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 $|x| < n^{1/e}$.

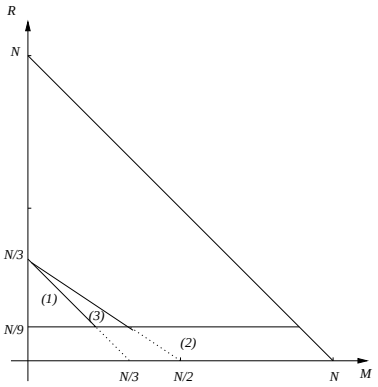


Fig. 1. Domains of validity for $e = 3$ of Coppersmith's stereotyped attack (1), Coppersmith's random padding attack (2) and our attack (3).

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

$$m_1 = \mu_1 = \text{PKCS}(m, r_1') \ .$$

$$\begin{aligned}
\Delta &= \texttt{00000001 fa75bf4e 390bdf4b 7a0524e0 b9ebed20 5758be2e f1685067 1de199af} \\
&\quad \texttt{0f8714f7 077a6c47 6870ea6d 2de9e7fb 3c40b8d2 017c0197 f9533ed1 f4fe3eab} \\
&\quad \texttt{836b6242 aa03181a 56a78001 7c164f7a c54ecfa7 73583ad8 ffeb3a78 eb8bcbe2} \\
&\quad \texttt{8869da15 60be7922 699dc29a 52038f7b 83e73d4e 7082700d 85d3a720} \\
p_1 &= \texttt{00000002}, \ p_2 = \texttt{00000007}, \ p_3 = \texttt{00000035}, \ p_4 = \texttt{000000c5}, \ p_5 = \texttt{4330e379} \\
p_6 &= \texttt{548063d7}, p_7 = \texttt{001ebf96 ff071021}, \ p_8 = \texttt{0000021b ac4d83ae 7dedba55} \\
p_9 &= \texttt{0000128a ec52c6ec 096996bf} \\
p_{10} &= \texttt{00000022 e3b1a6b0 13829b67 f604074a 5a1135b3 45be0835 ea407ed7 8138a27a} \\
&\quad \texttt{112e78c8 131f3bc3 b6d17dc0 e8a905f1 ca4b6aff 680bc58c 4962309d c7aaccad} \\
&\quad \texttt{2116235c b0d6803e e0a58ca7 55cbea23 e936f189 a76dfbeb} \\
\Delta_{25} &= \texttt{13bee453 6fba1cb1 6b2a5b6d d627ca60} \\
\mathcal{R}_{25}(z) &= z - m_1 \\
m_1/2^Z \bmod 2^M &\quad \texttt{0049276d 20612063 69706865 72746578 742c2070 6c656173 65206272 65616b20} \\
&\quad \texttt{6d652021}
\end{aligned}$$