From Fixed-Length to Arbitrary-Length RSA Encoding Schemes Revisited

Julien Cathalo[1], Jean-Sébastien Coron[2], and David Naccache[2,3]

[1] UCL Crypto Group
Place du Levant 3, Louvain-la-Neuve, B-1348, Belgium
cathalo@dice.ucl.ac.be
[2] Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
{jean-sebastien.coron, david.naccache}@gemplus.com
[3] Royal Holloway, University of London
Information Security Group
Egham, Surrey TW20 0EX, UK
david.naccache@rhul.ac.uk

Abstract. To sign with RSA, one usually encodes the message m as $\mu(m)$ and then raises the result to the private exponent modulo N. In Asiacrypt 2000, Coron *et al.* showed how to build a secure RSA encoding scheme $\mu'(m)$ for signing arbitrarily long messages from a secure encoding scheme $\mu(m)$ capable of handling only fixed-size messages, without making any additional assumptions. However, their construction required that the input size of μ be larger than the modulus size. In this paper we present a construction for which the input size of μ does not have to be larger than N. Our construction shows that the difficulty in building a secure encoding for RSA signatures is not in handling messages of arbitrary length, but rather in finding a secure encoding function for short messages, which remains an open problem in the standard model.

1 Introduction

A common practice for signing with RSA is to first apply some encoding function μ to the message m, and then raise the result to the signature exponent modulo N. This is the basis of numerous standards such as ISO/IEC-9796-1 [7], ISO 9796-2 [8] and PKCS#1 V2.0 [11].

For digital signature schemes, the strongest security notion was defined by Goldwasser, Micali and Rivest in [6], as *existential unforgeability under an adaptive chosen message attack*. This notion captures the property that an attacker cannot produce a valid signature, even after obtaining the signature of (polynomially many) messages of his choice.

Many RSA encoding schemes have been designed and many have been broken (see [9] for a survey). The Full Domain Hash (FDH) scheme and the Probabilistic Signature Scheme (PSS) [3] were among the first practical and provably secure RSA signature schemes. Those schemes are provably secure in the random oracle model [2], wherein the hash function is assumed to behave as a truly random function. However, security proofs in the random oracle model are not "real" proofs, and can be only considered as heuristic, since in the real world random oracles are necessarily replaced by functions which can be computed by all parties. A famous result by Canneti, Goldreich and Halevi [4] shows that a security proof in the random oracle model does not necessarily imply security in the "real world".

In this paper, we focus on the problem of finding a secure encoding scheme for arbitrarily long messages, given a secure encoding scheme for fixed-size messages. It is well known that this can be done using a collision-resistant hash function $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$ for both signing and verifying, where ℓ is the input size of $\mu(m)$. A standard argument shows that if the original signature scheme is secure against existential forgery under a chosen-message attack, then so is the signature scheme with the hash.

In Asiacrypt 2000, Coron, Koeune and Naccache [5] showed that for RSA signatures, the same result can be obtained without assuming the existence of collision-resistant hash-functions. Namely, they construct an encoding scheme $\mu'(m)$ for messages in $\{0,1\}^*$, given an encoding scheme $\mu(m)$ for messages of fixed-size. They show that if RSA signature with $\mu(m)$ is secure against existential forgery under a chosen-message attack (in the standard model), then so is RSA with $\mu'(m)$ for messages of arbitrary size, without any additional assumptions.

However, their construction requires that the input size ℓ of $\mu(m)$ be larger than the size of N (hereafter denoted k). Several standards (for example the ISO/IEC 9796-1 standard [7]) fail to comply with this property. The authors left as an open problem the case $\ell \leq k$.

In this paper, we solve this open problem and provide a construction for any input size ℓ. A variant of this problem was already solved by Arboit and Robert in [1], who proposed a construction similar to [5] that works for any ℓ, but at the cost of a new security assumption, namely the division intractability of the encoding function $\mu(m)$. The advantage of our construction is that we do not make any additional assumptions, namely if RSA signature with $\mu(m)$ is secure against existential forgery under a chosen-message attack, then so is RSA with $\mu'(m)$ for messages of arbitrary size. As is the case for the constructions in [5] and [1], a practical advantage of our construction is that it allows to perform some pre-computations on partially received messages, *e.g.* on IP packets which are typically received in random order.

