Index Calculation Attacks on RSA Signature and Encryption

Jean-Sébastien Coron[1], Yvo Desmedt[2], David Naccache[1],
Andrew Odlyzko[3], and Julien P. Stern[4]

[1] Gemplus Card International
{jean-sebastien.coron,david.naccache}@gemplus.com
[2] Florida State University
desmedt@cs.fsu.edu
[3] University of Minnesota
odlyzko@umn.edu
[4] Cryptolog International
julien@cryptolog.com

Abstract. At Crypto '85, Desmedt and Odlyzko described a chosen-ciphertext attack against plain RSA encryption. The technique can also be applied to RSA signatures and enables an existential forgery under a chosen-message attack. The potential of this attack remained untapped until a twitch in the technique made it effective against two very popular RSA signature standards, namely ISO/IEC 9796-1 and ISO/IEC 9796-2. Following these attacks ISO/IEC 9796-1 was withdrawn and ISO/IEC 9796-2 amended. In this paper, we explain in detail Desmedt and Odlyzko's attack as well as its application to the cryptanalysis of ISO/IEC 9796-2.

1 Introduction

RSA was invented in 1977 by Rivest, Shamir and Adleman [13], and is now the most widely used public-key cryptosytem. RSA can be used for both encryption and signature.

A chosen-ciphertext attack against plain RSA encryption was described at Crypto '85 by Desmedt and Odlyzko [4]. In the plain RSA encryption scheme, a message m is simply encrypted as :

$$c = m^e \mod N$$

where N is the RSA modulus and e is the public exponent. Informally, during a chosen-ciphertext attack, an attacker may obtain the decryption of any ciphertext of his choice; the attacker's goal being to decrypt (or to recover some information about) some given ciphertext. However, Desmedt and Odlyzko's attack did not seem to threaten real-world RSA encryption standards, because in practice, the message m is generally encoded as $\mu(m)$ before being encrypted :

$$c = \mu(m)^e \mod N$$

where μ is some (probabilistic) algorithm.

As noted in [11], Desmedt and Odlyzko's attack can also be applied to RSA signatures. Recall that the RSA signature of a message m is defined as:

$$s = \mu(m)^d \mod N$$

where $\mu(m)$ is an encoding function and d the private exponent. As we will see below, Desmedt and Odlyzko's attack on RSA signatures only applies if the encoding function $\mu(m)$ produces integers much smaller than N. In this case, one obtains an existential forgery under a chosen-message attack. In this setting, the attacker can ask for the signature of any message of his choice, and his goal is to forge the signature for some (possibly meaningless) message which was not signed before.

At Crypto '99 [3], Coron, Naccache and Stern published an attack against the ISO/IEC 9796-2 RSA signature standard [7] and a slight variant of the ISO/IEC 9796-1 signature standard [6]. Both attacks were an adaptation of Desmedt and Odlyzko's attack, which could not be applied directly since for both standards $\mu(m)$ happened to be as big as N. Shortly after, the attack was extended to the real ISO/IEC 9796-1 standard by Coppersmith, Halevi and Jutla [2]. Following this final blow ISO/IEC 9796-1 was withdrawn and ISO/IEC 9796-2 amended.

This paper is organized as follows: we first recall the definition of the RSA cryptosystem. Then we describe Desmedt and Odlyzko's attack against plain RSA encryption, and eventually its application to the cryptanalysis of the ISO/IEC 9796-2 standard.

