Evaluation of Security Level of Cryptography: RSA Signature Schemes (PKCS#1 v1.5, ANSI X9.31, ISO 9796)

Alfred Menezes
University of Waterloo
Contact: ajmeneze@uwaterloo.ca

July 31, 2002

Contents

1 Executive Summary **2**

2 Introduction **2**

3 Multiplicative Property of RSA **3**

3.1 RSA Key Generation 3

3.2 Basic RSA Signature Scheme 4

3.3 Multiplicative Attacks 4

4 RSA Signature Schemes with Appendix **6**

4.1 PKCS#1 v1.5 RSA Signature Scheme 6

4.2 ANSI X9.31 RSA Signature Scheme 7

4.3 Security 8

5 ISO 9796 RSA Signature Schemes with Message Recovery **10**

5.1 ISO 9796-1 RSA Signature Scheme 10

5.2 ISO 9796-2 RSA Signature Scheme 11

5.2.1 Partial Message Recovery 11

5.2.2 Full Message Recovery 14

6 Conclusions **16**

References **17**

1 Executive Summary

In recent years, some attacks on fixed-padding RSA signature schemes have been reported in the literature. This report evaluates the implications of these attacks on the RSA signature schemes specified in the PKCS#1 v1.5 [27], ANSI X9.31 [1], ISO 9796-1 [14] and ISO 9796-2 [15] standards.

2 Introduction

The RSA signature scheme was introduced in 1977 [24]. Since then many versions have been standardized and used in practice. The different versions can be divided into two categories:

1. *RSA signature schemes with appendix* first compute the hash value of the message, and then compute a signature by applying the RSA private-key operation to the (possibly padded) hash value. Both the message and the signature are then transmitted to the receiver. The receiver applies the RSA public-key operation to the signature, and compares the result to the hash of the message. The signed message is accepted if and only if these values are the same. The PKCS#1 v1.5 [27] and ANSI X9.31 [1] RSA signature schemes are examples of signature schemes with appendix.
2. *RSA signature schemes with message recovery* first add some redundancy to the message, and then apply the RSA private-key operations to the padded message to obtain the signature. The signature only is transmitted to the receiver, who applies the RSA public-key operation to recover the padded message and thereafter the message. The signed message is accepted if and only if the padded message contains the correct form of redundancy. The RSA signature schemes specified in ISO 9796-1 [14] and ISO 9796-2 [15] are examples of signature schemes with message recovery.

SECURITY. A signature scheme is said to be *secure* if it is existentially unforgeable against chosen-message attacks [11]. Informally, this means that an adversary who is able to obtain entity A's signatures on any messages of its choice is unable to successfully forge A's signature on some other message. This notion of security is a very strong one—the adversary may have no control over the contents of the message in the existential forgery and therefore the forgery may be of no practical use to her. A potentially more damaging attack is a *selective forgery* whereby the adversary is able to forge A's signature of some particular message of her choosing.

A necessary condition for the security of all RSA signature schemes is that the problem of inverting the RSA function (i.e., finding e^{th} roots modulo n) be intractable. Another necessary condition for the security of all RSA signature schemes with appendix is that the hash function employed be collision resistant.

In this report, we shall not be concerned with the intractability of the RSA function (nor of the integer factorization problem), with specific security properties of the hash function, or with the efficiency of the schemes. Instead, we shall focus our attention on attacks that exploit the fixed-padding mechanisms used in the PKCS#1 v1.5, ANSI X9.31, ISO 9796-1 and ISO 9796-2 RSA signature schemes. We note that unlike RSA-PSS and RSA-FDH[1] [3], none of these schemes have been proven to be secure under reasonable assumptions.

ORGANIZATION. The remainder of this report is structured as follows. The basic RSA signature scheme and the Desmedt-Odlyzko multiplicative attack which exploits its multiplicative property are reviewed in Section 3. The resistance of the PKCS#1 v1.5 and ANSI X9.31 RSA signature schemes with appendix to fixed-padding attacks is studied in Section 4. Section 5 considers the resistance of the ISO 9796-1 and ISO 9796-2 signatures schemes to fixed-padding attacks. Our conclusions are stated in Section 6.

3 Multiplicative Property of RSA

3.1 RSA Key Generation

The RSA key generation procedure is common to all RSA-based signature schemes. To generate an RSA key pair, each entity A does the following:

1. Randomly select two large distinct primes p and q of the same bit length.
2. Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$.
3. Select an arbitrary integer e, $1 < e < \phi(n)$, such that $\gcd(e, \phi(n)) = 1$.
4. Compute d, $1 < d < \phi(n)$, such that $ed \equiv 1 \pmod{\phi(n)}$.
5. A's public key is (n, e); A's private key is d.

NOTES.

1. n is called the *RSA modulus*.
2. e is called the *encryption exponent*.
3. d is called the *decryption exponent*.
4. If n is a k-bit integer, then the RSA scheme is referred to as *k-bit RSA*.