We believe that our result focuses more sharply the question of finding a secure encoding for RSA signatures, by showing that the difficulty is not in handling messages of arbitrary length, but rather in finding a securing encoding for short messages, which remains an open problem in the standard model.

2 Definitions

2.1 Signature Schemes

The digital signature of a message m is a string that depends on m and on some secret known only to the signer, in such a way that anyone can check the validity of the signature. The following definitions are based on [6].

Definition 1 (Signature scheme). *A signature scheme is defined by the following:*

- *The* key generation algorithm $\mathsf{Generate}$ *is a probabilistic algorithm which given* 1^k*, outputs a pair of matching public and secret keys,* $(\mathsf{pk}, \mathsf{sk})$*.*
- *The* signing algorithm Sign *takes the message* M *to be signed and the secret key* sk *and returns a signature* $x = \mathsf{Sign}_{\mathsf{sk}}(M)$*. The signing algorithm may be probabilistic.*
- *The* verification algorithm Verify *takes a message* M*, a candidate signature* x' *and the public key* pk*. It returns a bit* $\mathsf{Verify}_{\mathsf{pk}}(M, x')$*, equal to one if the signature is accepted, and zero otherwise. We require that if* $x \leftarrow \mathsf{Sign}_{\mathsf{sk}}(M)$*, then* $\mathsf{Verify}_{\mathsf{pk}}(M, x) = 1$*.*

2.2 Security of Signature Schemes

The security of signature schemes was formalized in an asymptotic setting by Goldwasser, Micali and Rivest [6]. Here we use the definitions of [3] which provide a framework for the concrete security analysis of digital signatures. Resistance against adaptive chosen-message attacks is considered: a forger $\mathcal{F}$ can dynamically obtain signatures of messages of its choice and attempt to output a valid forgery. A *valid forgery* is a message/signature pair (M, x) such that $\mathsf{Verify}_{\mathsf{pk}}(M, x) = 1$ whilst the signature of M was never requested by $\mathcal{F}$.

Definition 2. *A forger* $\mathcal{F}$ *is said to* $(t, q_{sig}, \varepsilon)$*-break the signature scheme* ($\mathsf{Generate}$, Sign, Verify) *if after at most* $q_{sig}(k)$ *signature queries and* $t(k)$ *processing time, it outputs a valid forgery with probability at least* $\varepsilon(k)$ *for any* $k > 0$*.*

Definition 3. *A signature scheme* ($\mathsf{Generate}$, Sign, Verify) *is* $(t,\ q_{sig},\ \varepsilon)$*-secure if there is no forger who* $(t, q_{sig}, \varepsilon)$*-breaks the scheme.*

2.3 The RSA Primitive

RSA [10] is the most widely used public-key cryptosystem. It can be used to provide both encryption schemes and digital signatures.

Definition 4 (The RSA cryptosystem). *RSA is a family of trapdoor permutations. It is 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 $N = p \cdot q$. It randomly picks an encryption exponent $e \in \mathbb{Z}^*_{\phi(N)}$ and computes the corresponding decryption exponent d such that $e \cdot d = 1 \bmod \phi(N)$. The generator returns $\{N, e, d\}$.*
- *The encryption function $f : \mathbb{Z}^*_N \rightarrow \mathbb{Z}^*_N$ defined by $f(x) = x^e \bmod N$.*
- *The decryption function $f^{-1} : \mathbb{Z}^*_N \rightarrow \mathbb{Z}^*_N$ defined by $f^{-1}(y) = y^d \bmod N$.*

2.4 RSA Encoding and Signature

Let μ be a encoding function taking as input a message of size ℓ bits and returning a k-bit integer. We consider in figure 1 the classical RSA signature scheme which signs fixed-length ℓ-bits messages.