2 The RSA cryptosystem

The first instance of public-key encryption and digital signatures was invented in 1977 by Rivest, Shamir and Adleman [13]:

Definition 1 (The RSA Primitive). *The RSA primitive is a family of trapdoor permutations, specified by:*

- *The RSA generator $\mathcal{RSA}$, which on input 1^k, randomly selects two distinct $k/2$-bit primes p and q and computes the modulus $N = p \times q$. It randomly picks an encryption exponent $e \in \mathbb{Z}^*_{\phi(N)}$, computes the corresponding decryption exponent $d = e^{-1} \bmod \phi(N)$ and returns (N, e, d);*
- *The function $f : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ defined by $f(x) = x^e \bmod N$;*
- *The inverse function $f^{-1} : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ defined by $f^{-1}(y) = y^d \bmod N$.*

2.1 The RSA Encryption Scheme

The standard practice for encrypting a message m with RSA is to first apply an encoding scheme μ and raise $\mu(m)$ to the public exponent e. The algorithm μ is generally chosen to be probabilistic. The ciphertext c is then

$$c = \mu(m)^e \bmod N \enspace .$$

where (N, e) is the public-key. Decryption simply consists in using the private key d to compute :

$$\mu(m) = c^e \bmod N \enspace .$$

and recover m from $\mu(m)$.

2.2 The RSA Signature Scheme

As previously, the public-key is (N, e) and the private key is d. The RSA signature scheme is specified by an encoding function μ, which takes as input a message m and returns an integer modulo N, denoted $\mu(m)$. The signature of a message m is then:

$$s = \mu(m)^d \mod N$$

The signature s is verified by checking that :

$$\mu(m) \overset{?}{=} s^e \mod N$$

3 Attack on RSA Encryption

In [4], Desmedt and Odlyzko describe a chosen-ciphertext attack against plain RSA encryption. Recall that for plain RSA encryption, a message m is directly encrypted as $c = m^e \mod N$. The attack's setting is the following :

1. The attacker receives the public-key (N, e).
2. The attacker can ask for the decryption of any ciphertext of his choice, *i.e.* he submits x and receives $m = x^d \mod N$ for any x of his choice. The number of decryption queries is unlimited.
3. Upon receiving a challenge ciphertext c, the attacker's ability to make decryption queries ceases. The attacker must now output $c^d \mod N$.

Desmedt and Odlyzko's attack works as follows. After receiving the public-key (step 1), we ask for the decryption $x^d \mod N$ of all integers $x \in S = S_1 \cup S_2$, where:

$$S_1 = \{p : \quad p \leq L_N[\alpha], p \text{ is prime}\}$$

$$S_2 = \{\lfloor\sqrt{N}\rfloor + 1, \lfloor\sqrt{N}\rfloor + 2, \ldots, \lfloor\sqrt{N}\rfloor + \lfloor L_N[\alpha]\rfloor\}$$

where $\alpha > 0$ is some fixed parameter and the function $L_N[\alpha]$ is defined as :

$$L_N[\alpha] = \exp\left(\alpha \cdot \sqrt{\log N \log\log N}\right)$$

Once we have obtained $x^d \mod N$ for all $x \in S$ (step 2), we receive the challenge ciphertext c. We must now output $c^d \mod N$, without using the decrypting facility anymore (step 3). The basic idea is to find a representation:

$$c = y^e \prod_{x \in S} x^{a_x} \mod N \tag{1}$$

for some integers a_x and y, since then :

$$c^d = y \prod_{x \in S} (x^d)^{a_x} \mod N$$

where y and all the x^d are known.

To obtain the representation (1), we proceed in two steps. In the first step we find some integer y and primes $q_i \le L_N[2\alpha]$ such that:

$$c = y^e \prod_{i=1}^{h} q_i \mod N \tag{2}$$

To obtain the representation (2), we chose a random y, compute :

$$b = c \cdot y^{-e} \mod N$$

and check whether b factors into primes $q \le L_N[2\alpha]$. We use the following theorem [1] to estimate the average number of y values before such factorization is obtained.

Theorem 1. *Let x be an integer and let $L_x[\beta] = \exp\left(\beta \cdot \sqrt{\log x \log\log x}\right)$. Let z be an integer randomly distributed between zero and x^γ for some $\gamma > 0$. Then for large x, the probability that all the prime factors of z are lesser than $L_x[\beta]$ is given by :*

$$L_x\left[-\frac{\gamma}{2\beta} + o(1)\right]$$

Taking $\gamma = 1$ and $\beta = 2\alpha$, it appears that we need to generate on average $L_N[1/(4\alpha) + o(1)]$ values of y before such a factorization is obtained. Moreover, for each y, it takes $L_N[o(1)]$ bit operations to test whether such a factorization exists, using Lenstra's elliptic curve factorization algorithm [10]. Therefore this stage is expected to take time $L_N[1/(4\alpha) + o(1)]$. Although Lenstra's algorithm is asymptotically faster, it may be more efficient in practice to use trial division, for small enough prime factors.

