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.