System parameters
- two integers $k > 0$ and $\ell > 0$
- a function $\mu : \{0,1\}^\ell \rightarrow \{0,1\}^k$

Key generation: Generate
- $(N, e, d) \leftarrow RSA(1^k)$
- public key : (N, e)
- private key : (N, d)

Signature generation: Sign
- let $y \leftarrow \mu(m)$
- return $y^d \bmod N$

Signature verification: Verify
- let $y \leftarrow x^e \bmod N$
- let $y' \leftarrow \mu(m)$
- if $y = y'$ then return one else return zero.

Fig. 1. The Classical RSA Paradigm: Using μ for Signing Fixed-Length Messages.

3 The Coron-Koeune-Naccache Construction

We recall in figure 2 the construction proposed in [5]. It assumes that the encoding function μ can handle inputs of size $k+1$ where k is the size of the modulus and allows to sign $2^a \cdot (k-a)$ bit messages where $0 \leq a \leq k-1$. The construction can be recursively iterated to sign messages of arbitrary length. Throughout this paper, $m_1||m_2$ will denote the concatenation of m_1 and m_2.

It is shown in [5] that the scheme described in figure 2 is secure against existential forgery under a chosen message attack :

System parameters
- two integers $k > 0$ and $a \in [0, k-1]$
- a function $\mu : \{0,1\}^{k+1} \rightarrow \{0,1\}^k$

Key generation: Generate*
- $(N, e, d) \leftarrow RSA(1^k)$
- public key : (N, e)
- private key : (N, d)

Signature generation: Sign*
- Split the message m into $(k-a)$-bit blocks such that $m = m[1]||\ldots||m[r]$.
- let $\alpha = \prod_{i=1}^{r} \mu(0||i||m[i]) \bmod N$ where i in $0||i||m[i]$ is an a-bit string representing i.
- let $y \leftarrow \mu(1||\alpha)$
- return $y^d \bmod N$

Verification: Verify*
- let $y \leftarrow x^e \bmod N$
- let $\alpha = \prod_{i=1}^{r} \mu(0||i||m[i]) \bmod N$
- let $y' \leftarrow \mu(1||\alpha)$
- if $y = y'$ then return one else return zero.

Fig. 2. Coron-Koeune-Naccache Encoding of Arbitrary Length Messages.

Theorem 1. *If the signature scheme* (Generate, Sign, Verify) *is* $(t, q_{sig}, \varepsilon)$ *secure, then the signature scheme* (Generate*, Sign*, Verify*) *which signs* $2^a \cdot (k-a)$ *bit messages is* $(t^*, q^*_{sig}, \varepsilon^*)$ *secure, where:*

$$t^*(k) = t(k) - 2^a \cdot q_{sig}(k) \cdot \mathcal{O}(k^2)\,, \tag{1}$$

$$q^*_{sig}(k) = q_{sig}(k) - 2^{a+1}\,, \tag{2}$$

$$\varepsilon^*(k) = \varepsilon(k)\ . \tag{3}$$

4 Bimodular Encoding

The drawback of the previous construction is that the the input size ℓ of $\mu(m)$ needs to be larger than the size of the modulus N. In this section, we describe a construction wherein the input size ℓ of the encoding function μ does not need larger than k. We denote by $\ell(k)$ the input size of the encoding function μ as a function of the security parameter k. In the following, we assume that $\ell(k)$ is an increasing function of k. For example, for the ISO/IEC 9796-1 standard [7], we have $\ell(k) \simeq k/2$.

The new signature scheme (Generate', Sign', Verify') is described in figure 3. The new signature scheme is parameterized by two security parameters k_1, k_2

System parameters
- two positive integers k_1, k_2 such that $k_2 > k_1$
- an integer $a \in [0, k_1 - 1]$
- two functions $\mu_i : \{0,1\}^{\ell_i} \to \{0,1\}^{k_i}$ for $i = 1, 2$ such that $\ell_2 \geq k_1$.