[1] Bellare and Rogaway proved that RSA-PSS and RSA-FDH are secure in the so-called random oracle model [2] under the assumptions that the problem of finding e^{th} roots modulo n is intractable and that the hash function employed is a public random function.

5. RSA signature verification involves modular exponentiations by the number e—computations of the form $M^e \bmod n$. To speed signature verification, a widely-employed strategy is to select a small encryption exponent e, such as $e = 3$ or $e = 2^{16} + 1$. With such a choice, RSA signature verification is much faster than RSA signature generation. Note that RSA signature generation cannot be sped up by selecting a small decryption exponent d without incurring a significant security risk [29, 4].

3.2 Basic RSA Signature Scheme

Let (n, e) be A's RSA public key, and let d be A's corresponding private key.

SIGNATURE GENERATION. To sign a message m, A does the following:

1. Compute $M = H(m)$ where H is a cryptographic hash function such as SHA-1.
2. Use the private key d to compute $s = M^d \bmod n$.
3. Send the signature s and the message m to B.

SIGNATURE VERIFICATION. The verifier B does the following:

1. Obtain an authentic copy of A's public key (n, e).
2. Compute $M = H(m)$.
3. Compute $M' = s^e \bmod n$.
4. Accept A's signature provided that $M = M'$.

3.3 Multiplicative Attacks

Multiplicative attacks on RSA signature schemes all exploit the *multiplicative property*[2] that

$$(m_1 m_2)^d \equiv m_1^d m_2^d \pmod{n}. \tag{1}$$

If RSA signature generation did not use a hash function, i.e., $s = m^d \bmod n$ for $m \in [0, n-1]$, then an active attacker could forge the signature of any message m as follows:

1. Select arbitrary $m_1 \in [1, n-1]$ and compute $m_2 = m m_1^{-1} \bmod n$ (so $m = m_1 m_2 \bmod n$).
2. Obtain A's signatures s_1, s_2 on m_1 and m_2, respectively.
3. Then it follows from (1) that A's signature on m is $s = s_1 s_2 \bmod n$.

[2] also known as the *homomorphic property*.

To avoid this selective forgery attack, the message is padded (or formatted) prior to application of the RSA operation. We let $P(m)$ denote the padded message corresponding to m, so the signature on m is $s = P(m)^d \bmod n$. The padding mechanism is chosen so that given a message m it is (hopefully) infeasible to find messages m_1 and m_2 such that $P(m) = P(m_1)P(m_2) \bmod n$.

The most common way to format a message m is $P(m) = H(m)$, where H is a cryptographic hash function. This results in the basic RSA signature scheme presented in Section 3.2. In 1985, Desmedt and Odlyzko [8] discovered the following selective forgery attack on basic RSA signatures.

DESMEDT-ODLYZKO SELECTIVE FORGERY ATTACK. For concreteness, we describe the attack in the case where n is a 1024-bit integer, e is prime (e.g., $e = 3$ or $e = 2^{16} + 1$), and the hash function H is SHA-1. The adversary's goal is to forge A's signature s on a variant m' of a message m.[3]

1. Select a smoothness bound $B = p_t$, where p_i denotes the i^{th} prime number.
2. Select random messages $m_1, m_2, \ldots, m_t$ until $H(m_i)$ are B-smooth:

$$H(m_i) = \prod_{j=1}^{t} p_j^{v_{ij}}, \quad 1 \le i \le t$$

   and the exponent vectors $v_i = (v_{i1}, v_{i2}, \ldots, v_{it})$ are linearly independent over $\mathbb{Z}_e$.
3. Obtain the signatures s_i of the m_i: $s_i = H(m_i)^d \bmod n$.
4. Select variants m' of m until $H(m')$ is B-smooth:

$$H(m') = \prod_{j-1}^{t} p_j^{w_j}.$$

5. Solve the linear system of equations

$$c_1 v_1 + c_2 v_2 + \cdots + c_t v_t = w$$

   for $c_1, c_2, \ldots, c_t \in \mathbb{Z}_e$.
6. Now,

$$w = c_1 v_1 + c_2 v_2 + \cdots + c_t v_t + eu$$

   over the integers. Hence

$$H(m') = H(m_1)^{c_1} H(m_2)^{c_2} \cdots H(m_t)^{c_t} \left(\prod_{j=1}^{t} p_j^{u_j} \right)^e.$$

[3] For example, if m is a fragment of plain text then the variants m' can be obtained by inserting extra spaces between words of m so that the meaning of m is unchanged.

Raising both sides to the power d and reducing modulo n gives:

$$s = s_1^{c_1} s_2^{c_2} \cdots s_t^{c_t} \prod_{j=1}^{t} p_j^{u_j} \bmod n. \tag{2}$$

Thus s can be computed according to the formula (2).

Using standard results on the distribution of smooth integers, we find that if $B \approx 2^{20}$ then the number of chosen messages in the attack is about 2^{16}, while the number of steps in the algorithm (hash function evaluations + smoothness testing) is about 2^{40}. Thus the attack must be considered as practical.

