIX server j publishes $\{\rho_{I\lambda j}\}$ for $1 \le \lambda \le \kappa$.
2. Each MIX server computes $\rho_{I\lambda} = \prod_{j\in Q} \rho_{I\lambda j}$ and

$$\bar{\sigma}_{I\lambda} \stackrel{\triangle}{=} \bar{\sigma}_{II\lambda}^{1/\rho_{I\lambda}} = \pi_{II\lambda} \circ \pi_{I\lambda}(c_1^{\delta\rho_{II}}, \ldots, c_N^{\delta\rho_{II}}) \tag{5}$$

   for $1 \le \lambda \le \kappa$.
3. The lists $\bar{\sigma}_{I\lambda}$ with $1 \le \lambda \le \kappa$ are sorted and compared. If they are all equal, and no element is zero, then the result is labeled valid, otherwise invalid.

Next in Blinding II for $\lambda = 1$, let M_j denote the product (modulo p) of all the elements constituting the input to MIX server j. Similarly, S_j denotes the product of all the output elements of MIX server j. Then it must hold that

$$S_j = M_j^{\delta_j\rho_{II_j}}. \tag{6}$$

On the other hand, from eq. (1), we have $\Delta_j^{\rho_{II_j}} = g^{\delta_j\rho_{II_j}}$. Therefore, it holds that

$$S_j = M_j^z \text{ and } \Delta_j^{\rho_{II_j}} = g^z$$

for $z = \delta_j\rho_{II_j}$. Unblinding II, based on this observation, is described as follows.

Unblinding II: (for valid results only)

1. The MIX servers publish $\{\pi_{I1j}\}_{j\in Q}$.
2. The computation of $\bar{\mu}_{I1}$ in "Blinding I" is verified.
3. The MIX servers publish $\{\rho_{IIj}\}_{j\in Q}$.
4. Each MIX server j proves that

$$S_j = M_j^z \text{ and } \Delta_j^{\rho_{II_j}} = g^z. \tag{7}$$

   holds for some z by using one of the methods of [7, 25].
5. The MIX servers compute $\bar{\sigma}_1 = \bar{\sigma}_{I1}^{1/\rho_{II}}$, and output $\bar{\sigma}_1$. Note that $\bar{\sigma}_1$ is a permutation of $(c_1^\delta, \ldots, c_N^\delta)$ from eq. (5).

Jakobsson claims that the final output $\bar{\sigma}_1$ is a permutation of $(c_1^\delta, \ldots, c_N^\delta)$ if the above protocol (MIXEXP) ends successfully.

3.2 Our Attack

We show that Jakobsson's MIXEXP is not robust. This means that his MIX net is not robust. Our attack succeeds if at least one MIX server is malicious.

We exploit a homomorphic property. A dishonest MIX server will first multiply the received inputs. The data is then organized to prevent detection. We now describe the details.

For simplicity, suppose that the last MIX server $t+1$ of Q is malicious. In Blinding II, let her input be $(d_{\lambda 1}, \ldots, d_{\lambda N})$ for $1 \le \lambda \le \kappa$. Let

$$X_\lambda \stackrel{\triangle}{=} d_{\lambda 1} * \cdots * d_{\lambda N}.$$

In our attack, she first chooses random numbers $\alpha_1, \cdots, \alpha_N$ such that

$$\alpha_1 + \alpha_2 + \cdots \alpha_N = 1 \bmod q.$$

Next she outputs

$$\tilde{\sigma}_{II\lambda} = (X_\lambda^{\alpha_1 \delta_{t+1} \rho_{II_{t+1}}}, X_\lambda^{\alpha_2 \delta_{t+1} \rho_{II_{t+1}}}, \ldots, X_\lambda^{\alpha_N \delta_{t+1} \rho_{II_{t+1}}}) \tag{8}$$

for $1 \le \lambda \le \kappa$.

We next show that the MIXEXP ends successfully and our cheating is not detected.

Theorem 1. *The check of Unblinding I is satisfied.*

Proof: In Blinding II, the output of MIX server $t+1$ is $\bar{\sigma}_{II\lambda}$ of eq. (4) if she is honest. Therefore, her input must be

$$\begin{aligned}(d_{\lambda_1}, \ldots, d_{\lambda_N}) &= \pi_{II\lambda_{t+1}}^{-1}(\bar{\sigma}_{II\lambda}^{1/\delta_{t+1}\rho_{II_{t+1}}}) \\ &= \theta_\lambda(c_1^{\rho_{I\lambda}\delta\rho_{II}/\delta_{t+1}\rho_{II_{t+1}}}, \ldots, c_N^{\rho_{I\lambda}\delta\rho_{II}/\delta_{t+1}\rho_{II_{t+1}}})\end{aligned}$$

for some permutation θ_λ for $1 \leq \lambda \leq \kappa$. Therefore,

$$X_\lambda = d_{\lambda_1} * \cdots * d_{\lambda_N} = (c_1 * \cdots * c_N)^{\rho_{I\lambda}\delta\rho_{II}/\delta_{t+1}\rho_{II_{t+1}}} = Z^{\rho_{I\lambda}\delta\rho_{II}/\delta_{t+1}\rho_{II_{t+1}}},$$

where $Z \stackrel{\triangle}{=} c_1 * \cdots * c_N$. Then eq. (8) is written as

$$\tilde{\sigma}_{II\lambda} = (Z^{\alpha_1\rho_{I\lambda}\delta\rho_{II}}, \ldots, Z^{\alpha_N\rho_{I\lambda}\delta\rho_{II}}).$$

Finally, at Step 2 of Unblinding I, each MIX server computes

$$\tilde{\sigma}_{I\lambda} = \tilde{\sigma}_{II\lambda}^{1/\rho_{I\lambda}} = (Z^{\alpha_1\delta\rho_{II}}, \ldots, Z^{\alpha_N\delta\rho_{II}}) \tag{9}$$

for $1 \leq \lambda \leq \kappa$. Note that $(Z^{\alpha_1\delta\rho_{II}}, \ldots, Z^{\alpha_N\delta\rho_{II}})$ is independent of λ. Therefore, we see that $\tilde{\sigma}_{I1} = \tilde{\sigma}_{I2} = \cdots = \tilde{\sigma}_{I\kappa}$. This means that the check of Unblinding I is satisfied. □

Theorem 2. *The check of Unblinding II is satisfied.*

Proof: Note that $M_{t+1} = X_1$ and S_{t+1} is the product of all the elements of eq. (8) for $\lambda = 1$. Therefore, we have

$$\begin{aligned} S_{t+1} &= X_1^{\alpha_1\delta_{t+1}\rho_{II_{t+1}}} * \cdots * X_1^{\alpha_N\delta_{t+1}\rho_{II_{t+1}}} = (X_1^{\delta_{t+1}\rho_{II_{t+1}}})^{\alpha_1+\cdots+\alpha_N} \\ &= (M_{t+1})^{\delta_{t+1}\rho_{II_{t+1}}}. \end{aligned}$$

Thus, eq. (6) is satisfied. Hence, eq. (7) is satisfied for some z. □

Finally, from eq. (9) and Step 5 of Unblinding II, the output of the MIXEXP becomes as follows.

$$\begin{aligned} \tilde{\sigma}_1 &= \tilde{\sigma}_{I1}^{1/\rho_{II}} \\ &= (Z^{\alpha_1\delta}, \cdots, Z^{\alpha_N\delta}) \\ &= ((c_1 * \cdots * c_N)^{\alpha_1\delta}, \cdots, (c_1 * \cdots * c_N)^{\alpha_N\delta}). \end{aligned}$$

This is clearly different from a permutation of $(c_1^\delta, \ldots, c_N^\delta)$. (See Step 5 of Unblinding II.) Therefore, the MIXEXP does not compute the correct output without being detected.

4 Proposed MIX Net

In this section, we show an efficient t-resilient MIX net by using a certain combinatorial structure over the set of MIX servers.

In Sect. 4.1, we introduce two new concepts, existential-honesty and limited-open-verification. They will be useful for distributed computation in general. We also define a combinatorial structure which guarantees that our scheme is t-resilient.

Our scheme is given in Sect. 4.2 and Sect. 4.3. We further show an efficient construction of the combinatorial structure by using *covering* [29] in Sect. 4.4. *Covering* has recently been used in another cryptographic application: robust secret sharing by Rees et al. [23].

4.1 Existential-Honesty and Limited-Open-Verification

A set system is a pair $(X, \mathcal{B})$, where $X \stackrel{\triangle}{=} \{1, 2, \ldots, v\}$ and $\mathcal{B}$ is a collection of blocks $B_i \subset X$ with $i = 1, 2, \ldots, b$. First, we define (v, b, t)-verifiers set systems.

Definition 1. *We say that $(X, \mathcal{B})$ is a (v, b, t)-verifiers set system if*

1. $|B_i| = t + 1$ *for* $i = 1, 2, \ldots, b$ *and*
2. *for any subset* $F \subset X$ *with* $|F| \leq t$, *there exists a* $B_i \in \mathcal{B}$ *such that* $F \cap B_i = \emptyset$.

Let $(X, \mathcal{B})$ be a (v, b, t)-verifiers set system. We identify X with the set of MIX servers. Therefore, B_i is a subset of MIX servers of size $t+1$. We choose $P_i \in B_i$ arbitrarily for $1 \leq i \leq b$. P_i is called a prover. The other MIX servers of B_i are called verifiers.

We introduce two new concepts in this paper,

- Existential-honesty and
- t-open-verification

which we now describe.

Existential honesty follows from Definition 1. Although, existential honesty is not limited to applications in the MIX context, and may be useful in other distributed computation, we focus on its MIX application. In each block one designated party will mix the ciphertexts. *As long as one block of MIX servers is free of dishonest machines, the goal of mixing has been achieved.* Now we do not know which block satisfies this property. However, Definition 1 guarantees that there always exists one block of honest parties. So, we let *each* block mix the ciphertexts (i.e. the designated party of that block). What do we do when the designated party of a block B_j is dishonest? Since a block has $t+1$ parties, it must be detected. Indeed, there are at most t dishonest parties. If it is detected, then we ignore the output and proceed with the output of block B_{j-1} (or an even earlier one if the designated party in B_{j-1} was dishonest). Now, what happens when one of the verifiers falsely accuses the mixing party of having been dishonest. Then we know that this block is not free of dishonest parties, and therefore the block under consideration is not the one of Definition 1, so we can just ignore the output of the block. In other words, we do not have to decide whether the one who mixed it was honest or not.

We now explain *t-open-verification*. In many secure distributed computation protocols zero-knowledge is used to prove that the computation was done correctly. In our approach we do not need zero-knowledge, the prover will reveal the secrets he used. However, he will only do this to t parties. The existential honesty guarantees that all parties in *at least one* of the blocks of MIX servers will *all* be honest. So, the prover can reveal the secret he used. This speeds up the verification dramatically.

We now formally describe the scheme in full detail.

4.2 Initialization

Let $y(= g^x \bmod p)$ be a public key of the ElGamal scheme as shown in Sect. 2.2. We assume that the secret key x is distributed among v MIX servers by using Shamir's $(t+1, v)$ secret sharing scheme. Actually, we use a robust $(t+1, v)$ threshold ElGamal decryption scheme. (See the end of the next subsection for more details.)

1. Each user i computes a ciphertext $c_i = (a_i, b_i, aux_i, sig_i)$ by the non-malleable ElGamal encryption scheme [28] as shown in Sect. 2.3. That is, $(a_i, b_i) = (g^{\gamma_i}, m_i y^{\gamma_i})$ is the ciphertext of the usual ElGamal scheme, aux_i is the auxiliary information and sig_i is the Schnorr's signature of (a_i, b_i, aux_i) such that a_i is a public key and γ_i is the secret key.
2. Each user i posts his ciphertext c_i to the bulletin board.
3. c_i is discarded if the signature is not valid.

4.3 Main Protocol

We assume that all MIX servers of B_i share a common key e_i for $1 \le i \le b$. We extract (a_i, b_i) from a valid ciphertext c_i. Let

$$A_0 = ((a_1, b_1), \ldots, (a_N, b_N)).$$

We wish to produce a random permutation of the list $(m_1, \ldots, m_N)$, where $m_i = b_i / a_i^x$ is the plaintext of (a_i, b_i). A prover of a block B_i first publishes A_1 which is a randomly permuted list of reencrypted ciphertexts of A_0. He then privately broadcasts the secret random string R_i he used to the verifiers in the same block B_i. Each verifier of B_i checks the validity of A_0 by using R_i.

For $j = 1, \ldots, b$, do:

Step 1. Let

$$A_0 = ((\hat{a}_1, \hat{b}_1), \ldots, (\hat{a}_N, \hat{b}_N)).$$

The prover P_j of block B_j chooses random numbers $s_1, \ldots, s_N$ and a random permutation π_j. She computes

$$A_1 = \pi_j((\hat{a}_1 g^{s_1}, \hat{b}_1 y^{s_1}), \ldots, (\hat{a}_N g^{s_N}, \hat{b}_N y^{s_N})).$$

and then publishes A_1. (A_1 is commonly used for all the verifiers of B_j.)

Step 2. P_j encrypts $s_1, \ldots, s_N$ and π_j by the key e_j of block B_j. Then P_j publishes these ciphertexts. (P_j is broadcasting $s_1, \ldots, s_N$ and π_j secretly to all the verifiers of B_j.)

Step 3. Each verifier of block B_j decrypts the above ciphertexts and checks whether A_1 is computed correctly by using $s_1, \ldots, s_N$ and π_j. He outputs "ACCEPT" if A_1 is computed correctly and "REJECT" otherwise.

Step 4. If some verifier of block B_j outputs "REJECT", then A_1 is *ignored*. Otherwise, let $A_0 := A_1$.

Let the final result be $A_0 = ((\hat{c}_1, \hat{d}_1), \ldots, (\hat{c}_N, \hat{d}_N))$.

Next any $(t+1)$ MIX servers decrypt each $(\hat{c}_i, \hat{d}_i)$ by using a robust $(t+1, v)$ threshold ElGamal decryption scheme. Finally, we obtain a random permutation of the list $(m_1, \ldots, m_N)$.

Gennaro et al. showed a robust threshold RSA signature scheme in [12]. A robust $(t+1, v)$ threshold ElGamal decryption scheme is easily obtained by applying their technique to ElGamal decryption.

4.4 Construction of the Set System

Let $v = (t+1)^2$, $b = t+1$, $X = \{1, 2, \cdots, (t+1)^2\}$ and $B_i = \{(i-1)(t+1)+1, \cdots, i(t+1)\}$ for $1 \leq i \leq b$. Then it is easy to see that $(X, \mathcal{B})$ is a $((t+1)^2, t+1, t)$-verifiers set system.

We next show a more efficient (v, b, t)-verifiers set system. A set system $(X, \mathcal{B})$ is called a (v, k, t)-covering if [29]

1. $|B_i| = k$ for $1 \leq i \leq b$ and
2. every t-subset of X is included in at least one block.

From [17], we have the following proposition. (We learned about proposition 1 from [23].)

Proposition 1. *Suppose that $k = even$ and*

$$3 \leq s \leq \frac{t+3}{2},$$
$$k\left(t - \frac{s-3}{2}\right) \leq v < k\left(t - \frac{s-4}{2}\right).$$

Then there exists a $(v, v-k, t)$-covering such that $b = t+s$. Further, each element $a \in X$ is included in at most two blocks.

See [17, 23] for the construction.

We next borrow the following lemma from [23] in which the lemma was used for robust secret sharing schemes. The proof will be clear.

Lemma 1. *$(X, \mathcal{B})$ is a (v, b, t)-verifiers set system if and only if the set system $(X, \mathcal{B}^c)$ is a $(v, v-t-1, t)$-covering, where $\mathcal{B}^c \stackrel{\triangle}{=} \{X \backslash B_i \mid B_i \in \mathcal{B}\}$.*

Then we obtain a (v, b, t)-verifiers set system as follows.