Key generation: Generate$'$
- $(N_1, e_1, d_1) \leftarrow RSA(1^{k_1})$
- $(N_2, e_2, d_2) \leftarrow RSA(1^{k_2})$
- public key : (N_1, N_2, e_2)
- private key : (N_1, N_2, d_2)

Signature generation: Sign$'$
- Split the message m into $(\ell_1 - a)$-bit blocks such that $m = m[1]||\ldots||m[r]$.
- let $\alpha = \prod_{i=1}^{r} \mu_1(i||m[i]) \bmod N_1$ where i in $i||m[i]$ is an a-bit string representing i.
- let $y \leftarrow \mu_2(\alpha)$
- return $y^{d_2} \bmod N_2$

Verification: Verify$'$
- $y \leftarrow x^{e_2} \bmod N_2$
- let $\alpha = \prod_{i=1}^{r} \mu_1(i||m[i]) \bmod N_1$
- let $y' \leftarrow \mu_2(\alpha)$
- if $y = y'$ then return one else return zero.

Fig. 3. Bimodular Encoding of Arbitrary Length Messages.

such that $k_1 < k_2$. As the previous construction, it is a deterministic signature scheme. The construction uses the same encoding function μ with two distinct moduli N_1 and N_2 of sizes k_1 and k_2 bits, respectively. For the sake of clarity and since encoding functions take the modulus as a parameter, we will write μ_i when μ is used with modulus N_i. We denote by $\ell_1 = \ell(k_1), \ell_2 = \ell(k_2)$ the input sizes of μ_1, μ_2 respectively. Our construction requires that $\ell_2 \geq k_1$. Since by assumption $\ell(k)$ is an increasing function of k, this means that a sufficiently large security parameter k_2 must be selected in order to have $\ell_2 = \ell(k_2) \geq k_1$

Our construction enables to sign $2^a \cdot (\ell_1 - a)$ bit messages where $0 \leq a \leq \ell_1 - 1$. The maximum length that can be handled by the new construction is therefore $2^{\ell_1 - 1}$ bits for $a = \ell_1 - 1$ or $a = \ell_1 - 2$ and, as in [5], the construction can be recursively iterated so as to sign arbitrarily long messages.

A possible realization example is the following: assume that we are given an encoding function μ that takes as input $k/2$-bit messages and outputs k-bit strings, for signing with a k-bit RSA modulus. If we take for example $k_1 = 1024$, $k_2 = 2048$ and $a = 24$, then messages of size up to $2^{24} \cdot 488 \simeq 8.2 \cdot 10^9$ bits can be signed. First, one applies the encoding function $\mu_1 : \{0,1\}^{512} \to \{0,1\}^{1024}$ to the 2^{24} blocks of 488 bits; then one multiplies together the resulting 1024-bit integers modulo N_1 and obtains a 1024-bit integer which is finally signed using the encoding function $\mu_2 : \{0,1\}^{1024} \to \{0,1\}^{2048}$ modulo N_2. Notice that d_1 is not used for signing and e_1 is not needed for the verification either; thus (e_1, d_1) is to be deleted after the generation of N_1.

The following theorem states that this construction preserves the resistance against chosen message attacks of the original signature scheme:

Theorem 2. *If the signature scheme* (Generate, Sign, Verify) *is* $(t, q_{sig}, \varepsilon)$ *secure, then the signature scheme* (Generate', Sign', Verify') *which signs* $2^a \cdot (\ell_1 - a)$ *bit messages is* $(t', q'_{sig}, \varepsilon')$ *secure, where:*

$$t'(k_1, k_2) = t(k_1) - q'_{sig} \cdot 2^a \cdot \left(T_\mu(k_2) + \mathcal{O}({k_2}^3)\right) \ , \tag{4}$$

$$q'_{sig}(k_1, k_2) = q_{sig}(k_1) - 2^{a+1} \ , \tag{5}$$

$$\varepsilon'(k_1, k_2) = 4 \cdot \varepsilon(k_1) \ . \tag{6}$$

and $T_\mu(k_2)$ *is the time required to compute* $\mu(m)$ *for security parameter* k_2*.*

Proof. Without loss of generality, we can assume that $t(k)$, $q_{sig}(k)$ and $T_\mu(k)$ are increasing functions of k, and that $\varepsilon(k)$ is a decreasing function of k.