The attack can be prevented by using a hash function whose range is as large as the modulus size—this is the idea behind RSA-FDH. In that case, the probability that a random hash value is smooth is negligible. Indeed, running the attack will be slower than the number field sieve factoring algorithm which searches for smooth integers of significantly smaller sizes. Another approach to preventing the attack is to pad the hash value so that the resulting padded hash value has the same bitlength as n—this is the idea behind the padding mechanisms in PKCS#1 v1.5 and ANSI X9.31.

4 RSA Signature Schemes with Appendix

4.1 PKCS#1 v1.5 RSA Signature Scheme

We present the PKCS#1 v1.5 RSA signature scheme as described in [27][4]. This scheme is a refinement of the basic RSA signature scheme (Section 3.2). It specifies a method for padding the hashed message in order to defeat known attacks on basic RSA such as the Desmedt-Odlyzko multiplicative attack described in Section 3.3.

Let (n,e) be A's RSA public key, and let d be A's corresponding private key. The integer n is k octets in length. (For example, if n is a 1024-bit modulus, then $k = 128$.) H is a hash function such as SHA-1 that is identified by some octet string HID. For SHA-1, HID is 15 octets in length. For SHA-256, SHA-384 and SHA-512, HID is 18 octets in length.

SIGNATURE GENERATION. To sign a message m, A does the following:

1. Compute $h = H(m)$.
2. Form the following octet string M of length k octets:

$$00 \,\|\, 01 \,\|\, \mathrm{PS} \,\|\, 00 \,\|\, \mathrm{HID} \,\|\, h,$$

where PS denotes a string of 'FF' octets. The length of PS must be at least 8 octets.

[4]The PKCS#1 v1.5 RSA signature scheme described in PKCS#1 v2.1 [27] is the same as the original description in the PKCS#1 v1.5 standard [26].

3. Use the private key d to compute $s = M^d \bmod n$.
4. Send the signature s and the message m to B.

SIGNATURE VERIFICATION. The verifier B does the following:

1. Obtain an authentic copy of A's public key (n, e).
2. Compute $M' = s^e \bmod n$, and convert M' to an octet string of length k octets.
3. Compute $h = H(m)$.
4. Form the following octet string M of length k octets:

$$00 \,\|\, 01 \,\|\, \mathrm{PS} \,\|\, 00 \,\|\, \mathrm{HID} \,\|\, h,$$

   where PS denotes a string of 'FF' octets.
5. Compare M and M'. If $M = M'$ then accept the signature; otherwise reject the signature.

4.2 ANSI X9.31 RSA Signature Scheme

We present the ANSI X9.31 RSA signature scheme [1]. This scheme is a refinement of the basic RSA signature scheme (Section 3.2). It specifies a method for padding the hashed message in order to defeat known attacks on basic RSA such as the Desmedt-Odlyzko multiplicative attack described in Section 3.3.

Let (n, e) be A's RSA public key, and let d be A's corresponding private key. The integer n is k octets in length. (For example, if n is a 1024-bit modulus, then $k = 128$.) H is a hash function such as SHA-1 that is identified by some string HID of length 2 octets. For SHA-1, HID is the hexadecimal string '33CC'.

SIGNATURE GENERATION. To sign a message m, A does the following:

1. Compute $h = H(m)$.
2. Form the following octet string M of length k octets:

$$6 \,\|\, \mathrm{PS} \,\|\, h \,\|\, \mathrm{HID},$$

   where PS denotes the hexadecimal string 'BBBB...BBA'.
3. Use the private key d to compute $s = M^d \bmod n$.
4. Send the signature s and the message m to B.

SIGNATURE VERIFICATION. The verifier B does the following:

1. Obtain an authentic copy of A's public key (n, e).

2. Compute $M' = s^e \bmod n$, and convert M' to an octet string of length k octets.
3. Compute $h = H(m)$.
4. Form the following octet string M of length k octets:

$$6 \| \text{PS} \| h \| \text{HID},$$

   where PS denotes the hexadecimal string 'BBBB...BBA'.
5. Compare M and M'. If $M = M'$ then accept the signature; otherwise reject the signature.

STRONG PRIMES. ANSI X9.31 *mandates* that the primes p and q satisfy the following *strong prime* conditions:

1. $p-1$, $p+1$, $q-1$ and $q+1$ each have a prime factor of bitlength between 100 and 120.
2. p and q are different in at least one of the first 100 bits.