Corollary 1. *For $t = odd$, there exists a (v, b, t)-verifiers set system such that*

$$v = \frac{3}{4}(t+1)^2 \text{ and } b = \frac{3}{2}(t+1). \tag{10}$$

Further, each element $a \in X$ is included in at most two blocks.

Proof: In Proposition 1, let $k = t + 1$ and $s = (t + 3)/2$. □

The fact that each MIX server is included in at most two blocks is primordial to understand the efficiency analysis described further on.

We show a small example of Corollary 1. Let $t = 3, b = 6, v = 12$ and

$$B_1 = (1, 2, 3, 4), B_2 = (3, 4, 5, 6), B_3 = (5, 6, 1, 2),$$

$$B_4 = (7, 8, 9, 10), B_5 = (9, 10, 11, 12), B_6 = (11, 12, 7, 8)$$

Then it is easy to see that this is a $(12, 6, 3)$-verifiers set system which satisfies Corollary 1.

4.5 Efficiency

In the (v, b, t)-verifiers set system of Corollary 1, each MIX server is included in at most two blocks. Therefore, each MIX server acts as a prover at most twice and acts as a verifier at most twice.

In Step 1 and Step 2, each prover computes A_1 and encrypts $s_1, \cdots, s_N$ and π_j. This computation cost is $O(N)$. He publishes A_1 and the ciphertexts of $s_1, \cdots, s_N$ and π_j. This communication cost is $O(N)$. Therefore, the total cost of the prover is $O(N)$. In Step 3, each verifier decrypts the ciphertexts of $s_1, \cdots, s_N$, π_j and checks the validity of A_1. This computation cost is $O(N)$. He publishes "ACCEPT" or "REJECT". This communication cost is $O(1)$. Therefore, the total cost of the verifier is $O(N)$. In the end, the total cost of each MIX server is $O(N)$.

An alternative method to compute the computation cost per user is to analyze the total cost and then divide by the total number of MIX servers. One needs then to take into account that the number of MIX servers is $O(t^2)$, compared to $O(t)$ in previous work.

5 Security of the Protocol

5.1 Verifiability

Suppose that the prover P_j of B_j is malicious and A_1 is not correctly computed. Then there exists at least one honest verifier in B_j because $|B_j| = t+1$ and there exist at most t malicious MIX servers. The honest verifier outputs "REJECT" at Step 3. Therefore, A_1 is ignored at Step 4.

5.2 Robustness

For any t malicious MIX servers, there exists at least one B_i in which all MIX server are honest from Def.1. This B_i computes A_1 correctly. On the other hand, any invalid A_1 is ignored from the verifiability. Therefore, our protocol outputs a random permutation of $(m_1, \ldots, m_N)$ correctly even if there are at most t malicious MIX servers.

5.3 Privacy (Sketch)

The ElGamal based encryption scheme of [28] is known to be non-malleable under adaptive chosen ciphertext attack. Let c_1, c_2 be two ciphertexts and m_1, m_2 be the plaintexts. Then by using the result of [4], we can show that there exists no probabilistic polynomial time (p.p.t.) Turing machine D (distinguisher) which can distinguish (c_1, c_2, m_1, m_2) and (c_1, c_2, m_2, m_1) with meaningful probability. This is the minimum requirement that any MIX net of this type must satisfy. We also assume that it satisfies *plaintext awareness* [5] which means that no p.p.t. adversary can create a ciphertext c without knowing its underlying plaintext m.

Now consider an adversary M_0 who can control at most t MIX servers and at most $N-2$ out of the N users posting encrypted messages. It is the goal of the adversary M_0 to match each one of the two plaintexts m_1, m_2 to their corresponding ciphertexts c_1, c_2 that he does not control. In other words, M_0 wishes to distinguish (c_1, c_2, m_1, m_2) and (c_1, c_2, m_2, m_1).

Suppose that there exists a p.p.t. adversary M_0 who can distinguish (c_1, c_2, m_1, m_2) and (c_1, c_2, m_2, m_1) with meaningful probability. For simplicity, suppose that M_0 controls users $3, \cdots, N$.

We will show a distinguisher D. The input to D is (c_1, c_2, z_1, z_2), where $(z_1, z_2) = (m_1, m_2)$ or (m_2, m_1). D first gives c_1, c_2 to M_0 and runs the users part of M_0. Then M_0 outputs $c_3, \cdots, c_N$. From the *plaintext awareness* assumption, M_0 knows the plaintexts $m_3, \cdots, m_N$ for $c_3, \cdots, c_N$. Therefore, D knows the set of $\{m_1, m_2, m_3, \cdots, m_N\}$.

D next runs the main body of our protocol in such a way that D simulates the part of honest MIX servers faithfully and uses M_0 for the part of malicious MIX servers. Let the output of the main body be $A_0 = ((\hat{c}_1, \hat{d}_1), \ldots, (\hat{c}_N, \hat{d}_N))$. Note that A_0 is a random permutation of randomized ciphertexts $c_1, \cdots, c_N$ from Sect. 5.2.

Let π be a random permutation. Let $\hat{m}_i$ denote the plaintext of $(\hat{c}_i, \hat{d}_i)$ for $1 \leq i \leq N$. Then we can show that M_0 cannot distinguish $(\hat{c}_i, \hat{d}_i, \hat{m}_i)$ and $(\hat{c}_i, \hat{d}_i, m_{\pi(i)})$ under the decision Diffie-Hellman assumption. D finally generates a view of M_0 for the robust $(t+1, n)$ threshold ElGamal decryption scheme with $(\hat{c}_i, \hat{d}_i, m_{\pi(i)})$ by using the technique of [8, 12] for $1 \leq i \leq N$.

Then M_0 can distinguish (c_1, c_2, m_1, m_2) and (c_1, c_2, m_2, m_1) with meaningful probability from our assumption on M_0. Hence, D can distinguish (c_1, c_2, m_1, m_2) and (c_1, c_2, m_2, m_1) with meaningful probability. However, this is a contradiction.

6 Open Problems

This paper introduces several open problems, in particular:

- whether the new tools of existential-honesty and limited-open-verification can be used in other secure distributed computation.

- whether there are other choices of v. Indeed, when t is large the required number of MIX servers only grows quadratic. Although this is reasonable for a theoretician, from a practical viewpoint, the question is worth addressing.
- is $O(N)$ the minimum required effort per MIX server while maintaining t-privacy, t-verifiability, and t-robustness, in a network with $O(t^2)$ servers.

Acknowledgement

The authors thank Prof. Stinson for providing in August 1999 the second author of the paper with a preliminary copy of their Rees et al. paper [23]. The authors are grateful for the many comments received from the referee and from Dr. Jakobsson who shepherded the rewriting of the paper. Many of these comments have significantly contributed to increase the readability of the text.

References

1. M. Abe, "Universally verifiable mix-net with verification work independent of the number of mix-centers," Eurocrypt '98, pp. 437–447.
2. M. Abe, "A mix-network on permutation networks," ISEC Technical report 99-10 (in Japanese) (May, 1999)
3. M. Abe, "Mix-networks on permutation networks," Asiacrypt '99, pp. 258–273.
4. M. Bellare, A. Desai, D. Poincheval, P. Rogaway, "Relations among notions of security for public key encryption schemes," Crypto '98, pp. 26–45.
5. M. Bellare, P. Rogaway, "Optimal asymmetric encryption - How to encrypt with RSA," Eurocrypt '94, pp. 92–111.
6. D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Communications of the ACM, Vol. 24, 1981, pp. 84-88.
7. D. Chaum, H. Van Antwerpen, "Undeniable signatures," Crypto '89, pp. 212-216.
8. Y. Desmedt, Y. Frankel, "Threshold cryptosystems," Crypto '89, pp. 307–315.
9. D. Dolev, C. Dwork, M. Naor, " Non-malleable cryptography," STOC '91, pp. 542-552.
10. T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," Crypto '84, pp. 10-18.
11. A. Fujioka, T. Okamoto, K. Ohta, "A practical secret voting scheme for large scale elections," Auscrypt '92, pp. 244-251.
12. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Robust and efficient sharing of RSA functions," Crypto '96, pp. 157–172.
13. M. Jakobsson, "A practical MIX," Eurocrypt '98, pp. 448–461.
14. M. Jakobsson, D. M'Raihi, "Mix-based electronic payments," SAC'98, pp. 157–173.
15. M. Jakobsson, "Flash mixing," PODC'99, pp. 83–89.
16. M. Jakobsson, A. Juels "Millimix: Mixing in small batches," DIMACS Technical report 99-33 (June 1999)
17. W. H. Mills, "Covering design I: coverings by a small number of subsets," Ars Combin. 8, (1979), pp. 199–315.
18. W. Ogata, K. Kurosawa, K. Sako, K. Takatani, "Fault tolerant anonymous channel," ICICS '97, pp. 440-444.
19. C. Park, K. Itoh, K. Kurosawa, "All/nothing election scheme and anonymous channel," Eurocrypt '93, pp. 248-259.

20. T. P. Pedersen, "A threshold cryptosystem without a trusted party," Eurocrypt '91, pp. 522-526.
21. B. Pfitzmann, A. Pfitzmann. "How to break the direct RSA-implementation of MIXes," Eurocrypt '89, pp. 373-381.
22. D. Pointcheval, J. Stern, "Security proofs for signature schemes," Eurocrypt '96, pp. 387-398.
23. R. Rees, D. R. Stinson, R. Wei, G. H. J. van Rees, "An application of covering designs: Determining the maximum consistent set of shares in a threshold scheme," Ars Combin. 531 (1999), pp. 225–237.
24. K. Sako, J. Kilian, "Receipt-free mix-type voting scheme," Eurocrypt '95, pp. 393-403.
25. C. P. Schnorr, "Efficient signature generation for smart cards," Crypto '89, pp. 239–252.
26. C. P. Schnorr, M. Jakobsson, "Security of discrete log cryptosystems in the random oracle + generic model," `http://www.bell-labs.com/user/markusj/`
27. A. Shamir, "How to share a secret," Communications of the ACM, Vol. 22, 1979, pp. 612-613
28. Y. Tsiounis, M. Yung, "On the security of ElGamal based encryption," PKC'98, pp. 117–134.
29. Edited by C. J. Colbourn and J. H. Dinitz, Handbook of Combinatorial Design, CRC Press (1996)

Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5

Anne Canteaut[1] and Michaël Trabbia[1,2]

[1] INRIA projet CODES
B.P. 105
78153 Le Chesnay Cedex - France
Anne.Canteaut@inria.fr
[2] Ecole Polytechnique
91128 Palaiseau Cedex - France
michael.trabbia@enst.fr

Abstract. This paper describes new techniques for fast correlation attacks, based on Gallager iterative decoding algorithm using parity-check equations of weight greater than 3. These attacks can be applied to any key-stream generator based on LFSRs and it does not require that the involved feedback polynomial have a low weight. We give a theoretical analysis of all fast correlation attacks, which shows that our algorithm with parity-check equations of weight 4 or 5 is usually much more efficient than correlation attacks based on convolutional codes or on turbo codes. Simulation results confirm the validity of this comparison. In this context, we also point out the major role played by the nonlinearity of the Boolean function used in a combination generator.

1 Introduction

Stream ciphers form an important class of secret-key encryption schemes. They are widely used in applications since they present many advantages: they are usually faster than common block ciphers and they have less complex hardware circuitry. Moreover, their use is particularly well-suited when errors may occur during the transmission because they avoid error propagation. In a binary additive stream cipher the ciphertext is obtained by adding bitwise the plaintext to a pseudo-random sequence s, called the *running-key* (or the *key stream*). The running-key is produced by a pseudo-random generator whose initialization is the secret key shared by the users. Most attacks on such ciphers therefore consist in recovering the initialization of the pseudo-random generator from the knowledge of a few ciphertext bits (or of some bits of the running-key in known-plaintext attacks).

Linear feedback shift registers (LFSRs) are the basic components of most key-stream generators since they are appropriate to hardware implementations, produce sequences with good statistical properties and can be easily analyzed. Different classes of key-stream generators can be distinguished depending on the

techniques used for combining the constituent LFSRs [16]: combination generators, filter generators, clock-controlled generators ... . In all these systems, the secret key usually consists of the initial states of the constituent LFSRs. The secret key has then $\sum_{i=1}^{n} L_i$ bits, where L_i denotes the length of the i-th LFSR and n is the number of involved LFSRs. Any key-stream generator based on LFSRs is vulnerable to *correlation attacks*. These cryptanalytic techniques introduced by Siegenthaler [20] are "divide-and-conquer" methods: they exploit the existence of a statistical dependence between the running-key and the output of one constituent LFSR for recovering the initialization of each LFSR separately. The secret key can then be recovered with only $\sum_{i=1}^{n} 2^{L_i}$ tests.

A classical method for generating a running-key is to combine n LFSRs by a nonlinear Boolean function f. Such a combination generator is depicted in Figure 1. For combination generators, the original correlation attack presented by

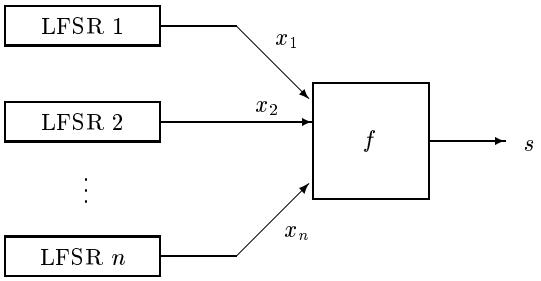


Fig. 1. Combination generator

Siegenthaler can be prevented by using a *correlation-immune* combining function [19]. In this case, the running-key is statistically independent of the output of each constituent LFSR; any correlation attack should then consider several LFSRs together. More generally, a correlation attack on a set of k LFSRs, namely LFSR $i_1, \ldots$, LFSR i_k, exploits the existence of a correlation between the running-key s and the output σ of a smaller combination generator, which consists of the k involved LFSRs combined by a Boolean function g with k variables (see Fig. 2). Since $Pr[s_n \neq \sigma_n] = Pr[f(X_1, \ldots, X_n) \neq g(X_{i_1}, \ldots, X_{i_k})] = p_g$, this attack only succeeds when $p_g < 0.5$. The number k of involved LFSRs should then be strictly greater than the correlation immunity order t of the combining function f. This cryptanalysis therefore requires that all $2^{\sum_{j=1}^{t+1} L_{i_j}}$ initial states be examined; it becomes infeasible when the correlation-immunity order t of the combining function is high.

