Padding Attacks on RSA

[Published in *Information Security Technical Report* **4**(4):28–33, 1999.]

David Naccache

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

Abstract. This paper presents a non-technical overview of the the recent attacks against RSA encryption and signature standards. It is intended as both a system design aid and a temporary reference text beginning at a level suitable for engineers, risk managers and system architects with no or little previous exposure to padding attacks.

We have used a straightforward approach to the essential consequences of each attack, coupled with a pragmatic in-depth selection of some of the most popular padding schemes.

1 Introduction

At a recent count (`http://www.rsa.com`), over 450 million RSA-enabled products had been shipped worldwide. This popularity, and the ongoing standardizations of signature and encryption formats [1, 14, 19–21, 30] highlight the need to challenge claims that such standards eradicate RSA's multiplicative properties.

Exponentiation is homomorphic and RSA-based protocols are traditionally protected against chosen-plaintext forgeries [12, 13, 29] by using a *padding* (or *redundancy*) function μ to make sure that:

$$\mathrm{RSA}(\mu(x)) \times \mathrm{RSA}(\mu(y)) \neq \mathrm{RSA}(\mu(x \times y)) \bmod n$$

In general, $\mu(x)$ hashes x and concatenates its digest to pre-defined strings; in some cases, substitution and permutation are used as well.

While most padding schemes gain progressive recognition as time goes by, several specific results exist: a few functions were broken by *ad-hoc* analysis ([16, 23] showed, for instance, that homomorphic dependencies can still appear in $\mu(m) = a \times m + b$) while at the other extreme, assuming that the underlying building-blocks are ideal, some functions [3, 5] are provably secure in the random oracle model.

RSA encryption also employs padding. Encryption padding is necessary to avoid dictionary attacks: by adding a random string to the encrypted message, the re-encryption of very short messages (such as `yes` or `no`) does not allow *déjà vu* attacks (building-up dictionaries).

We refer the reader to the following bibliography for further details concerning RSA signature forgery: Misarsky's PKC'98 invited survey [26] and recent thesis are probably the best documented reference on multiplicative RSA forgeries. Davida's observation [12] is the basis of most RSA forgery techniques. [16, 23] forge signatures that are similar to PKCS #1 V1.5 but do not produce their necessary SHA/MD5 digests [27, 28]. [15] analyzes the security of RSA signatures in an interactive context. Michels *et al.* [24] create relations between the exponents of de Jonge-Chaum and Boyd's schemes. Baudron and Stern [2] apply lattice reduction to analyze the security of RSA $\circ\ \mu$ in a security-proof perspective.

The following references cover attacks on various encryption padding formats: In 1998, Bleichenbacher published an adaptive chosen ciphertext attack on PKCS #1 V1.5 [6] capable of recovering arbitrary plaintexts from a few hundreds of thousands of ciphertexts; PKCS #1 V1.5 was subsequently updated (PKCS #1 V2.0) [30] and patches were issued to users wishing to continue using the old version of the standard. Further attacks can be found in [8] (where an affine relation between the plaintexts in assumed) and [9] where lattice reduction techniques are used. We also refer the reader to Boneh's general survey of RSA attacks [7].

2 The Current Status of Various Signature Paddings

Throughout the following sections $\{n, e\}$ will represent an RSA public key and d be the corresponding secret key. μ will alternatively denote the considered signature padding standards.

2.1 The security of ISO/IEC 9796-1 signatures

ISO/IEC-9796-1 [20] was published in 1991 by ISO as the first international standard for digital signatures. It specifies padding formats applicable to algorithms providing message recovery (algorithms are not explicit but map r bits to r bits). ISO 9796-1 is not hashing-based and back in 1998 there were no attacks [16, 18] other than factoring on this scheme ([26]: *"...*ISO 9796-1 *remains beyond the reach of all multiplicative attacks known today..."*). The scheme is used to sign messages of limited length and works as follows when n and m are respectively $N = 2\gamma + 1$ and γ-bit numbers and $\gamma = 4\ell$ is a multiple of eight:

Define by $a \cdot b$ the concatenation of a and b, let ω_i be the i-th nibble of m and denote by $s(x)$ the hexadecimal substitution table[1]:

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

[1] actually, the bits of $s(x)$ are respectively $\overline{x_3} \oplus x_1 \oplus x_0$, $\overline{x_3} \oplus x_2 \oplus x_0$, $\overline{x_3} \oplus x_2 \oplus x_1$ and $x_2 \oplus x_1 \oplus x_0$.

