From Fixed-Length Messages to Arbitrary-Length Messages Practical RSA Signature Padding Schemes

Geneviève Arboit\(^1\) * and Jean-Marc Robert\(^2\)

\(^1\) School of Computer Science, McGill University, Montréal, CANADA
garboit@cs.mcgill.ca
\(^2\) Gemplus Card International, Montréal R&D Center, CANADA
jean-marc.robert@gemplus.com

Abstract. We show how to construct a \textit{practical} secure signature padding scheme for arbitrarily long messages from a secure signature padding scheme for fixed-length messages. This new construction is based on a one-way compression function respecting the division intractability assumption. By practical, we mean that our scheme can be instantiated using dedicated compression functions and without chaining. This scheme also allows precomputations on partially received messages. Finally, we give an instantiation of our scheme using SHA-1 and PKCS #1 ver. 1.5.

\textbf{Keywords}: Digital signature, padding scheme, provable security, atomic primitive, RSA, hash-and-sign, division intractability, smooth numbers.

1 Introduction

A common practice for signing with \textbf{RSA} is known as the hash-and-sign paradigm. First, a hash or redundancy function, which usually consists of a compression function and a chaining function, is applied to the message. Then some padding is added to the result, and this value is exponentiated using the signature exponent. This is the basis of several existing standards many of which have been broken (see [Mis98] for a survey).

Security reductions for \textbf{RSA} signature padding schemes are presented in [CKN00]. These reductions permit to go from fixed-length messages to arbitrary-length messages \textbf{RSA} signature padding schemes. Moreover, these new schemes also allow one to make precomputations on partially received messages, as in the case of IP packets, which are typically received in a random order. In [CKN00], a hash function \(\mu\) is an atomic primitive that is assumed to be a secure padding scheme for \textbf{RSA}. However, \(\mu\) takes a \(k + 1\) bit input and returns a \(k\) bit output where \(k\) is the length of the \textbf{RSA} modulus. This particularity of the scheme is not significantly modifiable: the bit length of the \(\mu\) output has to have about the same bit length as the \textbf{RSA} modulus. This limitation on the choice of \(\mu\) forces either to instantiate it with a non-dedicated hash function, or with a dedicated hash function that uses both compression and chaining primitives.

In this paper, with a similar construction, we give a practical instantiation based on the compression function of SHA-1 without any chaining function. Our solution has the great advantage over [CKN00] of removing the relation of the length of \(\mu\) output to the length of the \textbf{RSA} modulus. We are able to achieve this result simply by making an additional assumption about \(\mu\), namely division intractability. This property is slightly stronger than collision intractability.

2 Definitions

2.1 Signature schemes

The following definitions are based on [GMR88].

\* This work was done while visiting Gemplus Montréal R&D Center.
Definition 1. A digital signature scheme is defined by the following:

- The key generation algorithm Gen is a probabilistic algorithm which given $1^k$, outputs a pair of matching public and secret keys $(pk, sk)$.
- The signing algorithm Sign takes the message $m$ to be signed and the secret key $sk$ and returns a signature $s = \text{Sign}_{sk}(m)$. The signing algorithm may be probabilistic.
- The verification algorithm Verify takes a message $m$, a candidate signature $s'$ and the public key $pk$. It returns a bit $\text{Verify}(m, s')$, equal to 1 if the signature is accepted, and 0 otherwise. We require that if $\text{Sign}_{sk}(m)$ was indeed assigned to $s$, then $\text{Verify}(m, s) = 1$.

2.2 Security of signature schemes

The security of signature schemes was formalized in [GMR88] for the asymptotic setting. We will prove security against existential forgery by adaptive chosen plaintext attackers. The following definitions for the exact security of signature schemes are taken from [BR96].