The fast correlation attack proposed by Meier and Staffelbach [9, 10] relies on the same principle but it avoids examining all possible initializations of $(t+1)$ LFSRs together. Let us again consider the sequence σ produced by LFSR $i_1, \ldots$,

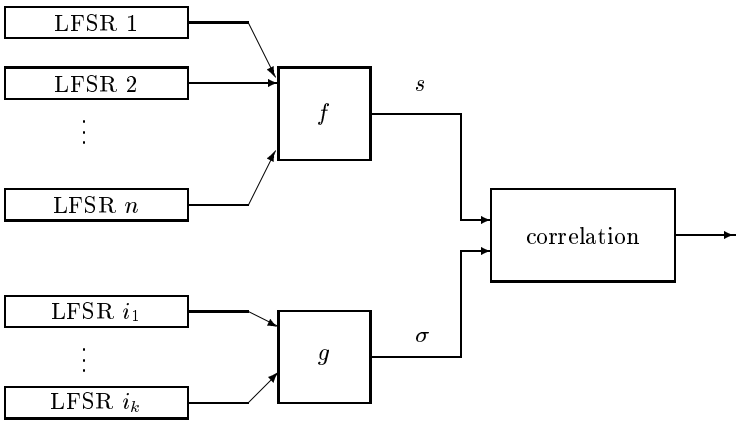


Fig. 2. Correlation attack involving k constituent LFSRs

LFSR i_k combined by g. The sequence σ obviously corresponds to the output of a unique LFSR of length L; the length and the feedback polynomial P of this LFSR can be derived from the feedback polynomials of the constituent LFSRs. Note that $L \leq g(L_{i_1}, \ldots, L_{i_k})$ where the function g is evaluated over integers. Equality notably holds when the feedback polynomials of the involved LFSRs are primitive and when their degrees are coprime [17]. Any subsequence of length N of σ is then a codeword of a linear code $\mathcal{C}$ of length N and dimension L defined by the feedback polynomial P. The running-key subsequence $(s_n)_{n<N}$ can then be seen as the result of the transmission of $(\sigma_n)_{n<N}$ through the binary symmetric channel with error probability (or crossover probability) $p = Pr[s_n \neq \sigma_n]$. The attack therefore aims at recovering L consecutive bits of σ (i.e., the initial state of the equivalent LFSR) from the knowledge of N bits of s. This can be done by decoding $(s_n)_{n<N}$ relatively to $\mathcal{C}$.

From the attacker's point of view, the main problem is to make the cryptanalysis feasible even if a small number N of bits of the running-key (or of the

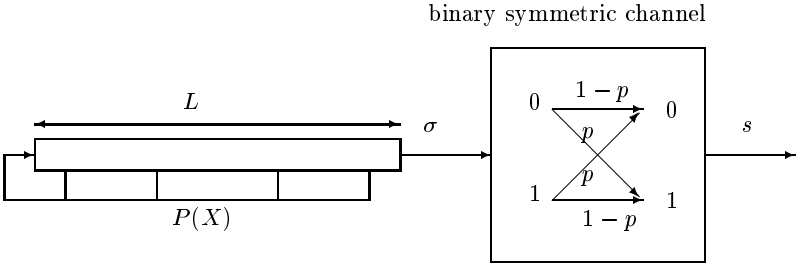


Fig. 3. Model for a fast correlation attack

ciphertext) is known. Shannon's channel coding theorem [18] gives a theoretical lower bound on N depending on the error-probability p: $N \geq LC(p)$, where $C(p)$ is the capacity of the binary symmetric channel with error-probability p, i.e., $C(p) = 1 + p\log_2(p) + (1-p)\log_2(1-p)$. Unfortunately no efficient general decoding algorithm is known for achieving the channel capacity. This means that practical correlation attacks require that the known running-key sequence be much longer than this theoretical bound. Any improvement of fast correlation attacks then consists in finding an efficient decoding procedure for the code $\mathcal{C}$, when N is as close as possible to Shannon's limit.

Meier and Staffelbach attack [10] uses the iterative decoding process due to Gallager for low-density parity-check codes [3]. Any polynomial P^{2^i} actually provides a parity-check equation for $\mathcal{C}$ as far as its degree is less than N. It follows that the received sequence $(s_n)_{n<N}$ can be decoded with Gallager algorithm when the feedback polynomial P has a low weight and when the error probability p is not too high. Several minor improvements of this original attack were proposed in [21, 12, 1, 13, 15] but these papers did not introduce any important modification of the basic underlying concepts. Johansson and Jönsson recently proposed two new techniques for fast correlation attacks: the main idea is to derive from $(s_n)_{n<N}$ a sequence which can be seen as a corrupted version of a word of a convolutional code [6] or of a turbo code [7]. These new attacks increase the highest achievable error probability p for given values of L and N (L is the length of the LFSR generating σ and N is the number of known bits of the running-key). Moreover, they do not require that the feedback polynomial P have a low weight. We here show that Gallager iterative decoding algorithm with parity-check equations of weight 4 or 5 is usually much more efficient than all previous attacks: it successfully decodes very high error probabilities with a feasible time and memory complexity, and it does not require that the feedback polynomial P have a low weight. As an example, for a LFSR of length $L = 40$ and an error-probability $p = 0.3$, the best previously known attack [7] requires the knowledge of $N = 40,000$ bits of s whereas our algorithm with parity-check equations of weight 5 is successful with only $9,770$ bits.

The paper is organized as follows. Section 2 focuses on the particular case of combination generators. Here, we prove that the lowest possible Hamming distance between a fixed t-resilient function and any Boolean function with $(t+1)$ variables is achieved by an affine function. It follows that the nonlinearity of the combining function plays a major role in the resistance of a combination generator to correlation attacks. The rest of the paper presents a new general method for fast correlation attacks which can be applied to any type of keystream generators based on LFSRs. The preprocessing step and the decoding step of the algorithm are respectively described in Section 3 and 4. Section 5 gives a theoretical analysis of the recent attacks proposed by Johansson and Jönsson. Most notably, we point out that our attack using parity-check equations of weight 4 or 5 has better performance than the attacks based on convolutional codes or on turbo codes. Section 6 finally presents some simulation results which confirm the validity of the previous comparison.

2 Approximation of a t-resilient function by a function with $t+1$ variables

This section is devoted to the special case of combination generators. It focuses on the choice of the Boolean function g which is used for combining the k LFSRs involved in the correlation attack (see Fig. 2). The attack will be even more efficient that the correlation between the running-key s and the sequence σ is high. This equivalently means that $Pr[s_n \neq \sigma_n]$ should be as small as possible. The Boolean function g with k variables should then minimize $p_g = Pr[f(X_1, \ldots, X_n) \neq g(X_{i_1}, \ldots, X_{i_k})]$. Moreover, since the length of the equivalent LFSR considered in a fast correlation attack is usually given by $L = g(L_{i_1}, \ldots, L_{i_k})$, it is obviously required that the degree of g be not too high.

We first recall some basic properties of Boolean functions (see e.g. [8], [11] and [2] for details). In the following, $\mathcal{F}_n$ denotes the set of all Boolean functions with n variables, i.e., the set of all functions from $\mathbf{F}_2^n$ into $\mathbf{F}_2$. A Boolean function is balanced if its output is uniformly distributed; balancedness is then an obvious requirement for combining functions. A Boolean function $f \in \mathcal{F}_n$ is t-th order correlation-immune if the probability distribution of its output is unaltered when any t input variables are fixed [19]. Balanced t-th order correlation-immune functions are called t-resilient. Note that a t-th order correlation-immune function is k-th order correlation-immune for any $k \leq t$. From now on, the correlation-immunity order of a function f then refers to the highest integer t such that f is t-th order correlation-immune. The *Walsh transform* of a Boolean function $f \in \mathcal{F}_n$ is the Fourier transform of the corresponding sign function $\chi_f(x) = (-1)^{f(x)}$:

$$\forall u \in \mathbf{F}_2^n, \quad \widehat{\chi_f}(u) = \sum_{x \in \mathbf{F}_2^n} (-1)^{f(x)} (-1)^{u \cdot x}$$

where $x \cdot y$ denotes the usual dot product between two n-bit vectors x and y. The Walsh coefficient $\widehat{\chi_f}(u)$ estimates the Hamming distance between f and the affine function $u \cdot x + \varepsilon$, $\varepsilon \in \mathbf{F}_2$:

$$Pr[f(X_1, \ldots, X_n) \neq u \cdot X + \varepsilon] = \frac{1}{2} - \frac{(-1)^{\varepsilon}}{2^{n+1}} \widehat{\chi_f}(u) \ .$$

The nonlinearity of f, $\mathcal{NL}(f)$, corresponds to its Hamming distance to the set of affine functions:

$$\mathcal{NL}(f) = 2^{n-1} - \frac{1}{2} \max_{u \in \mathbf{F}_2^n} |\widehat{\chi_f}(u)| \ .$$

We now consider the combination generator depicted in Figure 1 and we assume that the combining function f is t-resilient. The running-key produced by the combination generator is then independent of any set of t constituent LFSRs. The smallest number of LFSRs involved in a correlation attack is therefore $t+1$. We now prove that, in this case, the Boolean function g with $(t+1)$ variables which provides the best approximation to f (i.e., which minimizes p_g) is an affine function.

Theorem 1. *Let f be a t-resilient function with n variables and let T be a subset of $\{1,\ldots,n\}$ of cardinality $(t+1)$, $T=\{i_1,\ldots,i_{t+1}\}$. The lowest possible value over all $g\in\mathcal{F}_{t+1}$ of*

$$p_g = Pr[f(X_1,\ldots,X_n)\neq g(X_{i_1},\ldots,X_{i_{t+1}})]$$

is achieved by the affine function

$$g(x_{i_1},\ldots,x_{i_{t+1}}) = \sum_{i\in T} x_i + \varepsilon$$

with $\varepsilon=0$ if $\widehat{\chi_f}(1_T)>0$ and $\varepsilon=1$ otherwise, where 1_T denotes the n-bit vector whose i-th component equals 1 if and only if $i\in T$.

Moreover, we have

$$\min_{g\in\mathcal{F}_{t+1}} p_g = \frac{1}{2} - \frac{1}{2^{n+1}}|\widehat{\chi_f}(1_T)| \ .$$

Proof: For any vector $x\in\mathbf{F}_2^n$, $x=(y,z)$ refers to the decomposition of x with respect to T, i.e., y is the $(t+1)$-bit vector composed of all $x_i, i\in T$. Let $p_T(y)$, $y\in\mathbf{F}_2^{t+1}$, denote the probability $p_T(y)=Pr[f(Y,Z)=1|Y=y]$. For any $g\in\mathcal{F}_{t+1}$ we have

$$\begin{aligned} p_g &= Pr[f(Y,Z)\neq g(Y)] \\ &= \sum_{y\in g^{-1}(0)} Pr[f(Y,Z)=1|Y=y] + \sum_{y\in g^{-1}(1)} Pr[f(Y,Z)=0|Y=y] \\ &= \sum_{y\in g^{-1}(0)} p_T(y) + \sum_{y\in g^{-1}(1)} (1-p_T(y)) \ . \end{aligned}$$

It follows that p_g is minimal if and only if

$$g(x) = \begin{cases} 0 & \text{if } p_T(x)<1/2, \\ 1 & \text{if } p_T(x)>1/2 \ . \end{cases} \qquad (1)$$

Note that the value of $g(x)$ can be arbitrarily chosen when $p_T(x)=\frac{1}{2}$. For any $j\in T$, e_j denotes the $(t+1)$-bit vector whose all coordinates are zero except the j-th one. For any $y\in\mathbf{F}_2^{t+1}$ and any $j\in T$, we have

$$\begin{aligned} p_T(y)+p_T(y+e_j) &= Pr[f(Y,Z)=1|Y=y] + Pr[f(Y,Z)=1|Y=y+e_j] \\ &= 2\,(Pr[f(Y,Z)=1|\forall i\in T, Y_i=y_i]Pr[Y_j=y_j]+ \\ &\quad Pr[f(Y,Z)=1|\forall i\in T\setminus\{j\}, Y_i=y_i, Y_j\neq y_j]Pr[Y_j\neq y_j]) \\ &= 2Pr[f(Y,Z)=1|\forall i\in T\setminus\{j\}, Y_i=y_i] = 1 \end{aligned}$$

where the last equality comes from the fact that f is t-resilient and that the set $T\setminus\{j\}$ has cardinality t. Let $g\in\mathcal{F}_{t+1}$ be such that p_g is minimal. Since $p_T(x)+p_T(x+e_j)=1$ for any $x\in\mathbf{F}_2^{t+1}$ and for any $j\in T$, Condition (1) implies that

$$g(x)+g(x+e_j)\equiv 1 \bmod 2$$

when $p_T(x) \neq \frac{1}{2}$. Moreover, we can assume that this relation is satisfied for any $x \in \mathbf{F}_2^{t+1}$, because the value of $g(x)$ can be arbitrarily chosen when $p_T(x) = \frac{1}{2}$. It follows that, for any $x \in \mathbf{F}_2^{t+1}$,

$$g(x) = g(0) + \sum_{i \in T} x_i \ .$$

Since g is an affine function, p_g is given by

$$p_g = \frac{1}{2} - \frac{(-1)^{g(0)}}{2^{n+1}} \widehat{\chi_f}(1_T) \ .$$

This probability is then minimized when $(-1)^{g(0)}$ and $\widehat{\chi_f}(1_T)$ have the same sign. □

It follows that, in a fast correlation attack involving $(t+1)$ LFSRs, the same combining function g minimizes both the error probability p_g and the length of the LFSR generating σ. In this context, the feedback polynomial P of this equivalent LFSR is the least common multiple of the feedback polynomials P_i of the considered LFSRs [22]. Note that we generally have $P = \prod_{i \in T} P_i$ since all these feedback polynomials are usually primitive. The running-key s can be seen as the result of the transmission of the sequence σ generated by this LFSR through the binary symmetric channel with error probability

$$p = \frac{1}{2} - \frac{1}{2^{n+1}} |\widehat{\chi_f}(1_T)| \ .$$

A similar cryptanalytic method applies to a ciphertext-only attack. In this case, we make use of the redundancy of the plaintext sequence m, i.e., $Pr[m_n = 0] = p_0 > 1/2$. The attack now considers that the ciphertext sequence c results of the transmission of σ through the binary symmetric channel with error-probability

$$p = Pr[c_n \neq \sigma_n] = \frac{1}{2} - \frac{(2p_0 - 1)}{2^{n+1}} |\widehat{\chi_f}(1_T)| \ .$$

Theorem 1 points out the importance of the nonlinearity of the combining function f: any known-plaintext correlation attack on $(t+1)$ LFSRs should decode an error probability $p \geq \mathcal{NL}(f)/2^n$. The use of highly nonlinear combining function may then prevent this attack. In this case, an acceptable error probability can only be obtained when $(t+2)$ constituent LFSRs are involved and when the degree of g is at least 2. But this dramatically increases the length of the equivalent LFSR and it makes any correlation attack infeasible.

3 Generating parity-check equations

We now come back to the general cryptanalysis of any key-stream generator based on LFSRs. A fast correlation attack aims at recovering L consecutive bits

of σ from the knowledge of N bits of s (see Fig. 3). We use that any N-bit subsequence of σ is a codeword of a linear code $\mathcal{C}$ of length N and dimension L defined by the feedback polynomial P. The preprocessing step of the attack then consists in generating some parity-check equations for $\mathcal{C}$ (i.e., linear relations involving some bits of $(\sigma_n)_{n<N}$) in such a way that they provide an efficient decoding procedure. Here, we use a fast decoding algorithm due to Gallager [3] for low-density parity-check codes. In this context, the preprocessing step consists in searching for all linear equations involving d bits of the sequence $(\sigma_n)_{n<N}$:

$$\sigma_n + \sum_{j=1}^{d-1} \sigma_{i_j} = 0 \ .$$

These equations exactly correspond to the polynomials $Q(X)P(X)$ of weight d and of degree at most N, where P is the feedback polynomial of the LFSR generating σ. The cyclic structure of LFSR sequences implies that the set of all parity-check equations of weight d involving σ_i does not depend on i. It is therefore sufficient to find all polynomials $Q(X)P(X)$ of weight d whose constant term equals 1. These polynomials can be found with the following algorithm:

- Compute all residues $q_i(X) = X^i \bmod P(X)$ for $1 \leq i < N$ and store their values in a table T defined by

$$\forall 0 \leq a < 2^L, \ T[a] = \{i, q_i(X) = a\} \ .$$

- For each set of $d-2$ elements of $\{1, \cdots, N-1\}$
  compute $A = 1 + q_{i_1}(X) + \ldots + q_{i_{d-2}}(X)$
  for any $j \in T[A]$, $1 + X^{i_1} + \ldots + X^{i_{d-2}} + X^j$ is a multiple of P of weight d.