Once a factorization of the form (2) is obtained, we proceed to the second step, in which we represent each of the at most $O(\log(N)) = L_N[o(1)]$ primes $q = q_i \le L_N[2\alpha]$ in the form:

$$q = \prod_{x \in S} x^{u_x} \mod N \tag{3}$$

where only $O(\log N)$ of the u_x are non-zero (possibly negative). Once such a representation is obtained for each q, we quickly obtain (1).

To see how to represent a prime $q \le L_N[2\alpha]$ in the form (3), let :

$$m = \left\lfloor \frac{\sqrt{N}}{q} \right\rfloor \tag{4}$$

and determine those integers among :

$$m+1, m+2, \ldots, m + \lfloor L_N[\beta] \rfloor$$

that are divisible solely by primes $p \le L_N[\alpha]$, for some $\beta > 0$. Using the previous theorem, we expect to find $L_N[\beta - 1/(4\alpha) + o(1)]$ such integers, and finding them will take $L_N[\beta + o(1)]$ bit operations if we employ Lenstra's factorization algorithm.

We next consider two cases. If $\alpha \geq \frac{1}{2}$, we take $\beta = \frac{1}{4\alpha} + \delta$ for any $\delta > 0$. We then have $L_N[\delta + o(1)]$ integers $m + j$, $1 \leq j \leq L_N[\beta]$, all of whose prime factors are $\leq L_N[\alpha]$. For each such integer and any i such that $1 \leq i \leq L_N[1/(4\alpha)] \leq L_N[\alpha]$, we write :

$$q(m+j)(k+i) = t \mod N \tag{5}$$

where $k = \lfloor \sqrt{N} \rfloor$. Using equation (4) and the corresponding bounds for q, j and i, we obtain that :

$$t \mod N \leq N^{\frac{1}{2}+o(1)}$$

Therefore, if the integers t factor like random integers of the same size, we will find $L_N[\delta + o(1)]$ integers t that factor into primes $\leq L_N[\alpha]$, and any single one will yield a factorization of the form (3), which gives the desired result. Since the testing of each t takes $L_N[o(1)]$ bit operations, this stage requires $L_N[\beta + o(1)]$ bit operations, and since this holds for all $\delta > 0$, we conclude that for $\alpha \geq \frac{1}{2}$, this stage can be carried out in $L_N[1/(4\alpha) + o(1)]$ bit operations.

It remains to consider the case $\alpha < \frac{1}{2}$. Here we take $\beta = \frac{1}{2\alpha} - \alpha + \delta$. We expect to find $L_N[\beta - 1/(4\alpha) + o(1)] = L_N[1/(4\alpha) - \alpha + \delta + o(1)]$ values of $m + j$, $1 \leq j \leq L_N[\beta]$, which factor into primes $\leq L_N[\alpha]$, and it takes $L_N[\beta + o(1)]$ bit operations to find them. For each one and for $1 \leq i \leq L_N[\alpha]$, we test whether the t defined by (5) is composed of primes $\leq L_N[\alpha]$. We expect to find $L_N[\delta + o(1)]$ of them. Letting $\delta \to 0$, we obtain that this case takes $L_N[1/(2\alpha) - \alpha + o(1)]$ bit operations.

We thus conclude that if the attacker can obtain decryptions of $L_N[\alpha]$ chosen ciphertexts he will be able to decrypt any individual ciphertext in $L_N[1/(4\alpha) + o(1)]$ bit operations for $\alpha \geq \frac{1}{2}$ and in $L_N[1/(2\alpha) - \alpha + o(1)]$ bit operations for $0 < \alpha \leq \frac{1}{2}$. For $\alpha = \frac{1}{2}$ both steps require $L_N[1/2 + o(1)]$ operations.