Definition 2. A forging algorithm $F$ is said to $(t_{\text{proc}}, q_{\text{sign}}, \varepsilon)$-break the signature scheme given by $(\text{Gen}, \text{Sign}, \text{Verify})$ if after at most $q_{\text{sign}}$ adaptively chosen signature queries and $t_{\text{proc}}$ processing time, it outputs a valid forgery with probability at least $\varepsilon$. The probability is taken over the random bits of $F$, and given that the random bits in the signature are correctly distributed.

Definition 3. A signature scheme $(Gen, Sign, Verify)$ is $(t_{\text{proc}}, q_{\text{sign}}, \varepsilon)$-secure if there is no forging algorithm which $(t_{\text{proc}}, q_{\text{sign}}, \varepsilon)$-breaks the scheme.

To construct $\mu$ from a dedicated hash function without chaining, we make an additional assumption, which is strong but constructible. We use a definition of [GHR99] slightly modified for our purposes.

Definition 4. Let $H_t$ be a collection of compression functions that map strings of length $t$ into strings of length $l$. Such a collection is said to be division intractable if for $\mu \in H_t$, it is infeasible to find distinct inputs $X_1, \ldots, X_n, Y$ such that $\mu(Y)$ divides the product of the $\mu(X_i)$'s. Formally, for every probabilistic polynomial time algorithm $A$, there exists a negligible function $\text{negl}(1)$ such that:

$$\Pr_{\mu \in H_t} \left[ A(\mu) = \langle X_1, \ldots, X_n, Y \rangle \right. \left. \begin{array}{l} 
\text{s.t. } Y \neq X_i \text{ for } i = 1, \ldots, n, \\
\text{and } \mu(Y) \text{ divides the product } \prod_{i=1}^{n} \mu(X_i) \text{ mod } 2^t \right] = \text{negl}(1)$$

If $\mu$ is randomized, the adversary $A$ can choose both the input and the randomness. Given a randomly chosen function $\mu$ from $H_t$, $A$ needs to find pairs $(R_1, X_1), \ldots, (R_n, X_n), (R, Y)$ such that $Y \neq X_i$ for $i = 1, \ldots, n$, but $\mu(R, Y)$ divides the product $\prod_{i=1}^{n} \mu(R_i, X_i) \text{ mod } 2^t$.

2.3 The RSA cryptosystem

The RSA cryptosystem can be used to obtain both public key cryptosystems and digital signatures [RSA78].

Definition 5. The RSA cryptosystem is a family of trapdoor permutations. It is specified by:

- The RSA generator RSA, which on input $1^k$, randomly selects 2 distinct $k/2$-bit primes $p$ and $q$ and computes the modulus $N = p \cdot q$. It randomly picks an encryption exponent $e \in Z_\phi(N)$ and computes the corresponding decryption exponent $d$ such that $e \cdot d = \text{mod } \phi(N)$. The generator returns $(N, e, d)$.
- The encryption function $f : Z_N^* \rightarrow Z_N^*$ defined by $f(x) = x^e \text{ mod } N$.
- The decryption function $f^{-1} : Z_N^* \rightarrow Z_N^*$ defined by $f^{-1}(y) = y^d \text{ mod } N$. 

2.4 A practical standard RSA signature scheme

Let \( \mu \) be a randomized compression function taking as input a message of size \( t \), using \( r \) random bits, and outputting a message digest of length \( l \):

\[
\mu : \{0,1\}^r \times \{0,1\}^l \rightarrow \{0,1\}^l
\]

and \( \text{enc} \) be an encoding function taking as input a message digest of size \( l \), and outputting an encoded message of length \( k \):

\[
\text{enc} : \{0,1\}^l \rightarrow \{0,1\}^k
\]

Overall:

\[
\text{enc} \circ \mu : \{0,1\}^r \times \{0,1\}^l \rightarrow \{0,1\}^k
\]

We consider in Figure 1 the classical RSA signature scheme \((\text{Gen}, \text{Sign}, \text{Verify})\) which signs fixed-length \( t \)-bits messages. This is a modification of [CKN00, Figure 1].