Condition 1 ensures resistance to the $p-1$ and $p+1$ factoring algorithms (due to Pollard [23] and Williams [30]). Condition 2 ensures resistance to a factoring algorithm of Fermat (and extended by Lehman [18]).[5]

We recall that the $p-1$ and $p+1$ factoring algorithms are "special-purpose" in that they are most effective when $p-1$ or $p+1$ have only small factors. After the discovery of the elliptic curve factoring method (ECM) by Lenstra in 1985 [20], it became clear that the requirements that $p-1$ and $p+1$ each have a large factor are no longer necessary. This is because it is more important to guard against the ECM which is effective whenever an elliptic curve defined over $\mathbb{F}_p$ can be found whose order is smooth—this includes the cases where the elliptic curve has order $p-1$ or $p+1$. Moreover, Rivest and Silverman [25] (see also Silverman [28]) have shown that the $p-1$ and $p+1$ factoring algorithms are almost certainly to fail if the primes p and q are sufficiently large and chosen at random. Clearly, condition 2 is also satisfied with overwhelming probability if the primes p and q are randomly generated.

We conclude that while there is no security weakness if p and q are chosen to be strong primes, there is also no reason to require that p and q be strong primes.

4.3 Security

BCCN ATTACK. Brier, Clavier, Coron and Naccache (BCCN) [5] presented the following *existential forgery* attack on fixed padding RSA signature schemes. The attack improves earlier attack of Girault and Misarsky [9] and Misarsky [21]. The survey article by Misarsky [22]

[5] Another condition sometimes imposed on p (and q) is that the large factor r of $p-1$ must be so that $r-1$ also has a large prime factor. This condition maximizes resistance to a cyclic attack on the RSA encryption scheme. Since this attack is not relevant to signature schemes, the condition on r is not required.

provides a comprehensive overview on attacks known on fixed padding RSA signature schemes prior to 1998.

Let the padding of an RSA message m be

$$P(m) = wm + a. \tag{3}$$

For example, if the padded message is $R \| m \| L$ where L is s bits in length and m is l bits in length, then we can write $P(m) = wm + a$ where $w = 2^s$ and $a = L + 2^{s+l}R$.

Suppose now that messages are at most l bits in length, where l is an integer whose bitlength is at most $1/3$ that of n. The idea of the BCCN attack is to find messages m_1, m_2, m_3, m_4 such that

$$P(m_1)P(m_2) \equiv P(m_3)P(m_4) \pmod{n}. \tag{4}$$

Then, after obtaining the signatures of m_1, m_2 and m_4, the adversary can compute the signature of m_3 as follows:

$$P(m_3)^d = \frac{P(m_1)^d P(m_2)^d}{p(m_4)^d} \bmod n.$$

Finding m_1, m_2, m_3, m_4 satisfying (4) is equivalent to solving

$$Rz \equiv xy - tz \pmod{n} \tag{5}$$

for t, x, y, z where $R = a/w \bmod n$, and where

$$\begin{aligned} m_1 &= x + t \\ m_2 &= y + t \\ m_3 &= t \\ m_4 &= x + y + z + t \end{aligned}$$

are each at most l bits in length. The algorithm for finding t, x, y, z is the following:

1. Run the Euclidean algorithm with inputs R and n to find the continued fraction convergent P_j/Q_j to R/n satisfying $Q_j < n^{1/3} \leq Q_{j+1}$.
2. Let $z = |Q_j|$ and $u = |RQ_j - nP_j|$.
3. Select an arbitrary integer y with $n^{1/3} \leq y \leq 2n^{1/3}$ and $\gcd(y, z) = 1$.
4. Compute $t = -uz^{-1} \bmod y$.
5. Let $x = (u + tz)/y$.

Now, in the PKCS#1 v1.5 and ANSI X9.31 signature schemes, the padding function is of the form

$$P(m) = wH(m) + a \tag{6}$$

for some w and a. Note here that (6) differs from (3) in that the *hash value* of m is used instead of m itself. In the BCCN attack, the adversary does not have any control over the value of z (which is determined from R and n). Moreover, after the adversary has chosen y, the values of t and x are determined. Thus, the BCCN attack is doomed to fail for the PKCS#1 v1.5 and ANSI X9.31 signature schemes since the attacker would need to determine messages whose *hash values* are determined from t, x, y, z (since the adversary has to present the actual messages to the signing oracle, and not their hash values). Finding messages from hash values is infeasible assuming that the hash function is one-way.

We could not see how to modify the BCCN attack to be effective on the PKCS#1 v1.5 and ANSI X9.31 signature schemes.

LENSTRA-SHPARLINSKI ATTACK. Lenstra and Shparlinski [19] presented the following improvement to a *selective forgery* attack of Brier, Clavier, Coron and Naccache [5]. For a selective forgery of m_3, congruence (5) is rewritten as

$$(R + m_3)z \equiv xy \pmod{n}. \tag{7}$$

Given m_3, the adversary's task is to find x, y, z satisfying (7) so that the corresponding m_1, m_2, m_4 are at most l bits long. Lenstra and Shparlinski describe a method for finding such x, y, z. In this method, the adversary has some freedom in the selection of x, y and z. However, there is not enough freedom to allow solutions x, y and z to be chosen so that messages corresponding to hash values m_1, m_2, m_4 are known. Thus, as with the BCCN attack, the adversary is still faced with the insurmountable task of finding the messages corresponding to the derived hash values.