The number of operations required by this algorithm is then roughly

$$\binom{N-1}{d-2} \sim \frac{N^{d-2}}{(d-2)!} \ .$$

We can also use an algorithm based on a "birthday technique" as suggested in [10, Section 5]. This consists in storing in a table the values of all linear combinations of $\lfloor \frac{d-1}{2} \rfloor$ residues $q_i(X)$. The complexity of this algorithm is only $\binom{N}{\lceil \frac{d-1}{2} \rceil}$ but it requires $L\binom{N}{\lfloor \frac{d-1}{2} \rfloor}$ bits of memory. For $d > 4$ the choice of the algorithm used in the preprocessing step then highly depends on the available memory amount. Similar techniques for finding low-weight parity-check equations are presented in [14].

We now want to estimate the number of such parity-check equations of weight d. Let us first assume that P is a primitive polynomial in $\mathbf{F}_2[X]$ of degree L. Then the number $m(d)$ of polynomials $Q(X) = 1 + \sum_{i=1}^{N} q_i X^i$ of weight d such that P divides Q is approximatively

$$m(d) \simeq \frac{N^{d-1}}{(d-1)!2^L} \ . \tag{2}$$

This approximation is motivated as follows: when d is small, the number of multiples of P of weight d and of degree at most $2^L - 1$ can be approximated [8, p. 129] by

$$A = \frac{2^{(d-1)L}}{d!} .$$

Let us now assume that the probability p_d that P divides a polynomial of weight d is uniform. We then deduce that

$$p_d = \frac{A}{\binom{2^L-1}{d}} \simeq \frac{1}{2^L} .$$

We similarly obtain that

$$m(d) = p_d \binom{N-1}{d-1} \simeq \frac{N^{d-1}}{(d-1)!2^L} .$$

Simulations for $d \leq 6$ show the accuracy of this approximation when N is not too small. Moreover its validity does not depend on the weight of P. As an example the following table compares our approximation with the exact values of $m(3)$ for two polynomials of degree 17: $P_1(X) = 1 + X^3 + X^{17}$ and $P_2(X) = 1 + X^2 + X^4 + X^5 + X^6 + X^8 + X^9 + X^{10} + X^{11} + X^{13} + X^{14} + X^{15} + X^{17}$.

N	3000	4000	5000	6000	7000	8000
$m_1(3)$	38	61	95	131	183	238
$m_2(3)$	36	67	95	127	185	243
approximation	34	61	95	137	187	244

Since this approximation is also accurate when P is a product of primitive polynomials, we will now use Formula (2) as an approximation of the number of parity-check equations of weight d involving the i-th bit of $(\sigma_n)_{n<N}$. For the polynomial of degree 40 considered in [6], $P(X) = 1 + X + X^3 + X^5 + X^9 + X^{11} + X^{12} + X^{17} + X^{19} + X^{21} + X^{25} + X^{27} + X^{29} + X^{32} + X^{33} + X^{38} + X^{40}$, we obtain 9607 parity-check equations of weight 4 for $N = 400,000$ and 400 parity-check equations of weight 5 for $N = 10,000$. For these values of N, Formula (2) gives $m(4) = 9701$ and $m(5) = 379$.

4 Decoding procedure

Using the previous parity-check equations we recover $(\sigma_n)_{n<N}$ from $(s_n)_{n<N}$ using Gallager soft-input/soft-output decoding algorithm [3, 4]. It relies on the evaluation, for all $0 \leq i < N$, of the probability that σ_i equals 1 conditional on the known sequence $(s_n)_{n<N}$ and on the event S that all parity-check equations involving σ_i are satisfied. As usual in soft decoding algorithms, all probabilities are expressed in terms of log-likelihood ratios: the log-likelihood ratio of a binary random variable X, $L(X)$, is defined as

$$L(X) = \log \frac{Pr[X = 0]}{Pr[X = 1]} .$$

The sign of $L(X)$ corresponds to a hard decision on X ($\mathrm{sign}(L(X)) = (-1)^X$); its magnitude $|L(X)|$ is the reliability of this decision. Here, we have that

$$L(\sigma_i|(s_n)_{n<N}, S) = L(\sigma_i|(s_n)_{n<N}) + L(S|\sigma_i, (s_n)_{n<N}) \ .$$

The second term of the right hand member of this equation can be evaluated with the following approximation (see e.g. [5])

$$L(\sum_{i=1}^{k} X_i) = \left(\prod_{i}^{k} \mathrm{sign}(L(X_i))\right) \min_{1\leq i\leq k} |L(X_i)| \ .$$

The decoding procedure is then as follows:

- *Initialization:* for all i from 0 to $N-1$, $L[i] = \log \frac{1-p}{p}$.
- *Until convergence, repeat:*
  For all i from 0 to $N-1$
  $L'[i] = (-1)^{s_i} L[i]$.
  for any parity-check equation involving σ_i, written as $\sigma_i + \sum_{j\in J} \sigma_j = 0$,

$$L'[i] \leftarrow L'[i] + \left(\prod_{j\in J} (-1)^{s_j}\right) \min_{j\in J} L[j] \ .$$

  For all i from 0 to $N-1$, $s_i \leftarrow \mathrm{sign}(L'[i])$ and $L[i] \leftarrow |L'[i]|$.

The number of parity-check equations required for convergence of this decoding procedure highly depends on their weight d. Figures 4 and 5 present simulations results for $L = 21$ ($P(X) = 1 + X^2 + X^3 + X^5 + X^{10} + X^{11} + X^{12} + X^{14} + X^{21}$). Figure 5 clearly shows that the performance of the attack increases with the weight of the parity-check equations. For $p = 0.4$, the attack requires the knowledge of 16800 bits of s for $d = 3$, 2200 bits for $d = 4$ and 1100 bits for $d = 5$.

Simulations actually provide the following approximation of the minimum value of $m(d)$ for convergence (see Fig. 4):

$$m(d) \geq \frac{K_d}{C_{d-2}(p)} \tag{3}$$

where $C_{d-2}(p)$ is the capacity of the binary symmetric channel with error-probability $p_{d-2} = \frac{1}{2}(1-(1-2p)^{d-2})$, *i.e.* $C_{d-2}(p) = 1 + p_{d-2}\log_2(p_{d-2}) + (1-p_{d-2})\log_2(1-p_{d-2})$. $K_d \simeq 1$ if $d \geq 4$ and $K_3 \simeq 2$. Combining (2) and (3) we obtain that the correlation attack with parity-check equations of weight d is successful if the number of known bits of s is at least

$$N = 2^{\alpha_d(p) + \frac{L}{d-1}} \text{ with } \alpha_d(p) = \frac{1}{d-1}\log_2\left[(d-1)!\frac{K_d}{C_{d-2}(p)}\right] \ . \tag{4}$$

This formula points out that the influence of L decreases when we use higher-weight parity-check equations. When $m(d)$ satisfies (3) the decoding procedure

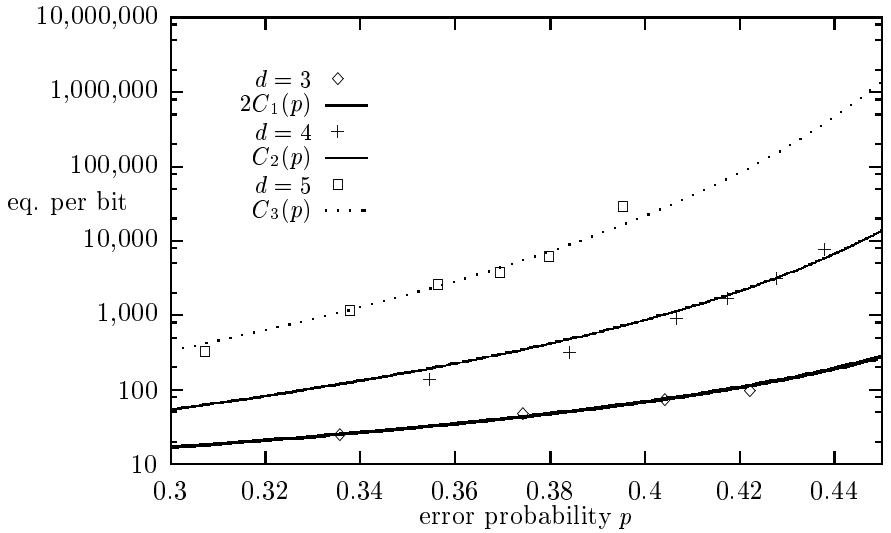


Fig. 4. Number of parity-check equations of weight d per bit required for convergence ($L = 21$)

requires at most 10 iterations. The algorithm then performs approximatively $5(d-1)m(d)N$ operations in average. The amount of involved memory is composed of $(d-1)m(d)$ computer words for storing the parity-check equations and of $2N$ computer words for storing of the sequence $(s_n)_{n<N}$ and the corresponding soft values $(L[n])_{n<N}$.

5 Comparison with previous correlation attacks

We first compare our attack with the correlation attack using convolutional codes described in [6]. This attack associates a convolutional code with memory B to the code $\mathcal{C}$ stemming from the LFSR with feedback polynomial P. The embedded convolutional code is defined by all parity-check equations involving σ_n and $d-1$ bits of σ outside positions $n-1, \ldots, n-B$:

$$\sigma_n + \sum_{i=1}^{B} \beta_i \sigma_{n-i} + \sum_{j=1}^{d-1} \sigma_{i_j} \ .$$

Johansson and Jönsson focus on the case $d = 3$. Using the algorithm described in [6] all these equations can be found with roughly $\frac{N^{d-2}}{(d-2)!}$ operations. Exactly as in Formula (2) the number of such parity-check equations involving the i-th bit of σ is approximatively given by

$$m_B(d) = \frac{N^{d-1} 2^B}{(d-1)! 2^L} \ .$$

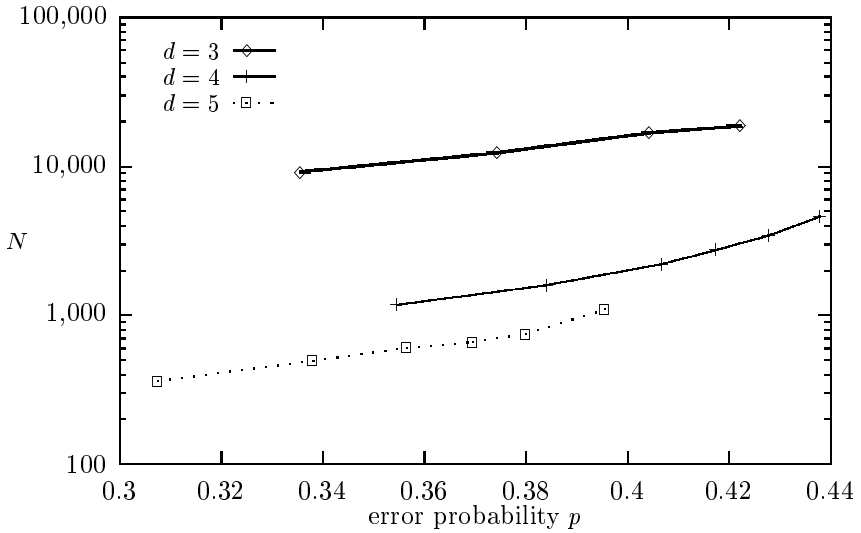


Fig. 5. Number of bits of s required for convergence ($L = 21$)

The decoding step of the attack now consists in deriving a sequence r from $(s_n)_{n<N}$. Decoding r with respect to the convolutional code then provides L consecutive bits of σ. By construction most bits of the corrupted sequence r satisfy

$$Pr[r_n \neq \sigma_n] = \frac{1}{2}(1 - (1-2p)^{d-1}) = p_{d-1} \ .$$

This obviously implies that the decoding procedure can not be successful if the transmission rate of the convolutional code R is greater than the capacity of the binary symmetric channel with error probability p_{d-1}. The simulation results presented in [6] actually provide the following maximum value of R for convergence of Viterbi algorithm:

$$R \leq \frac{C_{d-1}(p)}{K'}$$

where K' is a constant which slightly depends on L (we obtain $K' = 3$ for $L = 21$ and $K' = 2.5$ for $L = 40$). Since $R = 1/(m_B(d) + 1)$ we deduce the following convergence condition for Viterbi algorithm:

$$m_B(d) \leq \frac{K'}{C_{d-1}(p)} - 1 \ . \tag{5}$$

The number of known bits of s required by a correlation attack using a convolutional code with memory B is then

$$N = 2^{\beta_d(p) + \frac{(L-B)}{d-1}} \text{ with } \beta_d(p) = \frac{1}{d-1} \log_2 \left[(d-1)! \frac{K'}{C_{d-1}(p)} \right] \ . \tag{6}$$

The decoding step using Viterbi algorithm performs $2^B m_B(d)(L+10B)$ operations.

Figure 6 compares the number of bits of s required for a correlation attack using Gallager decoding algorithm with $d=3$, $d=4$ (Formula (4)) and for the attack using Viterbi algorithm with $d=3$, $B=18$ (Formula (6)). It points out that the use of Gallager algorithm with $d=4$ provides better performance than the use of Viterbi algorithm with $d=3$ and $B=18$. Moreover, this advantage increases for growing p and L. As an example, we now compare our

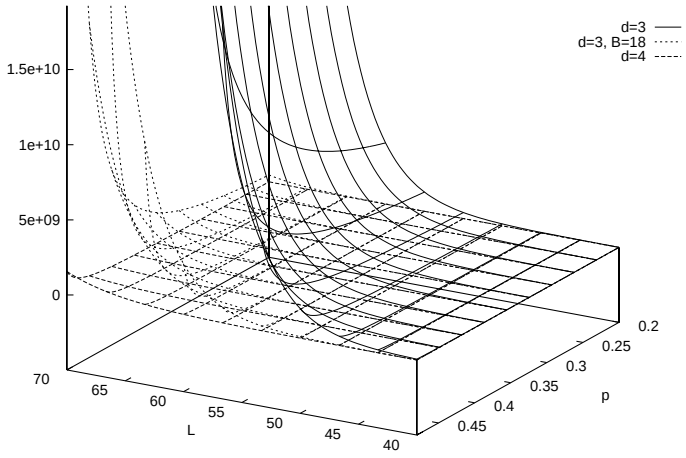


Fig. 6. Number of bits of s required by Gallager algorithm and by Viterbi algorithm

attack with $d=4$ and the attack using a convolutional code which was presented in [6]($d=3$). Let N be the minimum number of bits of the running-key which are required by our attack for a given value of p. The correlation attack using Viterbi algorithm only succeeds for these values of N and p when $m_B(3) \geq K'm(4)$, i.e., $B \geq \log_2(N)$ since $K' \simeq 3$. This high value of B makes the complexity of the decoding step with Viterbi algorithm higher than for our attack: the number of operations required for decoding is multiplied by $B+\frac{L}{10}$. Moreover, the memory requirement makes the decoding step intractable for large values of B (B can not exceed 20 or 30 in practice). The only advantage of the attack based on convolutional codes is the lower complexity of the preprocessing step: in our attack the number of operations performed for finding the parity-check equations is multiplied by $\frac{N}{2}$. But this part of the attack is performed once for all while the decoding step should be repeated for each new initialization of the system. A similar comparison can be made for higher weight parity-check equations. Note that, as pointed out by Formula (6), the advantage of increasing

the memory B in the attack based on convolutional codes decreases for higher values of d. Moreover, the value of d in the attack based on convolutional codes is limited due to the time complexity of Viterbi algorithm. Gallager algorithm should therefore be preferred in most situations.

A recent improvement [7] of the attack based on convolutional codes consists in using M parallel convolutional codes with memory B which all share the same information bits. This does not strongly modify the results of the previous comparison. When a turbo code is used, the number of operations performed by the decoding procedure is $6M2^B m_B(d)(L+9B)$ and the memory requirement is roughly the same than in Viterbi algorithm. The processing step now performs around $(L+9B)M\frac{N^{d-2}}{(d-2)!}$ operations.