Therefore, Desmedt and Odlyzko's attack is asymptotically faster than the quadratic-sieve factorization algorithm [12], which requires $L_N[1 + o(1)]$ steps to recover the factorization of N. However, the attack is asymptotically slower than the general number field sieve algorithm [9] which appeared later, whose complexity to factor N is given by :

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

for some constant $c \simeq 1.9$.

Given that in practice RSA encryption schemes use an encoding function $\mu(m)$, the attack did not appear directly applicable to real-world standards. The situation nonetheless proved very different for RSA signature schemes, as explained in the next sections.

4 Attack on RSA Signatures

The previously described attack against RSA encryption can be easily adapted to RSA signatures to provide an existential forgery under a chosen-message attack, as shown in [11]. The outline of such a scenario is the following :

1. Select a bound y and let $S = (p_1, \ldots, p_\ell)$ be the list of primes smaller than y.
2. Find at least $\ell + 1$ messages m_i such that each $\mu(m_i)$ is the product of primes in S.

3. Express one $\mu(m_j)$ as a multiplicative combination of the other $\mu(m_i)$, by solving a linear system given by the exponent vectors of the $\mu(m_i)$ with respect to the primes in S.
4. Ask for the signature of the m_i for $i \neq j$ and forge the signature of m_j.

The attack's complexity depends on the cardinality of S and on the difficulty of finding at step (2) enough $\mu(m_i)$ values which are the product of primes in S. Generally, the attack would apply only when $\mu(m)$ is small; otherwise, the probability that $\mu(m)$ has only small prime factors is too small.

In the following, we describe the attack in more detail. First, we assume that e is a prime integer. We let τ be the number of messages m_i obtained at step (2). We say that an integer is B-smooth if all its prime factors are smaller than B. The integers $\mu(m_i)$ obtained at step (2) are therefore y-smooth and we can write for all messages m_i, $1 \leq i \leq \tau$:

$$\mu(m_i) = \prod_{j=1}^{\ell} p_j^{v_{i,j}} \tag{6}$$

Step (3) works as follows : to each $\mu(m_i)$ we associate the ℓ-dimensional vector of the exponents modulo e :

$$\boldsymbol{V_i} = (v_{i,1} \bmod e, \ldots, v_{i,\ell} \bmod e)$$

The set of all ℓ-dimensional vectors modulo e forms a linear space of dimension ℓ. Therefore, if $\tau \geq \ell + 1$, one can express one vector, say $\boldsymbol{V_\tau}$, as a linear combination of the others modulo e, using Gaussian elimination:

$$\boldsymbol{V}_\tau = \sum_{i=1}^{\tau-1} \beta_i \boldsymbol{V}_i + \boldsymbol{\Gamma} \cdot e \tag{7}$$

for some $\boldsymbol{\Gamma} = (\gamma_1, \ldots, \gamma_\ell)$. Denoting

$$\delta = \prod_{j=1}^{\ell} p_j^{\gamma_j} \tag{8}$$

one obtains from (6) and (7) that $\mu(m_\tau)$ is a multiplicative combination of the other $\mu(m_i)$:

$$\mu(m_\tau) = \delta^e \cdot \prod_{i=1}^{\tau-1} \mu(m_i)^{\beta_i} \tag{9}$$

Then, at step (4), the attacker will ask for the signature of the $\tau - 1$ first messages m_i and forge the signature of m_τ using:

$$\mu(m_\tau)^d = \delta \cdot \prod_{i=1}^{\tau-1} \left(\mu(m_i)^d\right)^{\beta_i} \mod N \tag{10}$$