We could not see how to modify the Lenstra-Shparlinski attack to be effective on the PKCS#1 v1.5 and ANSI X9.31 signature schemes.

5 ISO 9796 RSA Signature Schemes with Message Recovery

The ISO 9796 standard covers signature schemes with message recovery. It has three parts:

1. ISO 9796-1 [14]: RSA and Rabin schemes without a hash function.
2. ISO 9796-2 [15]: RSA and Rabin schemes using a hash function.
3. ISO 9796-3 [16]: Discrete logarithm based mechanisms.

This section studies the RSA signature schemes in ISO 9796-1 and ISO 9706-2.

5.1 ISO 9796-1 RSA Signature Scheme

ISO 9796-1 [14] was first published in 1991. It is an RSA signature scheme with message recovery. Plaintext messages can be at most half the bitlength of the modulus. Rationale for

the redundancy function in ISO 9796-1 was provided in [13]. We will not present the details of the redundancy function because, as described below, the ISO 9796-1 signature scheme is now considered to be completely broken.

Coron, Naccache and Stern [7] in 1999 presented an attack on a slight modification of the ISO 9796-1 signature scheme. The attack is a refinement of the Desmedt-Odlyzko chosen-message attack that exploits the special structure of padded messages. Shortly after, Coppersmith, Halevi and Jutla [6] modified the attack to work on the (unmodified) ISO 9796-1 scheme. In 2000, Grieu [12] introduced a simpler attack that can efficiently find 4 messages m_1, m_2, m_3, m_4 such that

$$P(m_1)P(m_2) \equiv P(m_3)P(m_4) \pmod{n} \tag{8}$$

and can therefore construct the signature for any one of these messages given the signatures for the other three messages. Girault and Misarsky [10] also showed that several countermeasures proposed by Coppersmith, Halevi and Jutla [6] were not adequate for repairing the signature scheme.

We conclude from this recent work that the ISO 9796-1 signature scheme should be considered to be insecure and should not be used in practice.

We note that the attacks of Coron, Naccache and Stern [7], Coppersmith, Halevi and Jutla [6], and Grieu [12] do not appear to apply to the PKCS#1 v1.5 and ANSI X9.31 RSA signature schemes.[6]

5.2 ISO 9796-2 RSA Signature Scheme

ISO 9796-2 specifies an RSA signature scheme with partial message recovery and one with full message recovery.

We note that ISO 9796-2 is in the process of being revised (see [17]). The revision, currently under development, is expected to contain three signature schemes, one of which will be the same as the scheme specified the current edition [15]. All three schemes will be RSA-based, but will be distinguished by the use of three different formatting mechanisms.

5.2.1 Partial Message Recovery

Let (n,e) be A's RSA public key, and let d be A's corresponding private key. H is an l-bit hash function such as SHA-1. We assume that l and the bitlength k of n are multiples of 8, as is the bitlength of the message m.

[6] Coron, Naccache and Stern [7] observe that their attack does apply to the PKCS#1 v1.5 and ANSI X9.31 signature schemes if the RSA modulus is of the special form $n = 2^k \pm c$ where c is small. However since such RSA moduli are not used in practice, the attack is not effective in practice.

SIGNATURE GENERATION. To sign a message m, A does the following:

1. Write $m = m_L \,\|\, m_R$ where m_L has bitlength $k - l - 16$.
2. Compute $h = H(m)$.
3. Form the following octet string M of length k bits:
$$6A \,\|\, m_L \,\|\, h \,\|\, BC.$$
4. Use the private key d to compute $s = M^d \bmod n$.
5. Send the signature s and m_R to B.

SIGNATURE VERIFICATION. The verifier B does the following:

1. Obtain an authentic copy of A's public key (n, e).
2. Compute $M' = s^e \bmod n$ and parse M' as follows:
$$M' = XX \,\|\, m_L \,\|\, h' \,\|\, YY,$$
where 'XX' and 'YY' are hexadecimal strings of length 2.
3. Verify that $XX = 6A$ and $YY = BC$. If not, then reject the signature.
4. Compute $h = H(m_L \,\|\, m_R)$.
5. If $h \neq h'$ then reject the signature.
Else, accept $m = m_L \,\|\, m_R$ as the message signed by A.

SECURITY. Coron, Naccache and Stern (CNS) [7] observed that the Desmedt-Odlyzko multiplicative attack can be adapted to attack the ISO 9796-2 signature scheme with partial message recovery. In the following, we use bit strings and hexadecimal strings to represent integers.