6 Simulation results

We now present some simulation results of our attack based on a LFSR of length $L = 40$ with feedback polynomial $P(X) = 1 + X + X^3 + X^5 + X^9 + X^{11} + X^{12} + X^{17} + X^{19} + X^{21} + X^{25} + X^{27} + X^{29} + X^{32} + X^{33} + X^{38} + X^{40}$. This polynomial was used for all simulations in [12, 6, 7]. The results obtained by Gallager algorithm with parity-check equations of weight 4 and 5 are presented in Figure 7.

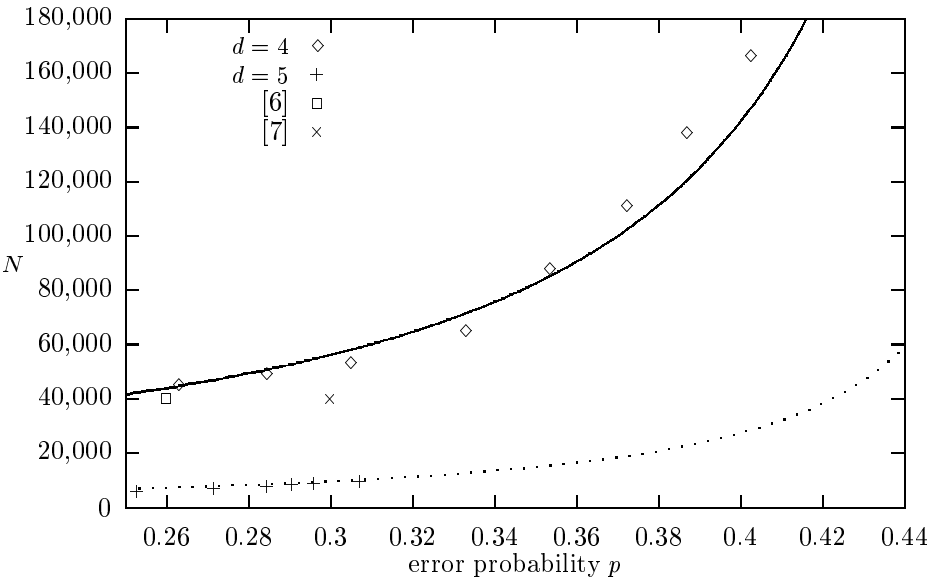


Fig. 7. Number of bits of s required for a fast correlation attack (L=40)

As an example, the following table compares the maximum error-probabilities achieved by the different correlation attacks when $N = 400,000$ bits of s are known:

	our attack $d = 4$	[6] (Viterbi) $d = 3, B = 15$	[7] (turbo) $d = 3, M = 8, B = 13$
maximum error probability p	0.44	0.40	0.41

For $N = 400,000$ and $p = 0.44$, the preprocessing step and the decoding step of our attack took respectively 9 hours and 1.5 hour on a DEC alpha workstation. Note that the attack based on convolutional codes with $d = 4$ and $B = 16$ can achieve $p = 0.482$, but it requires 2^{53} operations. This error-probability can be achieved by our attack with $d = 5$ and with only $N = 360,000$ bits of s. In this case, the number of operations required by the decoding step is 2^{52}.

Similarly, the correlation attack based on turbo codes achieves $p = 0.3$ for $40,000$ known bits of s (with $B = 15$ and $M = 16$) [7]. We here correct the same error-probability with only $9,770$ bits using parity-check equations of weight 5. In this last case, the preprocessing step takes roughly 30 hours, and the decoding step takes 12 seconds.

7 Conclusions

We have shown that the fast correlation attacks using Gallager iterative decoding algorithm with parity-check equations of weight 4 or 5 are more efficient than the attacks based on convolutional codes or on turbo codes. The performance of our algorithm is only limited by the time complexity of the preprocessing step; however, it is important to note that this part of the attack has to be performed once for all. The different techniques proposed by Johansson and Jönsson could also use higher-weight parity-check equations but the induced improvement is strongly limited by the memory requirement of the decoding procedure. Gallager algorithm should therefore be preferred in most situations. The previous theoretical analysis provides all necessary choices of parameters for practical implementations.

References

1. V. Chepyzhov and B. Smeets. On a fast correlation attack on certain stream ciphers. In *Advances in Cryptology - EUROCRYPT'91*, number 547 in Lecture Notes in Computer Science, pages 176–185. Springer-Verlag, 1991.
2. E. Filiol and C. Fontaine. Highly nonlinear balanced Boolean functions with a good correlation-immunity. In *Advances in Cryptology - EUROCRYPT'98*, number 1403 in Lecture Notes in Computer Science, pages 475–488. Springer-Verlag, 1998.
3. R.G. Gallager. Low-density parity-check codes. *IRE Trans. Inform. Theory*, IT-8:21–28, 1962.
4. R.G. Gallager. *Low-density parity-check codes.* MIT Press, Cambridge, MA, 1963.

5. J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inform. Theory*, 42(2):429–445, 1996.
6. T. Johansson and F. Jönsson. Improved fast correlation attack on stream ciphers via convolutional codes. In *Advances in Cryptology - EUROCRYPT'99*, number 1592 in Lecture Notes in Computer Science, pages 347–362. Springer-Verlag, 1999.
7. T. Johansson and F. Jönsson. Fast correlation attacks based on turbo code techniques. In *Advances in Cryptology - CRYPTO'99*, number 1666 in Lecture Notes in Computer Science, pages 181–197. Springer-Verlag, 1999.
8. F.J. MacWilliams and N.J.A. Sloane. *The theory of error-correcting codes*. North-Holland, 1977.
9. W. Meier and O. Staffelbach. Fast correlation attacks on stream ciphers. In *Advances in Cryptology - EUROCRYPT'88*, number 330 in Lecture Notes in Computer Science, pages 301–314. Springer-Verlag, 1988.
10. W. Meier and O. Staffelbach. Fast correlation attack on certain stream ciphers. *J. Cryptology*, pages 159–176, 1989.
11. W. Meier and O. Staffelbach. Nonlinearity criteria for cryptographic functions. In *Advances in Cryptology - EUROCRYPT'89*, number 434 in Lecture Notes in Computer Science, pages 549–562. Springer-Verlag, 1990.
12. M.J. Mihaljević and J.D. Golić. A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence. In *Advances in Cryptology - AUSCRYPT'90*, number 453 in Lecture Notes in Computer Science, pages 165–175. Springer-Verlag, 1990.
13. M.J. Mihaljević and J.D. Golić. A comparison of cryptanalytic principles based on iterative error-correction. In *Advances in Cryptology - EUROCRYPT'91*, number 547 in Lecture Notes in Computer Science, pages 527–531. Springer-Verlag, 1991.
14. W. T. Penzhorn. Correlation attacks on stream ciphers: computing low-weight parity checks based on error-correcting codes. In *Fast Software Encryption 96*, number 1039 in Lecture Notes in Computer Science, pages 159–172. Springer-Verlag, 1996.
15. W.T. Penzhorn and G.J. Kühn. Computation of low-weight parity checks for correlation attacks on stream ciphers. In *Cryptography and Coding - 5th IMA Conference*, number 1025 in Lecture Notes in Computer Science, pages 74–83. Springer-Verlag, 1995.
16. R.A. Rueppel. *Analysis and Design of stream ciphers*. Springer-Verlag, 1986.
17. E.S. Selmer. *Linear recurrence relations over finite fields*. PhD thesis, University of Bergen, Norway, 1966.
18. C.E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27, 1948.
19. T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Trans. Inform. Theory*, IT-30(5):776–780, 1984.
20. T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Computers*, C-34(1):81–84, 1985.
21. K. Zeng, C.H. Yang, and T.R.N. Rao. An improved linear syndrome algorithm in cryptanalysis with applications. In *Advances in Cryptology - CRYPTO'90*, number 537 in Lecture Notes in Computer Science. Springer-Verlag, 1990.
22. N. Zierler. Linear recurring sequences. *J. Soc. Indus. Appl. Math.*, 7:31–48, 1959.

Advanced Slide Attacks

Alex Biryukov* and David Wagner**

Abstract. Recently a powerful cryptanalytic tool—the slide attack—was introduced [3]. Slide attacks are very successful in breaking iterative ciphers with a high degree of self-similarity and even more surprisingly are independent of the number of rounds of a cipher. In this paper we extend the applicability of slide attacks to a larger class of ciphers. We find very efficient known- and chosen-text attacks on generic Feistel ciphers with a periodic key-schedule with four independent subkeys, and consequently we are able to break a DES variant proposed in [2] using just 128 chosen texts and negligible time for the analysis (for one out of every 2^{16} keys). We also describe *known-plaintext* attacks on DESX and Even-Mansour schemes with the same complexity as the best previously known *chosen-plaintext* attacks on these ciphers. Finally, we provide new insight into the design of GOST by successfully analyzing a 20-round variant (GOST$\oplus$) and demonstrating weak key classes for all 32 rounds.

1 Introduction

The *slide attack* is a powerful new method of cryptanalysis of block-ciphers introduced in [3]. The unique feature of this new cryptanalytic attack is its independence of the number of rounds used in the cipher of interest: when a slide attack is possible, the cipher can be broken no matter how many rounds are used. This capability is indispensable in a study of modern iterative block ciphers and hash functions. As the speed of computers grows, it is natural to use more and more rounds, which motivates our study of attacks that are independent of the number of rounds. While addition of a few rounds usually stops even a very sophisticated cryptanalytic attack (such as a differential or linear attack), in contrast a cipher vulnerable to slide attacks cannot be strengthened by increasing the number of its rounds. Instead, one must change the key-schedule or the design of the rounds.

In [3] it was shown that slide attacks exploit the degree of *self-similarity* of a block cipher and thus are applicable to iterative block-ciphers with a periodic key-schedule. It was also shown that slide attacks apply to auto-key ciphers (where the choice of the round subkeys is data-dependent). As an example an attack was presented on modified Blowfish [17], a cipher based on key-dependent S-boxes which so far had resisted all the conventional attacks.

* Applied Mathematics department, Technion - Israel Institute of Technology, Haifa, Israel 32000, and Computer Science department, The Weizmann Institute of Science, Rehovot 76100, Israel. Email: albi@wisdom.weizmann.ac.il

** University of California, Berkeley. Email: daw@cs.berkeley.edu

Table 1. Summary of our attacks on various ciphers.

Cipher	(Rounds)	Key bits	Best Previous Attack			Our Attack		
			Data	Type	Time	Data	Type	Time
2K-DES	(∞)	96	2^{32}	KP	2^{50}	2^{32}	KP	2^{33}
2K-DES	(∞)	96	2^{32}	KP	2^{50}	2^{17}	CP/CC	2^{17}
4K-Feistel	(∞)	192	—	–	—	2^{32}	KP	2^{33}
4K-Feistel	(∞)	192	—	–	—	2^{17}	CP/CC	2^{17}
4K-DES*	(∞)	192	—	–	—	2^{17}	CP/CC	2^{17}
Brown-Seberry-DES*	(∞)	56	—	–	—	128	CP/CC	2^{7}
DESX	(16)	184	2^{m}	CP	2^{121-m}	$2^{32.5}$	KP	$2^{87.5}$
DESX	(16)	184	2^{m}	CP	2^{121-m}	$2^{32.5}$	CO	2^{95}
Even-Mansour	(—)	$2n$	$2^{n/2}$	CP	$2^{n/2}$	$2^{n/2}$	KP	$2^{n/2}$
GOST$\oplus$	(20)	256	—	–	—	2^{33}	KP	2^{70}

CO — ciphertext-only, KP — known-plaintext, CP — chosen-plaintext, CP/CC — chosen plaintext/ciphertext. * – Our attack on 4K-DES and Brown-Seberry-DES works for $1/2^{16}$ of all keys. Note that attacks on 2K-DES work for all the keys.

The existence of attacks which are independent of the number of rounds is perhaps counter-intuitive. To illustrate this consider a quote from [15]:

> "Except in a few degenerate cases, an algorithm can be made arbitrarily secure by adding more rounds."

Slide attacks force us to revise this intuition, and this motivates our detailed study of advanced sliding techniques.

In this paper we introduce advanced sliding techniques—*sliding with a twist* and the *complementation slide*—that result in a more efficient slide attacks and allow to attack new classes of ciphers. We illustrate these techniques on generic Feistel constructions with two- or four-round self-similarity as well as a Luby-Rackoff construction and also the example ciphers 2K-DES and 4K-DES, which differ from DES only by having 64 rounds, a 96- or 192-bit key, and a simplified (periodic) key-schedule. Analysis of these ciphers is of independent interest since it demonstrates the dangers of some ways to extend DES. Specifically we show a very efficient attack on a variant of DES proposed in [2]: our attack uses only 128 chosen texts and negligible time of analysis (for a 2^{-16} fraction of all keys).

We then apply the newly developed methods to the DESX and Even-Mansour schemes, and we show known-plaintext slide attacks with the same complexity as the best previously known chosen-plaintext attacks. We also apply slide attacks to the GOST cipher (a Russian equivalent of DES) obtaining insights on its design.

See Table 1 for a summary of our results. For each cipher a number of rounds that our attack is able to cover is presented; ∞ is shown if our attack is independent of the number of rounds of a cipher. The block size in bits is denoted by n, and the 'Key bits' column denotes the number of secret key bits of the cipher.

This paper is organized as follows: In Section 2 we briefly describe conventional slide attacks. We develop several advanced sliding techniques in Section 3,

illustrating them on generic Feistel ciphers with periodic key-schedules. As a side effect we receive a distinguishing attack on the $\Psi(f,g,f,g,\ldots,f,g)$ Luby-Rackoff construction (see the end of Section 3.2). We then apply the newly developed techniques to the analysis of DESX and Even-Mansour schemes in Section 4. In Section 5 we turn advanced slide attacks to the analysis of GOST. Finally Section 6 summarizes some related work and Section 7 outlines some possible directions for further research.

2 Conventional Slide Attacks

Earlier work [3] described a simple form of slide analysis applicable to ciphers with self-similar round subkey sequences or autokey ciphers. We briefly sketch those ideas here; see [3] for full details and cryptanalysis of a number of ciphers, and Section 6 for other related work.

In the simplest case, we have an r-round cipher E whose rounds all use the same subkey, so that $E = F \circ F \circ \cdots \circ F = F^r$. Note that if the key schedule of a cipher is periodic with period p, we can consider F to be a "generalized" round consisting of p rounds of the original cipher. We call such ciphers **p-round self-similar**. Let $\langle P, C\rangle$ be a known plaintext-ciphertext pair for E. The crucial observation is

$$P' = F(P) \quad \text{implies} \quad C' = E(P') = F^r(F(P)) = F(F^r(P)) = F(C).$$

In a standard slide attack, we try to find pairs $\langle P, C\rangle$, $\langle P', C'\rangle$ with $P' = F(P)$; we call such a pair a **slid pair**, and then we will get the extra relation $C' = F(C)$ "for free."

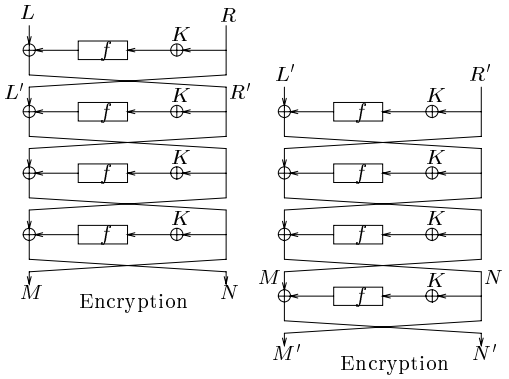


Fig. 1. A conventional slide attack on a generic Feistel cipher with one-round self-similarity. If $L' = R$ and $R' = L \oplus f(K \oplus R)$, the texts shown above will form a slid pair, and we will have $M' = N$ and $N' = M \oplus f(K \oplus N)$.