Let $\mathcal{F}'$ be a forger that breaks the signature scheme (Generate', Sign', Verify') for the parameters (k_1, k_2). We construct a forger $\mathcal{F}_1$ for the signature scheme (Generate, Sign, Verify) for the parameter $k = k_1$ and a forger $\mathcal{F}_2$ for same signature scheme with parameter $k = k_2$. When the same property holds for both $\mathcal{F}_1$ and $\mathcal{F}_2$, we write this property for a generic forger $\mathcal{F}$. The forger $\mathcal{F}$ will run $\mathcal{F}'$ in order to produce a forgery; it will answer the signature queries of $\mathcal{F}'$ by itself. $\mathcal{F}$ has access to a signing oracle $\mathcal{S}$ for (Generate, Sign, Verify) .

First, we pick a random bit b. If $b = 1$, we construct a forger $\mathcal{F}_1$ for the parameter $k = k_1$. If $b = 0$, we construct a forger $\mathcal{F}_2$ for the parameter $k = k_2$.

$\mathcal{F}$ is first given as input (N, e) where N, e were obtained by running Generate for the parameter k defined previously. The forger $\mathcal{F}$ then starts running $\mathcal{F}'$ with the public key (N_1, N_2, e_2), where N_1, N_2, e_2 are defined as follows :

If $b = 1$, the forger $\mathcal{F}_1$ sets $N_1 \leftarrow N$, $e_1 \leftarrow e$ and runs $RSA(1^{k_2})$ to obtain (N_2, e_2, d_2). Otherwise (if $b = 0$) the forger $\mathcal{F}_2$ sets $N_2 \leftarrow N$, $e_2 \leftarrow e$ and runs $RSA(1^{k_1})$ to obtain (N_1, e_1, d_1).

We observe that the view of the forger $\mathcal{F}'$ in independent of the bit b, since in both cases the moduli N_1 and N_2 are generated using $RSA(1^{k_1})$ and $RSA(1^{k_2})$, either by $\mathcal{F}$ itself or through (N, e) given as input to $\mathcal{F}$.

When $\mathcal{F}'$ asks the signature of the j-th message m_j with $m_j = m_j[1]||\ldots||$ $m_j[r_j]$, $\mathcal{F}$ computes:

$$\alpha_j = \prod_{i=1}^{r_j} \mu_1(i||m_j[i]) \bmod N_1$$

If $b = 0$ then $\mathcal{F}_2$ requests the signature s_j of α_j from $\mathcal{S}$. If $b = 1$ then $\mathcal{F}_1$ can compute $s_j = \mu_2(\alpha_j)^{d_2} \bmod N_2$ directly since it knows d_2. Let q'_{sig} be the total number of signatures requested by $\mathcal{F}'$.

Eventually $\mathcal{F}'$ outputs a forgery (m', s') for the signature scheme (Generate', Sign', Verify') with $m' = m'[1]||\ldots||m'[r']$, from which $\mathcal{F}$ computes:

$$\alpha' = \prod_{i=1}^{r'} \mu_1(i||m'[i]) \bmod N_1 \tag{7}$$

We denote by β the probability that $\alpha' \notin \{\alpha_1, \ldots, \alpha_q\}$. Note that since the view of $\mathcal{F}'$ is independent of b, this event is independent of b as well. We distinguish three cases:

First case: $\alpha' \notin \{\alpha_1, \ldots, \alpha_q\}$ and $b = 0$. From the remark above, this happens with probability $\beta/2$. In which case $\mathcal{F}_2$ outputs the forgery (α', s') and halts. This is a valid forgery for the signature scheme (Generate, Sign, Verify) since $s' = \mu_2(\alpha')^{d_2} \bmod N_2$ and the signature of α' was never asked to the signing oracle $\mathcal{S}$.