<table>
<thead>
<tr>
<th>SYSTEM PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>an integer ( k &gt; 0 )</td>
</tr>
<tr>
<td>a function ( \mu : {0,1}^r \times {0,1}^l \rightarrow {0,1}^l )</td>
</tr>
<tr>
<td>a function ( \text{enc} : {0,1}^l \rightarrow {0,1}^k )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KEY GENERATION: Gen</th>
</tr>
</thead>
<tbody>
<tr>
<td>((N,e,d) \leftarrow \text{RSA}(1^k))</td>
</tr>
<tr>
<td>public key: ((N,e))</td>
</tr>
<tr>
<td>private key: ((N,d))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SIGNATURE GENERATION: Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R \leftarrow \mu {0,1}^r )</td>
</tr>
<tr>
<td>( m \in {0,1}^l )</td>
</tr>
<tr>
<td>( y \leftarrow \text{enc} \circ \mu(R,m) )</td>
</tr>
<tr>
<td>return ( \langle R, y^d \bmod N \rangle )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SIGNATURE VERIFICATION: Verify</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y \leftarrow x^e \bmod N )</td>
</tr>
<tr>
<td>( y' \leftarrow \text{enc} \circ \mu(R,m) )</td>
</tr>
<tr>
<td>if ( y = y' ) then return 1 else return 0</td>
</tr>
</tbody>
</table>

Fig. 1. The classical RSA scheme using \( \text{enc} \circ \mu \) for signing fixed-length messages

3 The improved construction

We construct in Figure 2 a new signature scheme \((\text{Gen}', \text{Sign}', \text{Verify}')\) using the function \( \text{enc} \circ \mu \). The new construction allows the signing of messages of size \( 2^a(t-a) \) bits where \( a \) is between 0 and \( t-1 \). This is a modification of [CKN00, Figure 2].
**Theorem 1.** Fix $\varepsilon$ such that for all $\text{negl}(l)$ functions, $\varepsilon > \text{negl}(l)$, and suppose that $q_{\text{sign}}$ and $t_{\text{proc}}$ are polynomial in $l$. For a fixed $\text{negl}(l)$ function, if the signature scheme $(\text{Gen}, \text{Sign}, \text{Verify})$ is $(t_{\text{proc}}, q_{\text{sign}}, \varepsilon)$-secure and if $\mu$ is $\text{negl}(l)$-division intractable, then the signature scheme described in Fig. 2 $(\text{Gen}', \text{Sign}', \text{Verify}')$ is $(t'_{\text{proc}}, q_{\text{sign}}, \varepsilon)$-secure, where:

$$t'_{\text{proc}} = t_{\text{proc}} - 2^a \cdot q_{\text{sign}} \cdot O(t^2)$$

**proof:** Suppose there is a forger $F'$ that $(t'_{\text{proc}}, q_{\text{sign}}, \varepsilon)$-breaks the scheme $(\text{Gen}', \text{Sign}', \text{Verify}')$. Then, we can construct a forger $F$ that $(t_{\text{proc}}, q_{\text{sign}}, \varepsilon)$-breaks the scheme $(\text{Gen}, \text{Sign}, \text{Verify})$ using $F'$. The forger $F$ has oracle access to a signer $S$ for the scheme $(\text{Gen}, \text{Sign}, \text{Verify})$ and its goal is to produce a forgery for $(\text{Gen}, \text{Sign}, \text{Verify})$.

The forger $F$ answers the queries of $F'$. When $F'$ needs the signature of the $j^\text{th}$ message $m^j$, $F$ queries $S$ to obtain the signature $s_j$ of $a_j$ (refer to Fig. 2).