Letting $\bar{s}(x)$ force the most significant bit in $s(x)$ to 1 and $\tilde{s}(x)$ complement the least significant bit of $s(x)$, ISO 9796-1 specifies:

$$\begin{array}{rl} \mu(m) = & \bar{s}(\omega_{\ell-1}) \cdot \tilde{s}(\omega_{\ell-2}) \cdot \omega_{\ell-1} \cdot \omega_{\ell-2} \cdot \\ & s(\omega_{\ell-3}) \cdot s(\omega_{\ell-4}) \cdot \omega_{\ell-3} \cdot \omega_{\ell-4} \cdot \\ & \ldots \\ & s(\omega_3) \cdot s(\omega_2) \cdot \omega_3 \cdot \omega_2 \cdot \\ & s(\omega_1) \cdot s(\omega_0) \cdot \omega_0 \cdot 6_{16} \end{array}$$

[11] described an attack that applied to a slight variant of ISO 9796-1 where $\tilde{s}(x)$ was replaced by $s(x)$; this variant differed from ISO 9796-1 by one single bit. The attack was subsequently extended to the full ISO 9796-1 by Coppersmith, Halevi and Jutla [10]. More recently, in an unpublished note, François Grieu discovered a completely different attack capable of forging a new signature from three chosen ones. It is important to underline that both attacks are independent of n (once computed, the *same* forgery works with any n and the forgery can be easily adapted to different e values). A recent (unpublished) implementation of [10] used 273 chosen signatures to forge a fake one for $e = 3$.

The impact of these recent attacks on products using ISO 9796-1 depends on the application and several product analyzes reveal that system-level specifications (message contents, insufficient access to the signer *etc.*) frequently make the attack impossible.

We nevertheless underline the following:

- ISO 9796-1-based challenge-response identification with an even e can reveal the secret factors of n.
- ISO 9796-1-based public-key certificates (where the padded message is basically random-looking) can be forged if the forger submits for certification maliciously chosen binary strings (we are not aware of any existing system where such an attack is possible).
- ISO 9796-1-based e-cash applications where a valid signature of the authority on a random string chosen by the spender is a valid coin might be subject to forgery.

We consequently recommend to perform an ad-hoc audit of each existing system using ISO 9796-1 and/or use an updated version of ISO 9796-1 if the standardization group[2] actually decides to fix the standard.

2.2 The security of ISO/IEC 9796-2 signatures

ISO 9796-2 is a generic padding standard allowing total or partial message recovery. Hash-functions of different sizes are acceptable and the digest size (k_h in the standard) is consequently a variable. Section 5, note 4 of [21] recommends $64 \leq k_h \leq 80$ for total recovery (typically an ISO 10118-2 [22]) and $128 \leq k_h \leq 160$ for partial recovery.

[2] `http://www.iso.ch/jtc1/sc27`

Partial message recovery. For simplicity, assume that 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 $m = m[1] \cdot m[2]$ is separated into two parts where $m[1]$ consists of the $N - k_h - 16$ most significant bits of m and $m[2]$ of all the remaining bits of m. The padding function is:

$$\mu(m) = \texttt{6A}_{\texttt{16}} \cdot m[1] \cdot \text{HASH}(m) \cdot \texttt{BC}_{\texttt{16}}$$

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

Total message recovery. Assuming again that the hash function is known to both parties, that N and k_h are multiples of eight and that the size of m is $N - k_h - 16$, function μ is:

$$\mu(m) = \texttt{4A}_{\texttt{16}} \cdot m \cdot \text{HASH}(m) \cdot \texttt{BC}_{\texttt{16}}$$

The analysis published in [11] indicates that the choice $64 \leq k_h \leq 80$ is clearly dangerous. We consider $100 \leq k_h \leq 160$ to be within a gray area and recommend at least $k_h = 320$. The attack is again independent of the size of n (forging 1024-bit signatures is *not* harder than forging 512-bit ones) but, unlike the ISO 9796-1-attacks, forged messages are specific to a given n and can not be recycled when attacking different moduli. The rules of thumb that apply to ISO 9796-1 apply here as well.