Slide attacks provide a very general attack on iterated product ciphers with repeating round subkeys. The only requirement on F is that it is very weak

against known-plaintext attack with two pairs (we are able to relax this requirement later, in Section 3.5). More precisely, we call $F_k(x)$ a **weak** permutation if given the two equations $F_k(x_1) = y_1$ and $F_k(x_2) = y_2$ it is "easy" to extract the key k. Such a cipher (with a n-bit block) can be broken with only $2^{n/2}$ known texts, since then we obtain 2^n possible pairs $\langle P, C\rangle$, $\langle P', C'\rangle$; as each pair has a 2^{-n} chance of forming a slid pair, we expect to see one slid pair which discloses the key.

Feistel ciphers form an important special case for sliding, since the attack complexity can be substantially reduced from the general case. We depict in Figure 1 a conventional slide attack on a Feistel cipher with repeating round subkeys. The Feistel round structure gives us an n-bit filtering condition on slid pairs, which lets us reduce the complexity of analysis to about $2^{n/2}$ time and space, a significant improvement over the 2^n work required for the general attack listed above. Furthermore, there is a chosen-text variation which works against Feistel ciphers with about $2^{n/4}$ chosen plaintexts: we may simply use structures to 'bypass the first round'. See [3] for details.

In this paper, we focus on generalizing the slide attack to apply to a broader range of constructions.

3 Advanced Sliding Techniques

In this section we show several ways of extending the basic slide attack to apply to larger classes of ciphers. In the following subsections we introduce two new methods: the *complementation slide* and *sliding with a twist*.

We will describe these new techniques by applying them first to a generic Feistel cipher with a 64-bit block and self-similar round subkeys. (See Figure 1 for an example of such a cipher, where the subkeys exhibit one-round self-similarity. In this section, we consider up to four-round self-similarity.) For ease of illustration we will show graphically ciphers with only a small number of rounds, but we emphasize that the attacks described in this section apply to ciphers with any number of rounds. After describing the basic attack techniques we will show how to extend them to real ciphers.

3.1 The Complementation Slide

First we show a method to amplify self-similarity of Feistel ciphers with two-round self-similarity by exploiting its complementation properties, thus allowing for much better attacks. We call this approach the *complementation slide*.

In the conventional attack, to deal with two-round self-similarity one must slide by two rounds (thus achieving a perfect alignment of rounds with K_0 and K_1), but this yields inefficient attacks. In contrast, we suggest to slide by only one round. This introduces the difference $\Delta = K_0 \oplus K_1$ between slid encryptions in all the rounds. Notice that we have effectively amplified the self-similarity of the cipher from 2-round to 1-round self similarity. However together with amplified

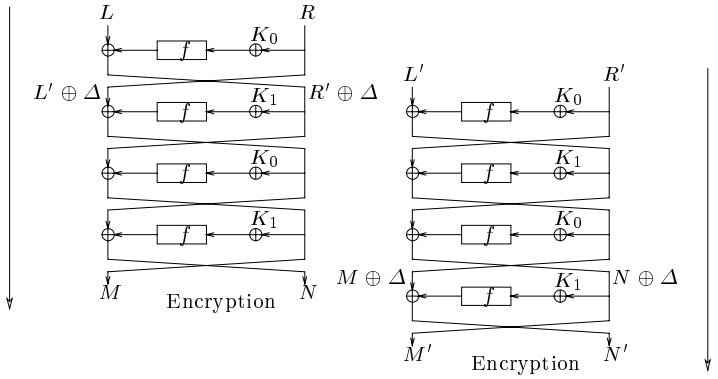


Fig. 2. A *complementation slide* attack on a Feistel cipher with two-round self-similarity. If $L' = R \oplus \Delta$ and $R' = L \oplus f(K_0 \oplus R) \oplus \Delta$, the texts shown above will form a slid pair, and we will have $M' = N \oplus \Delta$ and $N' = M \oplus f(K_1 \oplus N \oplus \Delta) \oplus \Delta$, where $\Delta = K_0 \oplus K_1$.

self-similarity we have introduced differences between rounds of encryption in a slid pair. How can the attack proceed?

Our answer is to choose a slid pair so that the plaintext differences will cancel the difference between the subkeys. Instead of searching for plaintexts with slid difference zero, we search for plaintexts with slid difference $\langle \Delta, \Delta \rangle$. (Note: We say that a pair of plaintexts P, P' has *slid difference* d if $F(P) \oplus P' = d$.) Such a slid difference will propagate with probability one through all the rounds, and thus will appear at the ciphertext. See Figure 2 for a pictorial illustration of the attack.

The slid pairs can be found in a pool of 2^{32} known plaintexts, as before. If we denote the plaintext by $P = \langle L, R \rangle$ and the ciphertext by $C = \langle M, N \rangle$, we get the following slid equations:

$$\langle L', R' \rangle = \langle R, L \oplus f(K_0 \oplus R) \rangle \oplus \langle \Delta, \Delta \rangle$$
$$\langle M', N' \rangle = \langle N, M \oplus f(K_1 \oplus N \oplus \Delta) \rangle \oplus \langle \Delta, \Delta \rangle.$$

Thus we have $L' \oplus M' = R \oplus N$ which is a 32-bit condition on a slid pair. Moreover the second equation suggests a 32-bit candidate for $\Delta = K_0 \oplus K_1$; if we have several slid pairs, this value should coincide for all of them (although we do not need the latter property in our attack). Thus the S/N ratio of this attack is very high. As soon as one slid pair is found, we derive $\Delta = K_0 \oplus K_1$. Then, if the round function f is weak enough, we will be able to derive the keys K_0 and K_1 themselves from the first and second equations. We will only need to examine 2^{31} pairs (due to the 32-bit filtering condition) and each pair suggests at most one candidate key, so the work-factor of the attack is very low.

To summarize, this gives a known plaintext attack on a generic Feistel cipher with two-round self-similarity. The complexity of the attack is quite realistic: we need just 2^{32} known texts and at most 2^{32} light steps of analysis. However, see Section 3.2 for an even better attack.

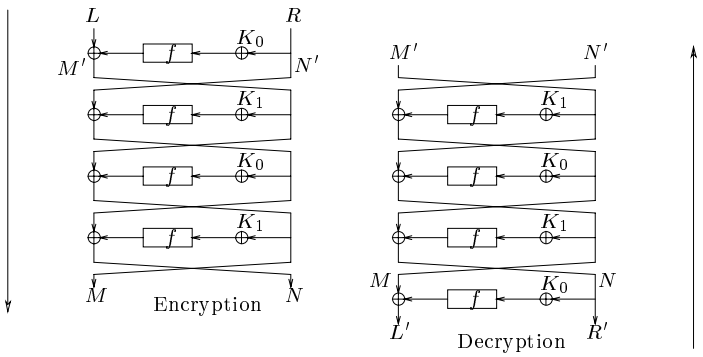


Fig. 3. *Sliding with a twist*, applied to a Feistel cipher with two-round self-similarity. If $N' = R$ and $M' = L \oplus f(K_0 \oplus R)$, the texts shown above will form a (twisted) slid pair, and we will have $R' = N$ and $L' = M \oplus f(K_0 \oplus N)$.

Even more interestingly: We can consider a variant with four independent subkeys, K_0, K_1, K_2, K_3, so that the key size is 128 bits. If we slide by two rounds we find that the XOR differences between subkeys are 2-round self-similar! A modified version of the above attack works, although the S/N ratio is not as high as before. Complementation sliding thus provides a powerful technique for amplifying self-similarity in iterated ciphers.

3.2 Sliding with a Twist

We next describe a novel technique of sliding with a *twist* on a Feistel cipher with two-round self-similarity. This allows for even better attacks than those presented above. See also our attack on DESX in Section 4 for an important application of sliding with a twist.

If we ignore the final swap for the moment, then decryption with a Feistel cipher under key K_0, K_1 is the same as encryption with key K_1, K_0[1]. Of course, Feistel encryption with key K_0, K_1 is very similar to encryption with key K_1, K_0: they are just out of phase by one round. Therefore, we can slide by one round a decryption process against an encryption process (the *twist*). This provides us with a slid pair with an overlap of all rounds except for one round at the top and one round at the bottom. Notice that due to the twist these rounds both use the same subkey K_0. See Figure 3 for a graphical depiction.

The attack begins by obtaining a pool of 2^{32} known texts, so that we expect to find one slid pair. For a slid pair, we have

$$\langle M', N' \rangle = \langle L \oplus f(K_0 \oplus R), R \rangle \qquad \langle L', R' \rangle = \langle M \oplus f(K_0 \oplus N), N \rangle$$

which gives us a 64-bit filtering condition on slid pairs (namely $N' = R$ and $R' = N$). Thus the slid pair can be easily found with a hash table and 2^{32} work, and it immediately reveals the subkey K_0.

[1] In [3] such cipher, based on DES was called 2K-DES.

The rest of the key material can be obtained in a second analysis phase with a simplified conventional sliding (by two rounds and without a twist) using the same pool of texts and with less than 2^{32} work. Pick a ciphertext from a pool, partially encrypt it with K_0 and search the pool of ciphertexts for one with coinciding 32 bits. If such a ciphertext is found perform a similar check on their plaintexts. If both conditions hold this is a slid pair that provides us with K_1. This attack requires just 2^{32} known texts and 2^{33} work.

Moreover, there is a chosen-plaintext/ciphertext variant that allows us to reduce the number of texts down to 2^{17} with the use of structures. We generate a pool of 2^{16} plaintexts of the form (L_i, R) and obtain their encryptions. Also, we build a pool of 2^{16} ciphertexts of the form (M'_j, N') and decrypt each of them, where the value $N' = R$ is fixed throughout the attack. This is expected to give one slid pair, and then the analysis proceeds as before.

This demonstrates that sliding with a twist is capable of attacking any n-bit Feistel block cipher with a two-round periodic key-schedule with $2^{n/2}$ known plaintexts and about $2^{n/2}$ time, or with about $2^{n/4}$ chosen plain-ciphertexts and about $2^{n/4}$ time. Also, sliding with a twist can be used to distinguish a Luby-Rackoff [13] construction with two alternating pseudo-random functions f and g and with an arbitrary number of rounds (an accepted notation is $\Psi(f, g, f, g, \ldots, f, g)$) from a random permutation with about $2^{n/2}$ known plaintexts and similar time (given that the block size is n bits), or with about $2^{n/4}$ chosen plaintext/ciphertext queries and similar time.

3.3 Better Amplification of Self-Similarity: Four-Round Periodicity

In this section we combine the *complementation slide* and *sliding with a twist* to amplify the self-similarity of round subkeys even further. Consider a Feistel cipher with key schedule that repeats every four rounds, using independent subkeys K_0, K_1, K_2, K_3, and suppose these keys are XORed at the input of the f-function. We call this generic cipher a *4K-Feistel* cipher.

One may naively slide by two rounds to amplify self-similarity, like this:

$$\begin{array}{l} K_0\ K_1\ K_2\ K_3\ K_0\ K_1\ \ldots \\ \qquad K_0\ K_1\ K_2\ K_3\ K_0\ K_1\ \ldots \end{array}$$

Then one may use a complementation slide technique using the slid difference $\langle K_1 \oplus K_3, K_0 \oplus K_2 \rangle$. However, there doesn't seem to be any way to make this attack work with less than $2^{n/2}$ texts, and the analysis phase is hard.

Better results are possible if one applies *sliding with a twist*. At a first glance, the twist may not seem to be applicable, but consider combining it simultaneously with the *complementation slide*, like this:

$$\begin{array}{l} K_0\ K_1\ K_2\ K_3\ K_0\ K_1\ K_2\ K_3\ K_0\ \ldots \\ \quad K_3\ K_2\ K_1\ K_0\ K_3\ K_2\ K_1\ K_0\ K_3\ \ldots \end{array}$$

The top row represents an encryption, and the bottom represents a decryption (or, equivalently, encryption by K_3, K_2, K_1, K_0, due to the similarity between encryption and decryption in Feistel ciphers).

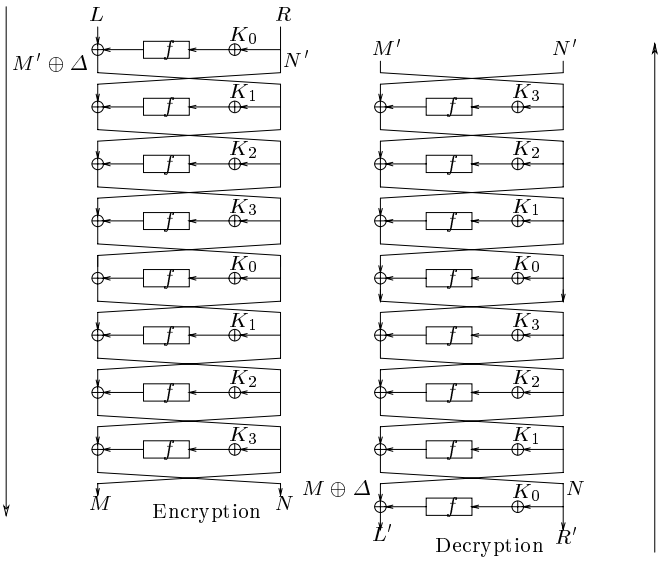


Fig. 4. Combining the complementation slide and sliding with a twist techniques in a single unified attack against a Feistel cipher with four-round self-similarity.

Now note that the odd rounds always line up, but the even rounds have the constant difference $K_1 \oplus K_3$ in the round subkeys. Therefore, we can apply the *complementation slide* technique, if we can get texts with a slid difference of $\langle 0, K_1 \oplus K_3 \rangle$. Then we get the attack shown in Figure 4.

Combining the two advanced sliding techniques provides a number of significant benefits. First, we obtain an n-bit filtering condition, so detecting slid pairs becomes easy. Consequently, the analysis phase is straightforward. Also, the combined approach makes it easier to recover key material from a slid pair. Finally, perhaps the most important improvement is that now we can reduce the data complexity of the attack to just $2^{n/4}$ texts, in the case where chosen-plaintext/ciphertext queries are allowed. Neither advanced sliding technique can—on its own—provide these advantages; in this respect, the whole is greater than the sum of the parts.

3.4 Attack on DES with Brown-Seberry Key-schedule

In [2] an alternative key-schedule for DES was proposed. This key-schedule was supposed to be "as effective as that used in the current DES" and was "suggested for use in any new algorithm" [2]. This variant of DES was already studied in [1] resulting in a related-key attack on it. In this section we show a chosen plaintext/ciphertext slide attack on this variant of DES, which uses only 128 chosen texts and negligible time for analysis. The attack works for 2^{40} out of 2^{56} keys.

To remind the reader: the DES key-schedule consists of two *permuted-choice* permutations PC1 and PC2, and a rotation schedule. The first permuted choice PC1 is used to reduce the key-size from 64 bits to 56 bits. Then the result is divided into two 28-bit registers C and D. Each round we cyclicly rotate both registers by one or two bits to the left. Permuted choice PC2 is applied to the result, which picks 24 bits from each 28-bit register and thus forms a 48-bit round subkey.

In [2] a key-schedule that rotates by 7 bits every round was proposed (instead of the irregular 1,2-bit rotations used in DES). Due to a larger rotation amount which spreads bits between different S-boxes the PC2 permutation was simplified to become an identity permutation which just discards the last 4 bits of each 28-bit register. We claim that for $1/2^{16}$ of the keys, this variant can be broken with our sliding with a twist techniques as follows: the known-plaintext attack will require $2^{32.5}$ texts, time and space; the chosen-plaintext/ciphertext, however, will require only 2^7 texts!