The attack's complexity depends on ℓ and on the probability that the integers $\mu(m_i)$ are y-smooth. We define $\psi(x, y) = \#\{v \leq x, \text{ such that } v \text{ is } y\text{-smooth}\}$. It is known [5] that, for large x, the ratio $\psi(x, \sqrt[t]{x})/x$ is equivalent to Dickman's function defined by :

$$\rho(t) = \begin{cases} 1 & \text{if } 0 \leq t \leq 1 \\ \rho(n) - \displaystyle\int_n^t \frac{\rho(v-1)}{v} dv & \text{if } n \leq t \leq n+1 \end{cases}$$

$\rho(t)$ is thus an approximation of the probability that a u-bit number is $2^{u/t}$-smooth; table 1 gives the numerical value of $\rho(t)$ (on a logarithmic scale) for $1 \le t \le 10$.

t	1	2	3	4	5	6	7	8	9	10
$\log_2 \rho(t)$	0	-1.7	-4.4	-7.7	-11.5	-15.6	-20.1	-24.9	-29.9	-35.1

Table 1. The value of Dickman's function.

In the following, we provide an asymptotic analysis of the algorithm's complexity, based on the assumption that the integers $\mu(m)$ are uniformly distributed between zero and some given bound x. Letting β be a constant and letting:

$$y = L_x[\beta] = \exp\left(\beta \cdot \sqrt{\log x \log\log x}\right)$$

one obtains from theorem 1 that, for large x, the probability that an integer uniformly distributed between one and x is $L_x[\beta]$-smooth is:

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

Therefore, we have to generate on the average $L_x[1/(2\beta) + o(1)]$ integers $\mu(m)$ before we can find one which is y-smooth.

Using the ECM factorization algorithm [10], a prime factor p of an integer n can be found in time $L_p[\sqrt{2} + o(1)]$. A y-smooth integer can thus be factored in time $L_y[\sqrt{2} + o(1)] = L_x[o(1)]$. The complexity of finding a random integer in $[0, x]$ which is y-smooth using the ECM is thus $L_x\left[1/(2\beta) + o(1)\right]$. Moreover, the number τ of integers which are necessary to find a vector which is a linear combination of the others is $\ell + 1 \le y$. Therefore, one must solve a system with $r = L_x[\beta + o(1)]$ equations in $r = L_x[\beta + o(1)]$ unknowns. Using Lanzos' iterative algorithm [8], the time required to solve such system is $\mathcal{O}(r^2)$ and space is roughly $\mathcal{O}(r)$.

To summarize, the time required to obtain the $L_x[\beta + o(1)]$ equations is asymptotically $L_x\left[\beta + 1/(2\beta) + o(1)\right]$ and the system is solved in time $L_x[2\beta + o(1)]$. The total complexity is minimized by taking $\beta = 1/\sqrt{2}$. We obtain a time complexity of :

$$L_x[\sqrt{2} + o(1)]$$

and a space complexity of :

$$L_x\left[\frac{\sqrt{2}}{2} + o(1)\right]$$

where x is a bound on $\mu(m)$.

This complexity is sub-exponential in the size of the integers $\mu(m)$. Therefore, without any modification, the attack will be practical only if $\mu(m)$ is small. In particular, when $\mu(m)$ is about the size of N, the attack's complexity is worse than factoring N. Note that the attack can easily be extended to any exponent e, and also to Rabin signatures (see [3]).

In table 2, we give the values of the functions $L_x[\sqrt{2}]$ et $L_x[\sqrt{2}/2]$ corresponding to the attack's time complexity and space complexities, as a function of the size $|x|$ of the integer

$\mu(m_i)$. This table should be handled with care: being just an approximation of the attack's practical complexity, the attack may demand more time in practice. The table suggests that the attack can be practical when the size of $\mu(m)$ is smaller than 128 bits, but the attack becomes quickly impractical for larger values of $|x|$.

$\lvert x\rvert$	$\log_2$ **time**	$\log_2$ **space**
64	26	13
99	35	18
119	39	20
139	43	22
144	44	22
176	49	25
200	53	27
256	62	31
368	77	39

Table 2. The Attack's Complexity.