$L = k_h$	$\log_2$ **time**	$\log_2$ **space**	recovery mode
128	54	36	partial
160	61	40	partial
64	47	30	total
80	51	34	total

Table 1. Attacks on ISO 9796-2, small public exponent.

The complexities summarized table 1 suggest a revision of this standard.

2.3 The security of DIN NI-17.4 signatures

An ISO 9796-2-variant is described in [14]. Although we know of no attacks on this scheme, one should keep in mind that [14]'s security is not formally proved and that a *related* scheme (namely ISO 9796-2) presents known security weaknesses.

2.4 The security of PKCS#1 v2.0 signatures

PKCS[3] is a *corpus* of specifications covering RSA encryption, Diffie-Hellman key agreement, password-based encryption, syntax (extended-certificates, cryptographic messages, private-key information and certification requests) and selected attributes.

[3] the acronym of *public-key cryptography standards*.

Historically, PKCS were 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 numerous products including Internet Privacy-Enhanced Mail.

The signature padding scheme defined in PKCS #1 V2.0 is:

$$\mu(m) = \mathtt{0001}_{16} \cdot \mathtt{FFFF}_{16} \ldots \mathtt{FFFF}_{16} \cdot \mathtt{00}_{16} \cdot c_{\text{SHA}} \cdot \text{SHA}(m)$$
$$\mu(m) = \mathtt{0001}_{16} \cdot \mathtt{FFFF}_{16} \ldots \mathtt{FFFF}_{16} \cdot \mathtt{00}_{16} \cdot c_{\text{MD5}} \cdot \text{MD5}(m)$$

where

$$c_{\text{SHA}} = \mathtt{3021300906052B0E03021A05000414}_{16}$$
$$c_{\text{MD5}} = \mathtt{3020300C06082A864886F70D020505000410}_{16}$$

Although the only known attacks on PKCS #1 V2.0 signatures [11] are slower than factoring, PKCS #1 V2.0's security is not formally proved. Interestingly, the two building-blocks composing PKCS #1 V2.0's signature format can be broken individually ([11] shows how to attack an unpadded hash value and [25] shows that padding the message with a fixed bit pattern is insecure if the fixed pattern is shorter than half the size of m). For the time being, no attacks capable of dealing simultaneously with both features are known.

A similar state of the art (no known attacks but also no provable security) also applies to SSL-3.02 where:

$$\mu(m) = \mathtt{0001}_{16} \cdot \mathtt{FFFF}_{16} \ldots \mathtt{FFFF}_{16} \cdot \mathtt{00}_{16} \cdot \text{MD5}(m) \cdot \text{SHA}(m)$$

and ANSI X9.31 where:

$$\mu(m) = \mathtt{6B}_{16} \cdot \mathtt{BBBB}_{16} \ldots \mathtt{BBBB}_{16} \cdot \mathtt{BA}_{16} \cdot \text{SHA}(m) \cdot \mathtt{33CC}_{16}$$

2.5 The security of FDH, PSS and PSS-R

FDH, PSS and PSS-R were published in 1996 by Bellare and Rogaway [5]. Assuming that the underlying hash functions are ideal these schemes are provably secure (*i.e.* a successful attack against FDH, PSS or PSS-R implies the ability to invert RSA). Needless to say, no attack is known on these formats.

Full domain hash, FDH In FDH, the message m is hashed into a string $f(m)$ using a hash function f that hashes uniformly into $\mathbb{Z}_n^*$. f can be easily constructed out of SHA-1 or similar hash functions as described in [3].

Probabilistic signature scheme, PSS PSS uses two hash functions g and h. To sign with PSS, the signer picks a random number r, hashes $w = h(m, r)$, and hashes w again using g to obtain a string regarded as the concatenation of two parts a and b. RSA is then applied to:

$$\mu(m, r) = \mathtt{0}_{\mathtt{02}} \cdot w \cdot (a \oplus r) \cdot b$$

To verify the signature the receiver begins by extracting w and computing a and b (by hashing the extracted w again). Then r is recovered from $a \oplus r$ and, using m (transmitted in clear along with the signature), $w = h(m, r)$ is re-computed and compared (as well as b).