First of all notice that since the new rotation amount (7 bits) divides the size of the key-schedule registers (28 bits) the registers C, D return to their original state every four rounds. This results in a key-schedule with a period of four, which can be analyzed by the methods that we developed in the previous sections for the four-round self-similar Feistel ciphers. We will extend the standard attack even further by noticing that DES key-schedule is used and not four independent round subkeys as in our previous model. However, DES-like ciphers introduce one small complication: the DES round function XORs the subkey against the 48-bit expanded input rather than the raw 32-bit input, so the complementation slide only works if the 48-bit subkey difference is expressible as the expansion of some 32-bit text difference.

Let $J_i = \langle C \lll 7i, D \lll 7i \rangle$ so that $K_i = \mathrm{PC2}(J_i)$. For the sliding with a twist to work in the case of DES we need $K_1 \oplus K_3$ to have an 'expandable' form in order to pass through the 32 to 48 expansion of the DES round function. Note also that if $J_1 = \langle u, v, u', v' \rangle$ where u, v, u, v' are all 14-bit quantities, then $J_3 = \langle v, u, v', u' \rangle$ in a Brown-Seberry key-schedule, and thus for $Z = J_1 \oplus J_3$ we have $Z_i = Z_{i+14}$ for $i \in \{0, 1, \ldots, 13, 28, 29, \ldots, 41\}$. The PC2 just discards Z_i for $i \in \{24, 25, \ldots, 27, 52, 53, \ldots, 55\}$ to get the 48-bit quantity $Y = \mathrm{PC2}(Z) = K_1 \oplus K_3$.

If we insist $Y = \mathrm{Expansion}(X)$ for some X, we get 16 constraints on Y: namely, $Y_i = Y_{i+2}$ for $i = 6j + k$, $j \in \{0, \ldots, 7\}$, $k \in \{4, 5\}$ where subscripts are taken modulo 48. Thus we have

$$Z_i = Z_{i+2} \text{ for } i \in \{4, 5, 10, 11, 16, 17, 32, 33, 38, 39, 44, 45\};$$

and $Z_i = Z_{i+6}$ for $i \in \{22, 23, 50, 51\}$. Therefore $Y = K_1 \oplus K_3$ is expandable if and only if $Z = J_1 \oplus J_3$ has the form

$$Z = \langle abcdcd\ cdefgh\ ghabcd\ cdcdef\ ghgh\ \ efklkl\ klabmn\ mnefkl\ klklab\ mnmn \rangle$$

where $a, b, .., n$ are 12 arbitrary bits. we see that there are exactly 2^{12} expandable values of $K_1 \oplus K_3$ that satisfy the required constraints. Moreover, for each expandable value of $K_1 \oplus K_3$, there are 2^{28} possible values of J_1 for which $K_1 \oplus K_3$

has the given value (since we may choose u and u' arbitrarily, setting v and v' as required to ensure that $\langle u \oplus v, u \oplus v, u' \oplus v', u' \oplus v' \rangle$ has an appropriate value for $J_1 \oplus J_3$).

This shows that there are 2^{40} values of J_1 that lead to four-round self-similarity with an expandable value for $K_1 \oplus K_3$. In other words, $1/2^{16}$ of the keys are breakable with our standard attack. Note that the standard attack for the case of four independent round subkeys uses $2^{32.5}$ known texts, time and space, or 2^{17} chosen texts, time and space. However, we may use the special structure of $K_1 \oplus K_3$ to significantly reduce the complexity of the chosen-text attack.

In particular, we choose 2^6 plaintexts of the form $\langle L_i, R \rangle$ and 2^6 ciphertexts of the form $\langle M'_j, N' \rangle$, where $R = N'$ is fixed throughout the attack and

$$\begin{aligned} L_i &= \langle bcdc\ def0\ 0abc\ dcde\ f000\ 0ab0\ \ 0ef0\ \ 000a \rangle \\ M'_j &= \langle 0000\ 000g\ \ h000\ 0000\ 0klk\ \ l00m\ n00k\ \ lkl0 \rangle, \text{ so that} \\ L_i \oplus M'_j &= \langle bcdc\ \ defg\ habc\ dcde\ \ fklk\ \ labm\ \ nefk\ lkla \rangle \end{aligned}$$

and thus $\text{Expansion}(L_i \oplus M'_j) = K_1 \oplus K_3$ for some i, j, which immediately gives us a slid pair. (We assume for ease of description that the cipher includes the final swap and no IP or FP, so that Figure 4 in Section 3.2 applies.) We can recognize the slid pair by a 64-bit filtering condition on $\langle M, N \rangle, \langle L', R' \rangle$, and so the analysis phase is easy.

To sum up, this provides an attack on the cipher that breaks $1/2^{16}$ of the keys with 2^7 chosen texts, time and space.

3.5 Generalizations for a Composition of Stronger Functions

In Section 2 we have seen how a typical slide attack may work. However, in many cases this approach is too restrictive, since it may be desirable to analyze ciphers which decompose into a product of stronger functions; in particular, the round function may be strong enough that multiple input/output pairs are required to recover any key material. In this section we show several techniques to handle this situation.

One approach is to use a differential analysis. Denote by n the block size of the cipher. Suppose there is a non-trivial differential characteristic $\Delta X \to \Delta Y$ of probability p for the round function. We associate to each plaintext P the plaintext $P \oplus \Delta X$ and to each plaintext P' another plaintext $P' \oplus \Delta Y$. Then, if $P' = F(P)$, we will also have $P' \oplus \Delta Y = F(P \oplus \Delta X)$ with probability p (thanks to the characteristic $\Delta X \to \Delta Y$), which provides two slid pairs. In this way we may obtain four known input/output pairs for the function F. We can generate a set of $3 \cdot 2^{n/2} p^{-1/2}$ chosen plaintexts such that for plaintext P in the chosen set the plaintexts $P \oplus \Delta X$ and $P \oplus \Delta Y$ are also in the set; then we will expect to see one pair P, P' satisfying both the slide and the differential patterns.

The second approach (which is probably the simplest) works like this. Suppose to recover the key we need N known texts for the round function F. For

each plaintext P, we suggest to get the encryption $E(P)$ of P, and the double-encryption $E^2(P) = E(E(P))$ of P, and so on, until we have obtained $E^{2N}(P)$. Then, if $P' = F(E^i(P))$, we find $2N - i$ slid pairs "for free" by the relation $E^j(P') = F(E^{j+i}(P))$ for $j = 1, .., 2N - i$. With $2^{(n+1)/2}N^{1/2}$ chosen texts, we expect to find about N slid pairs in this way (probably all in the same batch formed from a single coincidence of the form $P' = F(E^i(P))$). To locate the batch of slid pairs, one could naively try all 2^{n+2} possible pairings of texts (though in practice we would search for a more efficient approach); each pairing that gives N or more known texts for F will suggest a key value that can then be tested[2].

Normally this last attack would be classified as an adaptive chosen-plaintext attack. However, note that in many modes (CBC, CFB) it can be done with a non-adaptive chosen-plaintext attack. Furthermore, in the case of OFB mode, a known plaintext assumption suffices. However, these comments assume that re-encryption preserves the sliding property, which is not always the case.

Another possible generalization is in the case of Feistel-ciphers. In this case one can detect slid pairs even before trying to find the correct secret key k. In the case of a balanced Feistel cipher with block size n we have an $n/2$-bit condition on the ciphertexts of a slid pair. This increases the S/N ratio considerably, filtering out most of the incorrect pairs even before we start the analysis. This property allows an attacker to accumulate sufficient number of slid pairs before he starts an attack on a round-reduced variant of a cipher.

Notice also that if we use a technique for receiving many slid pairs in the case of a Feistel-cipher, we would need only $2 \cdot 2^{n/4}N$ chosen texts, and the S/N ratio will be excellent by comparing several halves of the ciphertexts.

Furthermore if $N^{1/2} > 2^{n/4}$, an absolutely different idea can be used. Choose a random starting point P. About $2^{n/2}$ times iterate the following operation $s \circ E$, where s denotes swap of the halves (the swap is needed only if E has no final swap at the last round). This way one can obtain more than $2^{n/2-\log r}$ slid pairs (here r denotes the number of rounds of a cipher). The S/N ratio is again excellent. The idea is that we essentially search for a symmetric point (A, A) of a round function, which happens after about $2^{n/2}$ rounds ($2^{n/2-\log r}$ encryptions). This does not necessarily happen in the middle of a cipher, so we may have to perform up to r times more encryptions before we reach a fixed point for E. In half of the cases (if the first symmetric point happened at an even round) we will receive an orbit "slidable" by two rounds, and in other half of the cases (symmetric point at odd rounds) an orbit will be "slidable" by one round. Even if an orbit is "slidable" only by two, and thus $n/2$-bit filtration will be unreachable to us, the encryption fixed point that ends our orbit helps us slide the orbit correctly (at most $r/2$ possibilities).

[2] If E were behaving like a random function, it would be enough to take $2^{n/2} + N$ encryptions, from an orbit of some arbitrarily chosen element P, but since E is expected to behave like a random permutation, an orbit of P will be a part of usually a very large cycle, leaving no place for collisions. Considering a few more orbits will not help either.

4 Cryptanalysis of DESX and Even-Mansour Schemes

DESX is an extension of DES proposed by Rivest in 1984. It makes DES more resistant to exhaustive search attacks by XORing two 64-bit keys: one at the input and another at the output of the DES encryption box[3]. See [10, 16] for theoretical analysis of DESX.

In this section we show the unexpected result that the DESX construction contains just enough symmetry to allow for slide attacks. These results are actually generally applicable to all uses of pre- and post-whitening (when applied using self-inverse operations like XOR), but for convenience of exposition we will focus on DESX.

The attacks presented here are another example of an application of the powerful new *sliding with a twist* technique. Our attacks on DESX are significantly better than the best previously known attacks: we need just $2^{32.5}$ *known* texts and $2^{87.5}$ time for the analysis, while the best generic attack reported in the literature is a *chosen*-plaintext attack with comparable complexity [10, 16][4]. Thus, sliding techniques allow one to move from the chosen-text attack model to the more realistic known-text attack model. Even more unexpectedly, our attack can also be converted to a ciphertext-only attack.

We briefly recall the definition of DESX. Let $E_k(x)$ denote the result of DES-encrypting the plaintext x under the key k. Then we define DESX encryption under the key $K = \langle k, k_x, k_y \rangle$ as $EX_K(p) = k_y \oplus E_k(p \oplus k_x)$. To set up the necessary slide relation, we imagine lining up a DESX encryption against a slid DESX decryption, as shown in Figure 5. More specifically, we say that the two known plaintext pairs $\langle p, c \rangle$ and $\langle p', c' \rangle$ form a slid pair if $c \oplus c' = k_y$. Consequently, for any slid pair, we will have

$$p' = k_x \oplus E_k^{-1}(c' \oplus k_y) = k_x \oplus E_k^{-1}(c)$$

as well as $p = k_x \oplus E_k^{-1}(c')$. Combining these two equations yields $k_x = p \oplus E_k^{-1}(c') = p' \oplus E_k^{-1}(c)$. As a result, we get a necessary property of slid pairs: they must satisfy

$$E_k^{-1}(c) \oplus p = E_k^{-1}(c') \oplus p'. \qquad (*)$$

To get a single slid pair, we obtain $2^{32.5}$ known plaintexts $\langle p_i, c_i \rangle$ and search for a pair which satisfies the sliding condition $(*)$. The pairs can be recognized efficiently with the following technique. We guess the DES key k. Next, we insert $E_k^{-1}(c_i) \oplus p_i$ into a lookup table for each i; alternatively, we may sort the texts by this value. A good slid pair $\langle p, c \rangle, \langle p', c' \rangle$ will show up as a collision in the table. Also, each candidate slid pair will suggest a value for k_x and k_y as above

[3] Note that an idea to use simple keyed transformations around a complex mixing transform goes back to Shannon [18, pp.713].

[4] One may apply differential or linear cryptanalysis to DESX, but then at least 2^{60}–2^{61} texts are needed [11]. In contrast, slide attacks allow for a *generic* attack with a much smaller data complexity.

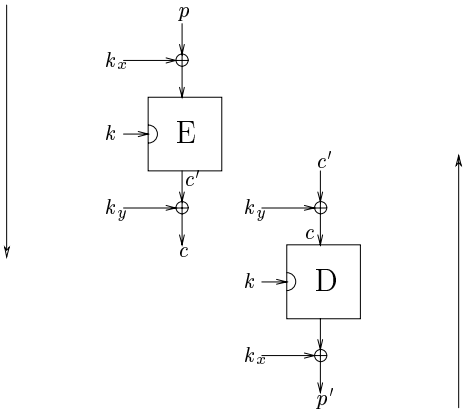


Fig. 5. Sliding with a twist, applied to DESX.

(e.g., $k_y = c \oplus c'$ and $k_x = p \oplus E_k^{-1}(c')$), so we try the suggested DESX key $\langle k, k_x, k_y \rangle$ immediately on a few known texts. With $2^{32.5}$ known texts, we expect to find one false match (which can be eliminated quickly) per guess at k, as well as one correct match (if our guess at k was correct). If this attack sketch is not clear, see the algorithmic description in Figure 6.

In total, the average complexity of our slide attack on DESX is $2^{87.5}$ offline trial DES encryptions, $2^{32.5}$ known texts, and $2^{32.5}$ space. The slide attack is easily parallelized. Compare this to the best attack previously reported in the open literature, which is a *chosen*-plaintext attack that needs 2^{121-m} time (average-case) when 2^m texts are available [10,16]. Therefore, our attack converts the chosen-plaintext assumption to a much more reasonable known-plaintext assumption at no increase in the attack complexity.

CIPHERTEXT-ONLY ATTACKS. Note that in many cases our slide attack on DESX can even be extended to a ciphertext-only attack. We suppose (for simplicity) that most plaintext blocks are composed of just the lowercase letters 'a' to 'z', encoded in ASCII, so that 24 bits of each plaintext are known[5]. For each i we calculate 24 bits of $E_k^{-1}(c_i) \oplus p_i$ and store the result in a lookup table. Due to the weak filtering condition, by the birthday paradox we expect to find about $2^{2 \cdot 32.5-1}/2^{24} = 2^{40}$ collisions in the table. Each collision suggests a value for k_y (as $k_y = c \oplus c'$) and for 24 bits of k_x, which we immediately try with a few DESX trial decryptions on other known ciphertexts. Therefore, for each guess of k the workfactor is 2^{40} DES operations.

This provides a simple ciphertext-only attack needing about $2^{32.5}$ ciphertexts and 2^{95} offline DES operations. The work-factor can be reduced somewhat to 2^{95} simple steps (where each step is much faster than a trial decryption), if 2^{33}

[5] The attack degrades gracefully if our model of the plaintext source is only probabilistic: for instance, if half of the texts follow the model, the attack will need only $\sqrt{2}$ times as many ciphertexts and only twice as much work.

ATTACK:
1. Collect $2^{32.5}$ known plaintexts $\langle p_i, c_i \rangle$.
2. For each $k \in \{0,1\}^{56}$, do
3. Insert $\langle E_k^{-1}(c_i) \oplus p_i, i \rangle$ into a hash table keyed by the first component.
4. For each $i \neq j$ with $E_k^{-1}(c_i) \oplus p_i = E_k^{-1}(c_j) \oplus p_j$, do
5. Set $k_y = c_i \oplus c'_j$ and $k_x = p_i \oplus E_k^{-1}(c_i \oplus k_y)$.
6. Test the validity of the guessed key $\langle k, k_x, k_y \rangle$ on a few more known texts.