Eventually, $F'$ outputs a forgery $(m', s')$ for the signature scheme $(\text{Gen}', \text{Sign}', \text{Verify}')$, from which $F$ computes, for $j = 1, ..., q_{\text{sign}}$:

$$a_j = \prod_{i=1}^{b_j} \mu(R_i^j || m^j[i]) \bmod 2^t$$

$$a' = \prod_{i=1}^{y'} \mu(R_i^j || m'[i]) \bmod 2^t$$

which takes additional time $\sum_{j=1}^{q_{\text{sign}}} b_j + y' \leq q_{\text{sign}} \cdot 2^{a+1}$, multiplied by the time necessary to compute multiplications modulo $2^t$, which is in time quadratic in $t$ (upper bound).

We distinguish two cases:

**First case:** $a' \notin \{a_1, ..., a_{q_{\text{sign}}}, s\}$. In this case, $F$ outputs the forgery $(a', s')$ and halts. This is a valid forgery for the signature scheme $(\text{Gen}, \text{Sign}, \text{Verify})$ since $s' = \langle R', (\text{enc} \circ \mu(R', a'))^d \bmod N \rangle$.
and $\alpha'$ was never signed by the signer $S$. This contradicts our assumption that the signature scheme is secure.

**Second case:** $\alpha' \in \{\alpha_1, \ldots, \alpha_{q_{\text{sig}}}\}$, so there exists a $c$ such that $\alpha' = \alpha_c$. Let us denote $m = m_c$, $(R_i) = (R_i^c)$, $\alpha = \alpha_c$, and $b = b_c$. We have:

$$\prod_{i=1}^{b'} \mu(R_i, i||m'[i]) \mod 2^t = \prod_{i=1}^{b} \mu(R_i, i||m[i]) \mod 2^t$$

Wlog, suppose $b' = b$. Since $m' \neq m$, for some $i$ we have $i||m'[i] \notin \{1||m[1], \ldots, b||m[b]\}$ and $\mu(R_i, i||m[i])$ divides the product $\prod_{i=1}^{b} \mu(R_i, i||m[i]) \mod 2^t$. Since $\epsilon > \text{negl}(l)$, and $t'$ and $q_{\text{sig}}$ are polynomial in $l$, this contradicts our assumption that $\mu$ is $\text{negl}(l)$-division intractable. 

### 4 Implementing a practical hashing family $H_{3\cdot160}$

We define the function $\mu$ by using the following standard dedicated primitives:

$$h = \text{SHA-1} : \{0,1\}^{512} \rightarrow \{0,1\}^{160}$$

$$\text{enc} = \text{PKCS \#1 ver. 1.5} : \{0,1\}^{140} \rightarrow \{0,1\}^k$$

and where $\text{enc} = \text{PKCS \#1 ver. 1.5}$ is a feasible randomized alternative.

Let $R$ be a uniformly distributed $2 \cdot 160$-bit string and $m$ the message to sign. Then $\mu$ is the compression function derived by [GHR99, Section 6] from the heuristic given in [BP97]:

$$\mu(R, m) = 2^{320} \cdot h(m) + R$$

which is defined only when $\mu(R, m)$ is a prime. Overall:

$$\mu : \{0,1\}^{320} \times \{0,1\}^{512} \rightarrow \{0,1\}^{480}$$

$$\text{enc} \circ \mu : \{0,1\}^{320} \times \{0,1\}^{512} \rightarrow \{0,1\}^k$$

That $\mu(R, m)$ is a prime guarantees division intractability, and the standard hashing lemma [GHR99, Lemma 9] provides a proof of efficiency, through a smooth numbers argument, with the parameter $k = 160$ (which is $k/3$ in our notation). Our Definition 4 is a modified version of their Definition 2, but division intractability holds, and their Lemma 9 applies nonetheless, as we show next.