Denote the padded message m by $\mu(m)$; that is,
$$\mu(m) = 6A \,\|\, m_L \,\|\, H(m) \,\|\, BC.$$
Dividing the integer $(6A+1)2^k$ by n yields a quotient q and a remainder r satisfying
$$(6A+1)2^k = qn + r, \text{ where } r < n < 2^k.$$
Now, let
$$n' = qn = (6A)2^k + (2^k - r) = 6A \,\|\, n'_L \,\|\, n'_R,$$
where n' has bitlength $k+7$, n'_L has bitlength $k - l - 16$, and n'_R has bitlength $l + 16$. Letting $m_L = n'_L$, we see that
$$T(m) = qn - \mu(m)2^8 = n'_R - (H(m) \,\|\, BC00) \tag{9}$$
has bitlength at most $l + 16$. The adversary, whose goal is to forge A's signature s on a variant m' of a message $m = n'_L \,\|\, m_R$ (for any m_R of the adversary's choice), does the following:

1. Select a smoothness bound $B = p_t$, where p_i denotes the i^{th} prime number. Let $p_{t+1} = -1$.
2. Select arbitrary $m_{R,1}, m_{R,2}, \ldots, m_{R,t+1}$ until the corresponding T values are B-smooth:

$$T(m_i) = qn - \mu(m_i)2^8 = \prod_{j=1}^{t} p_j^{z_{ij}}, \quad 1 \le i \le t+1,$$

where $m_i = n'_L \,\|\, m_{Ri}$, and the $t+1$ exponent vectors

$$v_i = (v_{i1}, v_{i2}, \ldots, v_{it}, v_{i,t+1}) = (z_{i1} - 8, z_{i2}, \ldots, z_{it}, 1)$$

are linearly independent over $\mathbb{Z}_e$. Note that by (9) we have

$$\mu(m_i) \equiv -T(m_i)2^{-8} \equiv \prod_{j=1}^{t+1} p_j^{v_j} \pmod{n} \quad \text{for } 1 \le i \le t+1.$$

3. Obtain the signatures s_i of the m_i: $s_i = \mu(m_i)^d \bmod n$.
4. Select variants m'_R of m_R until $T(m')$ is B-smooth:

$$T(m') = qn - \mu(m')2^8 = \prod_{j=1}^{t} p_j^{z_j},$$

where $m' = n'_L \,\|\, m'_R$. Set $w = (z_1 - 8, z_2, \ldots, z_t, 1)$.

5. Solve the linear system of equations

$$c_1 v_1 + c_2 v_2 + \cdots + c_{t+1} v_{t+1} = w$$

for $c_1, c_2, \ldots, c_{t+1} \in \mathbb{Z}_e$.

6. Now,

$$w = c_1 v_1 + c_2 v_2 + \cdots + c_{t+1} v_{t+1} + eu$$

over the integers. Hence

$$\mu(m') \equiv \mu(m_1)^{c_1} \mu(m_2)^{c_2} \cdots \mu(m_{t+1})^{c_{t+1}} \left(\prod_{j=1}^{t+1} p_j^{u_j} \right)^e \pmod{n}.$$

Raising both sides to the power d and reducing modulo n gives

$$s \equiv s_1^{c_1} s_2^{c_2} \cdots s_{t+1}^{c_{t+1}} \prod_{j=1}^{t+1} p_j^{u_j} \pmod{n}. \tag{10}$$

Thus s can be computed according to formula (10).

ANALYSIS. The running time of this attack depends on the choice of the smoothness bound $B = p_t$. The optimum choice for B depends on the distribution of smooth integers of bitlength $l+16$ (recall that the $T(m_i)$ have bitlength $l+16$), and not on the parameter k (the bitlength of the modulus n). If $l = 160$ (the bitlength of SHA-1 outputs), then the optimum choice of t is approximately 2^{20}. In this case, the adversary needs to query the signing oracle approximately 2^{20} times, and the expected running time (hash function evaluations + smoothness testing) is about 2^{61}—it is important to note that this running time is independent of the bitlength of the RSA modulus. The running time of 2^{61} is less than the expected running time of the number field sieve to factor 1024-bit RSA moduli (about 2^{80} steps). Thus the CNS attack is effective in that it is faster than the best known factoring algorithm. On the other hand, the attack still requires the signatures of 2^{20} chosen-messages, so it can be argued that the attack is not practical in many scenarios.

We conclude that the CNS attack is a concern for the ISO 9796-2 signature scheme with partial message recovery in environments where the attacker is capable of obtaining the signatures of a significant number (e.g., one million) of chosen messages. In environments where the attacker is not capable of obtaining these signatures, the CNS attack is not a concern.

5.2.2 Full Message Recovery

Let (n,e) be A's RSA public key, and let d be A's corresponding private key. H is an l-bit hash function. ISO 9796-2 recommends that $l \in [64,80]$. We assume that l and the bitlength k of n are multiples of 8, and that the message m has bitlength $k-l-16$.