Second case: $\alpha' \in \{\alpha_1, \ldots, \alpha_q\}$ and $b = 1$. This happens with probability $(1-\beta)/2$. Let c be such that $\alpha = \alpha_c$. We write $m = m_c$, $\alpha = \alpha_c$ and $r = r_c$, which gives using (7) :

$$\prod_{i=1}^{r'} \mu_1(i||m'[i]) \bmod N_1 = \prod_{i=1}^{r} \mu_1(i||m[i]) \bmod N_1 \tag{8}$$

We show that the previous equation leads to a multiplicative forgery for the modulus $N_1 = N$, which enables $\mathcal{F}_1$ to compute a forgery.

First, the message m' must be distinct from m because the signature of m has been requested by $\mathcal{F}'$ whereas the signature of m' was never requested by $\mathcal{F}$, since m' is the message for which a forgery was obtained. Consequently there exists an integer j such that either :

$$j||m'[j] \notin \{1||m[1], \ldots, r||m[r]\} \tag{9}$$

or :

$$j||m[j] \notin \{1||m'[1], \ldots, r'||m'[r']\} \tag{10}$$

We assume that condition (9) is satisfied (condition (10) leads to the same result). Therefore from (8) we can write :

$$\mu(j||m'[j]) = \Big(\prod_{i} \mu(i||m[i]\Big)\Big(\prod_{i \neq j} \mu(i||m'[i])\Big)^{-1} \bmod N_1 \tag{11}$$

Consequently, $\mathcal{F}_1$ asks the signing oracle $\mathcal{S}$ for the signatures x_i of the messages $i||m[i]$, $1 \leq i \leq r$, and for the signatures x'_i of the messages $i||m'[i]$, $1 \leq i \leq r'$, $i \neq j$. Using (11), $\mathcal{F}_1$ can compute the signature of $j||m'[j]$ from the other signatures :

$$x'_j = \mu(j||m'[j])^{d_1} = \Big(\prod_{i} x_i\Big)\Big(\prod_{i \neq j} x'_i\Big)^{-1} \bmod N_1$$

and $\mathcal{F}_1$ finally outputs the forgery $(j||m'[j], x'_j)$. This is a valid forgery for the signature scheme (Generate, Sign, Verify) since the signature of $j||m'[j]$ was never asked to the signing oracle.

Third case: $\alpha' \notin \{\alpha_1, \ldots, \alpha_q\}$ and $b = 1$, or $\alpha' \in \{\alpha_1, \ldots, \alpha_q\}$ and $b = 0$. In this case, $\mathcal{F}$ fails. This happens with probability 1/2.

To summarize, from a forger $\mathcal{F}'$ that breaks the signature scheme (Generate', Sign', Verify') with probability $\varepsilon'(k_1, k_2)$ for the parameters (k_1, k_2), we construct a forger $\mathcal{F}$ that breaks the signature scheme (Generate, Sign, Verify) with probability $\varepsilon' \cdot \beta/2$ for the parameter k_2, and with probability $\varepsilon' \cdot (1-\beta)/2$ for the parameter k_1, for some (unknown) β.

Therefore, if we assume that the signature scheme (Generate, Sign, Verify) cannot be broken in time $t(k)$ with probability greater than $\varepsilon(k)$ for all k, we must have :

$$\varepsilon'(k_1, k_2) \cdot \beta/2 \leq \varepsilon(k_2)$$

and

$$\varepsilon'(k_1, k_2) \cdot (1-\beta)/2 \leq \varepsilon(k_1)$$

which implies using $\varepsilon(k_2) \leq \varepsilon(k_1)$ that :

$$\varepsilon'(k_1, k_2) \leq 4 \cdot \varepsilon(k_1)$$

which gives (6).