Probabilistic signature scheme with recovery, PSS-R In PSS-R the message m can also be extracted from the signature. To sign using PSS-R the signer proceeds as in PSS but replaces b by $b \oplus m$:

$$\mu(m, r) = \mathtt{0_{02}} \cdot w \cdot (a \oplus r) \cdot (b \oplus m)$$

verification and message recovery are straightforward.

3 The Current Status of Various Encryption Paddings

3.1 The security of PKCS#1 v1.5 encryption

Amongst the PKCS collection, PKCS #1 V1.5 describes an 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.

PKCS(m, r') operates on strings of bytes (the plaintext m) and requires a fresh random number r' at each encryption, the standard specifies:

$$\mu(m, r') = \text{PKCS}(m, r') = \mathtt{0002_{16}} \cdot r' \cdot \mathtt{00_{16}} \cdot m$$

In 1998, Bleichenbacher published an adaptive chosen ciphertext attack on PKCS #1 V1.5 [6] capable of recovering arbitrary plaintexts from a few hundreds of thousands of ciphertexts; PKCS #1 V1.5 was subsequently updated (PKCS #1 V2.0) as described in the next session.

At an informal workshop held at Luminy a few months ago, a new attack on PKCS #1 V1.5 was disclosed. Letting N, M and R' denote the respective bit-lengths of n, m and r', and assuming that the opponent knows that the Z least significant bits of m are zeros, *i.e*:

$$\text{PKCS}(m, r') = \mathtt{0002_{16}} \cdot r' \cdot \mathtt{00_{16}} \cdot \bar{m} \cdot \overbrace{\mathtt{00\ldots00}}^{Z \text{ bits}}$$

m can be retrieved from a handful of ciphertexts provided that:

$$(e-2)\bar{M} + (e-1)R' + 10e - 34 + \log_2 e < Z$$

As recommended by RSA Laboratories, future products should use the enhanced version of the standard (PKCS #1 V2.0) described in the next section. Patches for fixing PKCS #1 V1.5 implementations and a comprehensive FAQ document covering the attack's consequences can be found at `www.rsa.com/rsalabs`.

3.2 The security of PKCS#1 v2.0 encryption

PKCS #1 V2.0 is based on *optimal asymmetric encryption* (OAEP [4]). A major feature of OAEP is its provable semantic security assuming that the hash functions used to implement it are ideal.

To encrypt m, the sender picks a random r and hashes it using a first hash-function g (denote $a = g(r)$), he then forms:

$$\mu(m, r) = (m \oplus a) \cdot (r \oplus h(m \oplus a))$$

and encrypts $\mu(m, r)$ using RSA.

To recover m, the receiver extracts the left part $(m \oplus a)$ of the padded message, hashes it with h and recovers r from the right part. Then, he computes $a = g(r)$ and recovers m from the left part of the padded message.

4 Conclusion

As a general rule of thumb and unless applicative constraints oblige to proceed otherwise, we recommend the use of PSS, PSS-R and OAEP in new designs. This trend is also being advocated by standardization groups such as IEEE 1363.

References

1. ANSI X9.31, *Digital signatures using reversible public-key cryptography for the financial services industry (rDSA)*, 1998.
2. O. Baudron and J. Stern, *To pad or not to pad: does formatting degrade security?*, 1999 RSA Data Security Conference proceeding book, 1999.
3. M. Bellare and P. Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*, Proceedings of the first annual conference on computer and communication security, ACM, 1993.
4. M. Bellare and P. Rogaway, *Optimal asymetric encryption*, Advances in cryptology EUROCRYPT'94, Springer-Verlag, Lectures notes in computer science 950, pp. 92–112, 1995.
5. M. Bellare and P. Rogaway, *The exact security of digital signatures: how to sign with RSA and Rabin*, Advances in cryptology EUROCRYPT'96, Springer-Verlag, Lectures notes in computer science 1070, pp. 399–416, 1996.
6. D. Bleichenbacher, *Chosen ciphertext attacks against protocols based on the RSA encryption standard* PKCS#1, Advances in cryptology CRYPTO'98, Springer-Verlag, Lectures notes in computer science 1462, pp. 1–12, 1998.
7. D. Boneh, *Twenty years of attacks on the RSA cryptosystem*, Notices of the American Mathematical Society (AMS), vol. 46, no. 2, pp. 203–213, 1999.
8. D. Coppersmith, M. Franklin, J. Patarin and M. Reiter, *Low exponent RSA with related messages*, Advances in cryptology EUROCRYPT'96, Springer-Verlag, Lectures notes in computer science 1070, pp. 1–9, 1996.
9. D. Coppersmith, *Finding a small root of a univariate modular equation*, Advances in cryptology EUROCRYPT'96, Springer-Verlag, Lectures notes in computer science 1070, pp. 155–165, 1996.