**Lemma 1.** The function $\mu(R, m) = 2^{320} \cdot h(m) + R$ as defined above is division intractable.

$$\Pr_{\mu \in H_3} \left[ A(\mu) = (X_1, \ldots, X_n, Y) \right.
\left. s.t. Y \neq X_i \text{ for } i = 1, \ldots, n,
\text{ and } \mu(Y) \text{ divides the product } \prod_{i=1}^{n} \mu(X_i) \mod 2^{512} \right]
\leq \frac{2^{32}}{2^{512} - 9} = \frac{1}{2^{341}} = \frac{1}{2^{81} - 9} = \text{negl}(l)$$

The function $\mu$ effectively outputs a random odd prime from the set $f^{-1}(h(X))$ for an input message $X$. As $n$ increases, $\mu$ is distributed more correctly, though Lemma 1 considers the worst case, when $n = 1$, using the theorem on the distribution of prime numbers. The following lemma shows that this can be done efficiently.

**Lemma 2.** [GHR99, Lemma 9] Let $U$ be a universal family from $\{0,1\}^i$ to $\{0,1\}^{i/3}$. Then, for all but a $2^{-i/3}$ fraction of the functions $f \in U$, for all $Y \in \{0,1\}^{i/3}$, a fraction of at least $3/c\cdot l$ of the elements in $f^{-1}(Y)$ are primes, for some small constant $c$. 

5 Improved communication complexity

The signature scheme \((Gen', Sign', Verify')\) described in Section 3 has significant overhead communication complexity: the number of random bits transmitted is proportional to the number of message blocks. This problem represents the main open problem of this paper.

However, we can sketch a first solution to this open problem using a (conjectured) pseudo-random number generator. The following definition is based on [Lub96, p.50-51].

**Definition 6.** Let \(g : \{0,1\}^r \rightarrow \{0,1\}^{(b+1)r} \) be a \(P\)-time function. We say \(g\) is a \((\delta,t)\)-secure pseudo-random number generator for which adversary \(A\) has success probability:

\[
\delta = \left| \Pr_{X \in \{0,1\}^r} [A(g(X)) = 1] - \Pr_{Z \in \{0,1\}^{(b+1)r}} [A(Z) = 1] \right|
\]

if every adversary \(A\) has a probability of success no less than \(\delta\) and a running time of at least \(t\).

The modified scheme would involve the transmission of \(r\) random bits, which would be stretched into \((b+1)r\) random bits via a pseudo-random number generator \(g\), by both the signer and the verifier. The pseudo-random bits take the place of their random counterparts in the original scheme described in Figure 2. The security of this modified scheme is implied by the one of the original scheme, and of the pseudo-random number generator.

For the practical implementation described in the previous section, \(\mu(R_i, m_i) = 2^{220} \cdot h(m_i) + R_i\) where \(R_i\) is the smallest integer greater than the integer defined by the \(i^{th}\) 320-bit block such that \(\mu(R_i, m_i)\) is prime. To quicken the verification process, \(R_i\) can be defined as \(i^{th}\) 320-bit block + \(inc_i\). In such a case, only the value of \(inc_i\) is transmitted with the message block trading time for communication and time of the receiver. This is generally accepted to provide the same distribution, in practice.

6 Conclusion

In [CKN00], the problem of designing a secure general-purpose padding scheme was reduced to the problem of designing a one-block secure padding scheme by providing an efficient and secure tool to extend the latter into the former. By modifying their construction for arbitrary-length messages, and adding one reasonable computational assumption, we provide a practical method of instantiating the secure padding function for short messages using the compression function of dedicated hash functions as well as dedicated encoding functions. We have presented an implementation that uses SHA-1 and PKCS \#1 ver. 1.5. This implementation is independent of the size of the RSA modulus. This was not true in [CKN00].

Dedicated hash functions usually consist of two primitive functions, one of compression and one of chaining. This paper presents an improvement on practicality, since it reduces the potential attacks on the one-block padding scheme to the ones on the hash function’s compression function, eliminating all worries about the chaining function, or its interactions with the compression function.

7 Acknowledgments

We would like to thank David Naccache, Jean-Sébastien Coron, Claude Crépeau and an anonymous referee for useful comments.

References