SIGNATURE GENERATION. To sign a message m, A does the following:

1. Compute $h = H(m)$.
2. Form the following octet string M of length k bits:

$$4A \,||\, m \,||\, h \,||\, BC.$$

3. Use the private key d to compute $s = M^d \bmod n$.
4. Send the signature s to B.

SIGNATURE VERIFICATION. The verifier B does the following:

1. Obtain an authentic copy of A's public key (n,e).
2. Compute $M' = s^e \bmod n$ and parse M' as follows:

$$M' = XX \,||\, m \,||\, h' \,||\, YY,$$

where 'XX' and 'YY' are hexadecimal strings of length 2.

3. Verify that $XX = 4A$ and $YY = BC$. If not, then reject the signature.
4. Compute $h = H(m)$.
5. If $h \neq h'$ then reject the signature.
   Else, accept m as the message signed by A.

SECURITY. Coron, Naccache and Stern (CNS) [7] observed that the Desmedt-Odlyzko multiplicative attack can also be adapted to the ISO 9796-2 RSA signature with full message recovery. The attack is similar to the one described in Section 5.2.1. Here again, we use bitstrings and hexadecimal strings to represent integers.

We write a plaintext message m as $m = m_L \| m_R$ where m_R has bitlength x. Here, x is a parameter to be chosen later. Denote the padded message m by $\mu(m)$; that is,

$$\mu(m) = 4A \| m \| H(m) \| BC.$$

Dividing the integer $(4A+1)2^k$ by n yields a quotient and a remainder r satisfying

$$(4A+1)2^k = qn + r, \text{ where } r < n < 2^k.$$

Now, let

$$n' = qn = (4A)2^k + (2^k - r) = 4A \| n'_L \| n'_R,$$

where n' has bitlength $k+7$, n'_R has bitlength $l+x+16$, and n'_L has bitlength $k-l-x-9$. Letting $m_L = n'_L$, we see that

$$T(m) = qn - \mu(m)2^8 = n'_R - (m_R \| H(m) \| BC00) \tag{11}$$

has bitlength at most $l+x+16$. The adversary can now modify m_R until $T(m)$ is smooth. The attack then proceeds as in the CNS attack described in Section 5.2.1.

ANALYSIS. The running time of this attack depends on the choice of x and the smoothness bound $B = p_t$. The optimum choice for B depends on the distribution of smooth integers of bitlength $l+x+16$ (recall that the $T(m_i)$ have bitlength $l+x+16$), and not on the parameter k (the bitlength of the modulus n). For example, if $l = 64$, then $x = 32$ and $t \approx 2^{15}$. In this case, the adversary needs to query the signing oracle approximately 2^{15} times, and the expected running time (hash function evaluations + smoothness testing) is about 2^{47}—it is important to note that this running time is independent of the bitlength of the RSA modulus. Similarly, if $l = 80$, then $x = 34$ and $t \approx 2^{17}$. In this case, the adversary needs to query the signing oracle approximately 2^{17} times, and the expected running time (hash function evaluations + smoothness testing) is about 2^{51}. These running times are significantly less than the expected running time of the number field sieve to factor 1024-bit RSA moduli (about 2^{80} steps). Thus the CNS attack is effective in that it is faster than the best known factoring algorithm. On the other hand, the attack still requires the signatures of about 2^{15} chosen-messages, so it can be argued that the attack is not practical in many scenarios.

We conclude that the CNS attack is a concern for the ISO 9796-2 signature scheme with full message recovery in environments where the attacker is capable of obtaining the signatures of a significant number (e.g., 50 thousand) of chosen messages. In environments where the attacker is not capable of obtaining these signatures, the CNS attack is not a concern.

6 Conclusions

The recently published attacks on fixed-padding RSA signature schemes completely break the ISO 9796-1 RSA signature scheme with message recovery. They also seriously threaten the security of the ISO 9796-2 RSA signature schemes in environments where an adversary is able to obtain the signatures of a significant number (e.g., one million) of chosen messages. For these reasons, we recommend that the ISO 9796-1 and ISO 9796-2 standards not be adopted by CRYPTREC.

The attacks are not a threat to the practical security of schemes described in the ANSI X9.31 and PKCS#1 v1.5 standards. Nevertheless, the possibility of an effective attack being discovered in the future still remains. It is strongly recommended that RSA signature schemes with pseudorandom padding be used instead. The best candidate is the RSA PSS signature scheme [3] which has been proven secure in the random oracle model.

References

[1] ANSI X9.31, *Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*, 1998.

[2] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", *1st ACM Conference on Computer and Communications Security*, 1993, 62-73. Full version available at http://www.cs.ucdavis.edu/~rogaway/papers/index.html.

[3] M. Bellare and P. Rogaway, "The exact security of digital signatures—How to sign with RSA and Rabin", *Advances in Cryptology—Eurocrypt '96*, Lecture Notes in Computer Science, **1070** (1996), Springer-Verlag, 399-416. Full version available at http://www.cs.ucdavis.edu/~rogaway/papers/index.html

[4] D. Boneh and G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$", *IEEE Transactions on Information Theory*, **46** (2000), 1339-1349.