5 The Security of ISO/IEC 9796-2 Signatures

ISO/IEC 9796-2 [7] is an encoding standard allowing total or partial message recovery. The standard uses a hash-function HASH during the message formatting process. Let us denote by k_h the output size of the hash function. Hash-functions of different sizes are acceptable. Section 5, note 4 of [7] recommended (before the standard's correction by ISO following the attack described in this paper) $64 \le k_h \le 80$ for total message recovery and $128 \le k_h \le 160$ for partial message recovery.

For ISO/IEC 9796-2 , the encoding function $\mu(m)$ has the same size as N. Therefore, Desmedt and Odlyzko's attack can not apply directly here. Our technique will consist in generating messages m_i such that a linear combination t_i of $\mu(m_i)$ and N is much smaller than N. Then, the attack will be applied to the integers t_i instead of $\mu(m_i)$.

5.1 Partial Message Recovery

For simplicity, assume that k (the size of the modulus N), k_h and the size of m are all multiples of eight and that the hash function is known to both parties. The message m is separated into two parts $m = m[1]\|m[2]$ where $m[1]$ consists of the $k - k_h - 16$ most significant bits of m and $m[2]$ of all the remaining bits of m. The padding function is :

$$\mu(m) = \mathtt{6A}_{16}\|m[1]\|\mathrm{HASH}(m)\|\mathtt{BC}_{16}$$

and $m[2]$ is transmitted in clear.

Dividing $(\mathtt{6A}_{16} + 1) \cdot 2^k$ by N we obtain :

$$(\mathtt{6A}_{16} + 1) \cdot 2^k = i \cdot N + r \quad \text{with} \quad 0 \le r < N < 2^k$$

Defining $N' = i \cdot N$ we get :

$$N' = \mathtt{6A}_{16} \cdot 2^k + (2^k - r)$$

Therefore, we can write N' as :

$$N' = \mathtt{6A}_{16} \| N'[1] \| N'[0]$$

where the $N'[1]$ block is $k - k_h - 16$ bits long, the same bit-size as $m[1]$. Then, one can take $m[1] = N'[1]$, and letting :

$$t = 2^8 \cdot \mu(m) - i \cdot N$$

we obtain that :

$$t = (\mathtt{6A}_{16} \| m[1] \| \mathrm{HASH}(m) \| \mathtt{BC00}_{16}) - (\mathtt{6A}_{16} \| N'[1] \| N'[0])$$
$$t = (\mathrm{HASH}(m) \| \mathtt{BC00}_{16}) - N'[0]$$

where the size of t is less than $k_h + 16$ bits.

The attacker modifies $m[2]$ (and therefore $\mathrm{HASH}(m)$) until he finds sufficiently many integers t which are the product of small primes. Then since $t = 2^8 \cdot \mu(m) \mod N$, one can apply Desmedt and Odlyzko attack's described in section 4 to the integers t (the factor 2^8 can be added to the set $\mathcal{S}$). The attack's complexity is independent of the size of N; it only depends on the hash size k_h. From table 2, we derive table 3 expressing the attack's complexity, as a function of the hash size. For example, for $k_h = 128$, the size of t is 144 bits and from table 2, we obtain that time complexity is roughly 2^{44}. Again, recall that this is only an estimate, and that practical complexity may be much higher. Nevertheless the table suggests that the attack may be practical for $k_h = 128$, but will be more demanding for $k_h = 160$. Note that the following complexities are much smaller than the complexities obtained in [3]. This is due to the fact that we have obtained a smaller complexity in section 4.

k_h	$\log_2$ **time**	$\log_2$ **space**
128	44	22
160	49	25

Table 3. Attack's Complexity with Partial Message Recovery

5.2 Full Message Recovery

Assuming again that the hash function is known to both parties, that k and k_h are multiples of eight and that the size of m is $k - k_h - 16$, the encoding function μ is then defined as :

$$\mu(m) = \mathtt{4A}_{16} \| m \| \mathrm{HASH}(m) \| \mathtt{BC}_{16}$$

Let us separate $m = m[1] \| m[0]$ into two parts where $m[0]$ consists of the Δ least significant bits of m and $m[1]$ of all the remaining bits of m and compute, as in the previous case, an integer i such that :

$$N' = i \cdot N = \mathtt{4A}_{16} \| N'[1] \| N'[0]$$

where $N'[0]$ is $(k_h + \Delta + 16)$-bit long and $N'[1] \| N'[0]$ is k-bit long.