Fig. 6. The DESX slide attack, in full detail. It is clear that—once discovered—the attack may be described without reference to sliding, but the *sliding with a twist* methodology made it possible to find the attack in the first place.

known ciphertexts are available, by considering candidate slid pairs two at a time and filtering on the suggested value of k_y, since then the correct value of k_y will be suggested at least twice and can therefore be recognized in this way before doing any trial decryptions. Note that these ciphertext-only attacks are applicable not only to ECB mode but also to most of the standard chaining modes, including CBC and CFB modes.

CRYPTANALYSIS OF THE EVEN-MANSOUR SCHEME. In [7], Even and Mansour studied a simple n-bit block cipher construction based on a fixed pseudo-random permutation and keyed n-bit XORs at the input and at the output. Due to the generic nature of our previous attack on DESX it can also be used to analyze the Even-Mansour construction[6]. In the case of Even-Mansour we replace E_k with an unkeyed mixing transformation E on n-bit blocks, so our slide attack succeeds with just $2^{(n+1)/2}$ known plaintexts and $2^{(n+1)/2}$ work. This provides a known-plaintext attack with the same complexities as the best previously-known chosen plaintext attack [6] and within a factor of $\sqrt{2}$ away from the Even-Mansour lower bound.

5 Analysis of GOST

GOST, the Russian encryption standard [19], was published in 1989.[7] Even after considerable amount of time and effort, no progress in cryptanalysis of the standard was made in the open literature except for a brief overview of a GOST structure in [4] and a related key attack in [9]. In this section we apply slide techniques to GOST and thus are able to produce cryptanalytic results that shed some light on its internal structure.

The GOST encryption algorithm is a block cipher with 256-bit keys and a 64-bit block length. GOST is designed as a 32-round Feistel network, with 32-bit round subkeys. See Figure 7 for a picture of one round of GOST.

[6] Of course, these attacks will apply with the same complexity to DESX when the DES key k is known somehow.

[7] It was translated into English in 1993 and since then became well known to open cryptographic community.

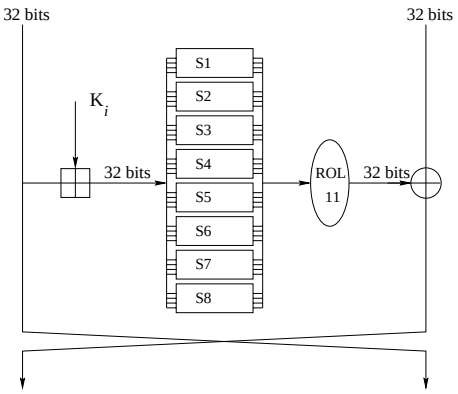


Fig. 7. One round of a GOST cipher.

The key schedule divides the 256-bit key into eight 32-bit words $K_0, \ldots, K_7$, and then uses those key words in the order $K_0, \ldots, K_7$, $K_0, \ldots, K_7$, $K_0, \ldots, K_7$, $K_7, K_6, \ldots, K_0$. Notice the 'twist' in the last 8 rounds.

THE ANALYSIS OF GOST. GOST looks like a cipher that can be made both arbitrarily strong or arbitrarily weak depending on the designer's intent since some crucial parts of the algorithm are left unspecified. A huge number of rounds (32) and a well studied Feistel construction combined with Shannon's substitution-permutation sequence provide a solid basis for GOST's security. However, as in DES everything depends on the exact choice of the S-boxes and the key-schedule. This is where GOST conceptually differs from DES: the S-boxes are not specified in the standard and are left as a secondary key common to a "network of computers"[8].

The second mystery of GOST is its key-schedule. It is very simple and periodic with the period of eight rounds except for the last eight rounds where a twist happens. It is intriguing to find a reason for the twist in the last eight rounds of the key schedule. Moreover, in many applications we may wish to use shorter 64- or 128-bit keys, yet it is not clear how to extend these to a full 256-bit GOST key securely (fill the rest with zeros, copy the bits till they cover 256 bits, copy bits in a reversed order?).

WHY THE TWIST? Consider a GOST cipher with a homogeneous key schedule, i.e., omitting the final twist (let us denote it GOST-H). Is this cipher less secure than GOST? We argue that, if one takes into account the slide attacks, it is. GOST-H can be decomposed into four identical transforms, each consisting of eight rounds of GOST. Furthermore, if one assumes that the round subkey is XORed instead of being ADDed, the cipher will have 2^{128} weak keys of the form $\langle A, B, C, D, A, B, C, D \rangle$ (here each letter represents a 32-bit GOST subkey). These keys are weak since they allow for a *sliding with a twist* attack.

[8] Contrary to common belief, the standard does not even require the S-boxes to be permutations.

There is a known plaintext attack with 2^{32} texts and time, and a chosen plaintext attack with 2^{16} texts and time; see Section 3.3 for more details.

Notice that the 2^{128} keys of the form $\langle A, B, C, D, D, C, B, A\rangle$ are also weak since GOST-H with these keys is an involution and thus double encryption will reveal the plaintext. Since these keys are invariant under a twist the same property holds for GOST itself. Also, there are 2^{32} fixed points for each key of this form, which demonstrates that there may be problems with using GOST to build a secure hash function.

THE ATTACK ON 20 ROUNDS OF GOST$\oplus$. Suppose again that the round subkey is XORed instead of being ADDed, (we will denote this variant of GOST as GOST$\oplus$). Here we show an application of *sliding with a twist* which results in an attack on the last 20 rounds of GOST$\oplus$.

Applying *sliding with a twist*, we get a picture that looks like this:

$$K_4\ K_5\ K_6\ K_7\ K_0\ K_1\ K_2\ K_3\ K_4\ K_5\ K_6\ K_7\ K_7\ K_6\ K_5\ K_4\ K_3\ K_2\ K_1\ K_0$$
$$K_0\ K_1\ K_2\ K_3\ K_4\ K_5\ K_6\ K_7\ K_7\ K_6\ K_5\ K_4\ K_3\ K_2\ K_1\ K_0\ K_7\ K_6\ K_5\ K_4.$$

Let F denote 4 rounds of GOST$\oplus$ with key $K_4, \ldots, K_7$. With a pool of 2^{33} known texts, we expect to find two slid pairs, and each slid pair gives two input/output pairs for F. Breaking F with two known texts is straightforward, and can be performed in time comparable to about 2^9 evaluations of 4-round GOST (equivalent to 2^5 20-round trial encryptions). Thus in our attack we examine all 2^{65} text pairs; each pair suggests a value for 128 bits of key material, which we store in a hash table (or sorted list). The right key will be suggested twice, so we expect to be able to recognize it easily. By the birthday paradox, there will be only about two false matches, and they can be eliminated in the next phase.

Once we have recovered $K_4, \ldots, K_7$, it is easy to learn the rest of the key in a second analysis phase. For example, we can peel off the first four rounds and look for fixed points in the same pool of texts. Since the round subkeys are palindromic in the last sixteen rounds of GOST, there are 2^{32} fixed points, and each has the value $\langle x, x\rangle$ before the last eight rounds of encryption. Thus, given a fixed point, we can try the 2^{32} values of $\langle x, x\rangle$, encrypt forward and backward eight rounds, and obtain two candidate input/output pairs for 4 rounds of GOST$\oplus$ with key $K_0, \ldots, K_3$, so that a value for $K_0, \ldots, K_3$ is suggested after 2^5 work; then the suggested 256-bit key value is tried on another known text pair.

In all, this gives an attack on the last 20 rounds of GOST$\oplus$ that needs 2^{33} known texts, 2^{70} work, and 2^{65} space to recover the entire 256-bit key. Note that this attack is generic and works for any set of (known) S-boxes. The large memory requirements make the attack highly impractical, but we view it as a first step towards a better understanding of the GOST design.

6 Related Work

The first step in the "sliding" direction can be dated back to a 1978 paper by Grossman and Tuckerman [8], which has shown how to break a weakened Feistel

cipher[9] by a chosen plaintext attack, independent of the number of rounds. We were also inspired by Biham's work on related-key cryptanalysis [1], and Knudsen's early work [12].

Some related concepts can be found in Coppersmith's analysis of fixed points in DES weak keys and cycle structure of DES using these keys [5]. This analysis was continued further by Moore and Simmons [14]. For a DES weak key, all round subkeys are constant, and so encryption is self-inverse and fixed points are relatively common: there are precisely 2^{32} fixed points. Note that this property will also be found in any Feistel cipher with *palindromic* round key sequences, so the slide attack is not the only weakness of ciphers with self-similar round subkey sequences.

7 Discussion

In this section we discuss possible extensions of slide attacks presented in this paper and possible directions of future research.

The most obvious type of slide attack is usually easy to prevent by destroying self-similarity in iterative ciphers, for example by adding iteration counters or fixed random constants. However more sophisticated variants of this technique are harder to analyze and to defend against. This paper is a first step towards advanced slide attacks which can penetrate more complex cipher designs.

One promising new direction is the *differential slide attack*. By sliding two encryptions against each other, we obtain new differential relations which in some cases are not available in the conventional differential analysis of a cipher. These might be very powerful, since they might for example violate the subtle design constraints placed on the system by its designer and thus result in unexpected differential properties. If key-scheduling is not self-similar or symmetric, differences in subkeys can cause constant XOR values to be introduced in the middle of the encryption process when slid pairs are considered. (In many cases, one can slide by different numbers of rounds and thus control the differences to some extent.) The drawback of this method is the same as in conventional methods: its complexity increases fast with the number of rounds, contrary to the general sliding technique, which works for arbitrary number of rounds.

Acknowledgments

We would like to thank Eli Biham for pointing to us the Brown-Seberry variant of DES key-schedule.

References

1. E. Biham, *New Types of Cryptanalytic Attacks Using Related Keys*, J. of Cryptology, Vol.7, pp.229–246, 1994.

[9] They analyzed an 8-round Feistel cipher with eight bits of key material per round used to swap between two S-boxes S_0 and S_1 in a Lucifer-like manner: a really weak cipher by modern criteria.

2. L. Brown, J. Seberry, *Key Scheduling in DES Type Cryptosystems*, proceedings of AUSCRYPT'90, LNCS 453, pp.221–228, Springer Verlag, 1990
3. A. Biryukov, D. Wagner, *Slide Attacks*, proceedings of FSE'99, LNCS 1636, pp.245–259, Springer Verlag, 1999.
4. C. Charnes, L. O'Connor, J. Pieprzyk, R. Safavi-Naini, Y. Zheng, *Comments on Soviet Encryption Algorithm*, proceedings of EUROCRYPT'94, LNCS 950, pp.433–438, Springer Verlag, 1994.
5. D. Coppersmith, *The Real Reason for Rivest's Phenomenon*, proceedings of CRYPTO'85, pp.535–536, Springer Verlag, 1986.
6. J. Daemen, *Limitations of the Even-Mansour Construction*, proceedings of ASIACRYPT'91, pp.495–498, Springer-Verlag 1992.
7. S. Even, Y. Mansour, *A Construction of a Cipher from a Single Pseudorandom Permutation*, Journal of Cryptology, Vol.10, No.3, pp.151–161, 1997.
8. E. K. Grossman, B. Tucherman, *Analysis of a Weakened Feistel-like Cipher*, 1978 International Conference on Communications, pp.46.3.1–46.3.5, Alger Press Limited, 1978.
9. J. Kelsey, B. Schneier, D. Wagner, *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, proceedings of CRYPTO'96, pp.237–251, Springer Verlag, 1996.
10. J. Kilian, P. Rogaway, *How to Protect Against Exhaustive Key Search*, proceedings of CRYPTO'96, LNCS 1109, pp.252–267, Springer Verlag, 1996.
11. B. Kaliski, M. Robshaw, *Multiple encryption: weighing security and performance*, Dr. Dobb's Journal, pp.123–127, Jan. 1996.
12. L. R. Knudsen, *Cryptanalysis of LOKI91*, proceedings of AUSCRYPT'92, LNCS 718, pp.196–208, Springer Verlag, 1993.
13. M. Luby, C. Rackoff, *How to Construct Pseudorandom Permutations from Pseudorandom Functions*, SIAM Journal of Computing, Vol. 17, pp.373–386, 1988.
14. J. H. Moore, G. J. Simmons, *Cycle Structure of the DES with Weak and Semi-Weak Keys*, proceedings of CRYPTO'86, pp.9–32, Springer Verlag, 1987.
15. B. Preneel, V. Rijmen, A. Bosselears, *Principles and Performance of Cryptographic Algorithms*, Dr. Dobb's Journal, Vol. 23, No. 12, pp.126–131, Miller Freeman, Dec. 1998.
16. P. Rogaway, *The Security of DESX*, RSA Laboratories' CryptoBytes, Summer 1996.
17. B. Schneier, *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*, proceedings of FSE'94, LNCS 809, pp.191–204, Springer Verlag, 1994.
18. C. Shannon, *Communication Theory of Secrecy Systems*, Bell Sys. Tech. J., Vol. 28, pp. 656–715, October 1949. (A declassified report from 1945.)
19. I. A. Zabotin, G. P. Glazkov, V. B. Isaeva, *Cryptographic Protection for Information Processing Systems. Cryptographic Transformation Algorithm*, Government Standard of the USSR, GOST 28147-89, 1989. (Translated by A. Malchik, with editorial and typographic assistance of W. Diffie.)

Author Index

Aardal, K. 1

Beaver, D. 340
Bellare, M. 140, 262
Biham, E. 414
Biryukov, A. 595
Bleichenbacher, D. 53
Boldyerva, A. 262
Boudot, F. 437
Boyko, V. 157
Brassard, G. 294

Camenisch, J. 246
Canetti, R. 459
Canteaut, A. 513, 579
Carlet, C. 513
Catalano, D. 191
Cavallar, S. 1
Charpin, P. 513
Coron, J.-S. 91, 374
Courtois, N. 398
Cramer, R. 321

Damgård, I. 321, 424
Desmedt, Y. 563
Di Crescenzo, G. 122
Dodis, Y. 459
Dodson, B. 1
Dumais, P. 305

Fontaine, C. 513

Gaudry, P. 19
Gennaro, R. 191
Gilchrist, J. 1
Girault, M. 81
Grieu, F. 70
Guillerm, G. 1

Halevi, S. 191, 459
Hirt, M. 545

Jarecki, S. 223
Jaulmes, E. 388
Joux, A. 388
Joye, M. 374

Kawamura, S. 529
Klimov, A. 398
Koike, M. 529
Kurosawa, K. 563
Kushilevitz, E. 104, 459

Lütkenhaus, N. 294
Lenstra, A. K. 1, 35
Leyland, P. 1
Lioen, W. 1
Lucks, S. 476
Lysyanskaya, A. 223

MacKenzie, P. 157
Maitra, S. 491
Malkin, T. 122
Marchand, J. 1
Maurer, U. 321, 356
Mayers, D. 305
Micali, S. 262
Michels, M. 246
Misarsky, J.-F. 81
Montgomery, P. L. 1
Mor, T. 294
Morain, F. 1
Muffet, A. 1
Murphy, B. 1

Naccache, D. 91, 374
Nguyen, P.Q. 53

Ostrovsky, R. 104, 122

Paillier, P. 374
Patarin, J. 398
Patel, S. 157
Pointcheval, D. 140
Poupard, G. 173
Putnam, Ch. 1
Putnam, Cr. 1

te Riele, H. 1
Rogaway, P. 140

Sahai, A. 459
Sako, K. 545

Sale, A. E. 423
Salvail, L. 305
Sanders, B.C. 294
Sano, F. 529
Sarkar, P. 491
Shamir, A. 35, 398
Shimbo, A. 529
Shoup, V. 209, 279, 451
Stern, J. 173

Trabbia, M. 579

Wagner, D. 595
Walker, M. 102
Wolf, S. 356

Zimmermann, P. 1