[5] E. Brier, C. Clavier, J. Coron and D. Naccache, "Cryptanalysis of RSA signatures with fixed-pattern padding", *Advances in Cryptology—Crypto 2001*, Lecture Notes in Computer Science, **2139** (2001), Springer-Verlag, 433-439.

[6] D. Coppersmith, S. Halevi and C. Jutla, "ISO 9796-1 and the new forgery strategy", working draft, 1999. Available at http://grouper.ieee.org/groups/1363/Research/Cryptanalysis.html.

[7] J. Coron, D. Naccache and J. Stern, "On the security of RSA padding", *Advanced in Cryptology—Crypto '99*, Lecture Notes in Computer Science, **1666** (1999), Springer-Verlag, 1-18.

[8] Y. Desmedt and A. Odlyzko, "A chosen text attack on RSA cryptosystem and some discrete logarithm schemes", *Advanced in Cryptology—Crypto '85*, Lecture Notes in Computer Science, **218** (1986), Springer-Verlag, 516-522.

[9] M. Girault and J. Misarsky, "Selective forgery of RSA signatures using redundancy", *Advances in Cryptology—Eurocrypt '97*, Lecture Notes in Computer Science, **1233** (1997), Springer-Verlag, 495-507.

[10] M. Girault and J. Misarsky, "Cryptanalysis of countermeasures proposed for repairing ISO 9796-1", *Advances in Cryptology—Eurocrypt 2000*, Lecture Notes in Computer Science, **1807** (2000), Springer-Verlag, 81-90.

[11] S. Goldwasser, S. Micali and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks", *SIAM J. Computing*, **17** (1988), 281-308.

[12] F. Grieu, "A chosen message attack on ISO/IEC 9796-1 signature scheme", *Advances in Cryptology—Eurocrypt 2000*, Lecture Notes in Computer Science, **1807** (2000), Springer-Verlag, 70-80.

[13] L. Guillou and J. Quisquater, "Precautions taken against various potential attacks in ISO/IEC DIS 9796", *Advances in Cryptology—Eurocrypt '90*, Lecture Notes in Computer Science, **473** (1991), Springer-Verlag, 465-473.

[14] ISO/IEC 9796-1, *Information Technology–Security Techniques-Digital Signature Scheme Giving Message Recovery, Part 1: Mechanisms Using Redundancy*, 1999.

[15] ISO/IEC 9796-2, *Information Technology–Security Techniques-Digital Signature Scheme Giving Message Recovery, Part 1: Mechanisms Using a Hash Function*, 1997.

[16] ISO/IEC 9796-3, *Information Technology–Security Techniques-Digital Signature Scheme Giving Message Recovery, Part 3: Discrete Logarithm Based Mechanisms*, 2000.

[17] ISO/IEC FDIS 9796-2 (2002), *Digital signature schemes giving message recovery - Part 2: Integer factorization based mechanisms*, 2002. Available at http://www.din.de/ni/sc27/doc7.html.

[18] R. Lehman, "Factoring large integers", *Mathematics of Computation*, **28** (1974), 637-646.

[19] A. Lenstra and I. Shparlinski, "Selective forgery of RSA signatures with fixed-pattern padding", *Proceedings of PKC 2002*, Lecture Notes in Computer Science, **2274** (2002), Springer-Verlag, 228-236.

[20] H. Lenstra, "Factoring integers with elliptic curves", *Annals of Mathematics*, **126** (1987), 649-673.

[21] J. Misarsky, "A multiplicative attack using LLL algorithm on RSA signatures with redundancy", *Advances in Cryptology—Crypto '97*, Lecture Notes in Computer Science, **1294** (1997), Springer-Verlag, 221-234.

[22] J. Misarsky, "How (not) to design RSA signature schemes", *Proceedings of PKC '98*, Lecture Notes in Computer Science, **1431** (1998), Springer-Verlag, 14-28.

[23] J. Pollard, "Theorems on factorization and primality testing", *Proceedings of the Cambridge Philosophical Society*, **76** (1974), 521-528.

[24] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, **21** (1978), 120-126.

[25] R. Rivest and R. Silverman, "Are 'strong' primes needed for RSA?", Report 2001/007, *Cryptology ePrint Archive*, 2001. Available at http://http://eprint.iacr.org/

[26] RSA Laboratories, *PKCS#1 v1.5: RSA Encryption Standard*, November 1993. Available at http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/

[27] RSA Laboratories, *PKCS#1 v2.1: RSA Cryptography Standard*), June 14 2002. Available at http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/

[28] R. Silverman, "Fast generation of random, strong RSA primes", *CryptoBytes*, **1**, Springer 1997, 9-13. Available at http://www.rsasecurity.com/rsalabs/cryptobytes/index.html

[29] M. Wiener, "Cryptanalysis of short RSA secret exponents", *IEEE Transactions on Information Theory*, **36** (1990), 553-558.

[30] H. Williams, "A $p+1$ method of factoring", *Mathematics of Computation*, **39** (1982), 225-234.