Setting $m[1] = N'[1]$ we get :

$$t = 2^8 \cdot \mu(m) - N' = (m[0] \| \text{HASH}(m) \| \texttt{BC00}_{16}) - N'[0]$$

where the size of t is less than $k_h + \Delta + 16$ bits.

The attacker will thus modify $m[0]$ (and therefore HASH(m)) as needed and conclude the attack as in the partial recovery case. As shown in section 4, the number of t-values necessary to forge a signature is roughly $L_x[\sqrt{2} + o(1)]$, where x is a bound on t. Therefore, the parameter Δ must be tuned so that $2^{\Delta} \simeq L_x[\sqrt{2}]$. From table 2, we obtain the attack complexities summarized in table 4, as a function of the hash size. For example, for $k_h = 64$, we take $\Delta = 39$ bits and the size of t is then $64 + 39 + 16 = 119$ bits and table 2 shows that the time complexity is roughly 2^{39}. This shows that the attack may be practical for $k_h = 64$. This actually led ISO to edit a revision of the ISO/IEC 9796-2 standard.

k_h	Δ	$\log_2$ **time**	$\log_2$ **space**
64	39	39	20
80	43	43	22
128	53	53	27

Table 4. Attack Complexity with Full Message Recovery

6 Conclusion

In this paper we explained in detail Desmedt and Odlyzko's attack and illustrated its potential by exhibiting a design flaw in the ISO/IEC 9796-2 signature standard. The publication of this attack drove ISO to correct and re-edit ISO/IEC 9796-2. A more elaborate variant (not described in this paper) [2, 3] led to the complete withdrawal of another signature standard, the ISO/IEC 9796-1 standard.

References

1. E. R. Canfield, P. Erdos and C. Pomerance, *On a Problem of Oppenheim Concerning 'Factorisation Numerorum'*, J. Number Th. 17, 1-28, 1983.
2. D. Coppersmith, S. Halevi and C. Jutla, *ISO 9796-1 and the new forgery strategy*, Research contribution to P1363, 1999, available at `http://grouper.ieee.org/groups/1363/contrib.html`.
3. J.S. Coron, D. Naccache and J.P. Stern, *On the security of RSA Padding*, Proceedings of Crypto '99, LNCS vol. 1666, Springer-Verlag, 1999, pp. 1-18.
4. Y. Desmedt and A. Odlyzko. *A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes*, Proceedings of Crypto '85, LNCS 218, pp. 516–522.
5. 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.
6. ISO/IEC 9796, *Information technology - Security techniques - Digital signature scheme giving message recovery, Part 1 : Mechanisms using redundancy*, 1999.

7. ISO/IEC 9796-2, *Information technology - Security techniques - Digital signature scheme giving message recovery, Part 2 : Mechanisms using a hash-function*, 1997.
8. C. Lanczos, *An iterative method for the solution of the eigenvalue problem of linear differential and integral operator*, J. Res. Nat. Bur. Standards, 1950, vol. 45, pp. 255–282.
9. A.K. Lenstra and H. W. Jr. Lenstra, *The Development of the Number Field Sieve*, Berlin: Springer-Verlag, 1993.
10. H. Lenstra, Jr., *Factoring integers with elliptic curves*, Ann. of Math. (2) 126 (1987) pp. 649-673.
11. J.-F. Misarsky, *How (not) to design RSA signature schemes*, Public-key cryptography, Springer-Verlag, Lectures notes in computer science 1431, pp. 14–28, 1998.
12. C. Pomerance, *The Quadratic Sieve Factoring Algorithm*, In Advances in Cryptology, Proceedings of Eurocrypt '84. Springer-Verlag, pp. 169-182, 1985.
13. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, CACM 21, 1978.