If $b = 0$, then for each of the q'_{sig} queries of $\mathcal{F}'$, the forger $\mathcal{F}_2$ makes at most 2^a multiplications modulo N_1 and one query to $\mathcal{S}$. Thus $\mathcal{F}_2$ runs in time

$$t(k_2) = t'(k_1, k_2) + q'_{sig} \cdot 2^a \cdot \left(T_\mu(k_1) + \mathcal{O}({k_1}^2)\right) \tag{12}$$

If $b = 1$ then for each query of $\mathcal{F}'$, the forger $\mathcal{F}_1$ makes at most 2^a multiplications modulo N_1 and one exponentiation modulo N_2. After it has received the forgery, it makes at most 2^{a+1} multiplications modulo N_1 to compute its own forgery. Thus $\mathcal{F}_1$ runs in time :

$$t(k_1) = t'(k_1, k_2) + q'_{sig} \cdot \left(2^a \cdot \left(T_\mu(k_1) + \mathcal{O}({k_1}^2)\right) + T_\mu(k_2) + \mathcal{O}({k_2}^3)\right) \tag{13}$$

From inequalities (12) and (13), and using $t(k_1) \leq t(k_2)$ and $T_\mu(k_1) \leq T_\mu(k_2)$ we obtain (4).

Finally, the forger $\mathcal{F}_2$ makes at most q'_{sig} queries to the signing oracle, and the forger $\mathcal{F}_1$ makes at most 2^{a+1} queries to the signing oracle. This gives $q_{sig}(k_2) \leq q'_{sig}(k_1, k_2)$ and $q_{sig}(k_1) \leq 2^{a+1}$. Using $q_{sig}(k_1) \leq q_{sig}(k_2)$, we obtain

$$q_{sig}(k_1) \leq 2^{a+1} + q'_{sig}(k_1, k_2),$$

which gives (5). $\square$

5 Conclusion

In this paper, we showed how to construct a secure RSA encoding scheme for signing arbitrarily long messages, given any secure encoding scheme for signing fixed-size messages. This solves a problem left open by Coron *et al.* in [5]. We believe that our work focuses the question of finding a secure encoding for RSA signatures, by showing that the difficulty in building secure encoding schemes for RSA is not in handling messages of arbitrary length, but rather in finding a secure redundancy function for short messages, which remains an open problem in the standard model.

References

1. G. Arboit and J.M. Robert, *From Fixed-Length to Arbitrary-Length Messages Practical RSA Signature Padding Schemes*, in LNCS 2020 – Topics in Cryptology CT-RSA 2001, Springer-Verlag, p. 44-51.
2. M. Bellare and P. Rogaway, *Random oracles are practical : a paradigm for designing efficient protocols*, proceedings of the First Annual Conference on Computer and Commmunications Security, ACM, 1993.
3. M. Bellare and P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*, proceedings of Eurocrypt'96, LNCS vol. 1070, Springer-Verlag, 1996, pp. 399-416.
4. R. Canetti, O. Goldreich and S. Halevi, *The Random Oracle Methodology, Revisited*, STOC '98, ACM, 1998.
5. J.S. Coron, F. Koeune, D. Naccache, *From fixed-length to arbitrary-length RSA padding schemes*, Proceedings of Asiacrypt 2000, LNCS vol. 1976, Springer-Verlag, 2000.
6. S. Goldwasser, S. Micali and R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal of computing, 17(2):281-308, april 1988.
7. ISO/IEC 9796, *Information technology - Security techniques - Digital signature scheme giving message recovery, Part 1 : Mechanisms using redundancy*, 1999.
8. ISO/IEC 9796-2, *Information technology - Security techniques - Digital signature scheme giving message recovery, Part 2 : Mechanisms using a hash-function*, 1997
9. J.F. Misarsky, *How (not) to design signature schemes*, proceedings of PKC'98, Lecture Notes in Computer Science vol. 1431, Springer Verlag, 1998.
10. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, CACM 21, 1978.
11. RSA Laboratories, PKCS #1 *: RSA cryptography specifications*, version 2.0, September 1998.