10. D. Coppersmith, S. Halevi, C. Jutla, ISO 9796-1 *and the new forgery strategy*, ISO/IEC/JTC1/SC27/N2362, 1999.
11. J.-S. Coron, D. Naccache, J.P. Stern, *On the security of RSA padding*, Advances in cryptology CRYPTO'99, Springer-Verlag, Lectures notes in computer science 1666, pp. 1–18, 1999.
12. G. Davida, *Chosen signature cryptanalysis of the RSA (MIT) public-key cryptosystem*, TR-CS-82-2, Department of electrical engineering and computer science, University of Wisconsin, Milwaukee, 1982.
13. Y. Desmedt and A. Odlyzko. *A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes*, Advances in cryptology CRYPTO'85, Springer-Verlag, Lectures notes in computer science 218, pp. 516–522, 1986.
14. DIN NI-17.4, *Specification of chipcard interface with digital signature application/function according to SigG and SigV*, version 1.0, 1998.
15. J. Evertse and E. van Heyst, *Which new RSA-signatures can be computed from certain given RSA signatures ?*, Journal of cryptology vol. 5, no. 1, 41–52, 1992.
16. M. Girault, J.-F. Misarsky, *Selective forgery of RSA signatures using redundancy*, Advances in cryptology EUROCRYPT'97, Springer-Verlag, Lectures notes in computer science 1233, pp. 495–507, 1997.
17. J. Gordon, *How to forge RSA key certificates*, Electronic Letters, vol. 21, no. 9, April 25-th, 1985.
18. L. Guillou, J.-J. Quisquater, M. Walker, P. Landrock and C. Shaer, *Precautions taken against various attacks in* ISO/IEC DIS 9796, Advances in cryptology EUROCRYPT'90, Springer-Verlag, Lectures notes in computer science 473, pp. 465–473, 1991.
19. K. Hickman, *The SSL Protocol*, December 1995. Available electronically at: `http://www.netscape.com/newsref/std/ssl.html`
20. ISO/IEC 9796, *Information technology - Security techniques - Digital signature scheme giving message recovery, Part 1: Mechanisms using redundancy*, 1999.
21. ISO/IEC 9796-2, *Information technology - Security techniques - Digital signature scheme giving message recovery, Part 2: Mechanisms using a hash-function*, 1997.
22. ISO/IEC 10118-2, *Information technology - Security techniques - Hash-functions; Part 2: Hash functions using an n-bit block-cipher algorithm*, 1994.
23. W. de Jonge and D. Chaum. *Attacks on some RSA signatures*, Advances in cryptology CRYPTO'85, Springer-Verlag, Lectures notes in computer science 218, pp. 18–27, 1986.
24. M. Michels, M. Stadler and H.-M. Sun, *On the security of some variants of the RSA signature scheme*, Computer security-ESORICS'98, Springer-Verlag, Lectures notes in computer science 1485, pp. 85–96, 1998.
25. J.-F. Misarsky, *A multiplicative attack using LLL algorithm on RSA signatures with redundancy*, Advances in cryptology CRYPTO'97, Springer-Verlag, Lectures notes in computer science 1294, pp. 221–234, 1997.
26. 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.
27. National Institute of Standards and Technology, *Secure hash standard*, FIPS publication 180-1, April 1994.
28. R. Rivest, *RFC 1321: The MD5 message-digest algorithm*, Internet activities board, April 1992.
29. 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.

30. RSA Laboratories, PKCS #1: *RSA cryptography specifications*, version 2.0, September 1998.