An Improved Fast Signature Scheme without on-line Multiplication*

Takeshi Okamoto[1], Mitsuru Tada[2] and Atsuko Miyaji[1]

[1] School of Information Science,
Japan Advanced Institute of Science and Technology (JAIST),
{kenchan, miyaji}@jaist.ac.jp
[2] Institute of Media and Information Technology,
Chiba University,
mt@math.s.chiba-u.ac.jp

Abstract. In this paper, we propose a fast signature scheme which is derived from three-pass identification scheme. Our signature scheme would require a modular exponentiation as preprocessing. However, no multiplication is used in the actual (i.e. on-line) signature generation. This means that the phase involves only a hashing operation, addition and a modular reduction. So far, some fast signature schemes called *on the fly* signatures were proposed. In those schemes the modular reduction is eliminated in the on-line phase. Therefore, our approach to obtain the fast on-line signature is different from theirs. This paper is the first approach for the fast signature scheme without on-line modular multiplication.

Keywords: three-pass identification, digital signature, on-line modular multiplication, random oracle model and provable security

1 Introduction

Nowadays, a signature scheme is an important tool for secure communication. Consequently, we are longing for more compact signature schemes to spread public-key infrastructure (PKI) system. In this case, the compactness means the efficiency of both computational work and transmitted data size. Such a compactness gives users' convenience, and is acceptable for various application to capacity limited devices such as a smart card.

To consider the computational efficiency in signature schemes, let us focus on the signer's computational work in *generic digital signature schemes*[1]. In such a signature scheme, there exist two kinds of computation for the signer: it consists

* This work has been supported by the Telecommunications Advancement Organization of Japan under the grant for international joint research related to information-communications.

[1] As well as in [17], in this paper, *a generic (digital) signature scheme* means a signature scheme which can be derived from a three-pass identification scheme by using an appropriate hash function.

of *pre-computation* and *(actual) signature generation*. The pre-computation can compute values without a message to be signed, and execute during idle time and completely independent of message to be signed. This means that such a computational cost does not influence the real-time computing. We say that such a computation is the *off-line* processing. On the other hand, the (actual) signature generation does directly influence the processing time because a message is indispensable to compute. We say that such a computation is *on-line* processing.

To estimate the efficiency of a signature scheme, we should separately consider the two types of computation. Needless to say, the *fast on-line signature*[2] can make digital signatures much more practical in a variety of scenarios.

To realize the fast on-line signature generation, Girault [6] modified Schnorr's signature scheme [22] in which an RSA-modulus[3] is used instead of a prime modulus. This modification leads to no modulo reduction in the on-line signature generation. Therefore, in Girault's scheme, faster processing of the signature generation is possible than in Schnorr's one. In 1998, Poupard and Stern [19] investigated and gave provable security for Girault's scheme, and named that scheme GPS-scheme. Due to the description of [19], in this paper, we call a generic signature scheme in which modulo reduction is not necessary in the on-line signature generation, *on the fly* signature scheme.

In 1999, Poupard and Stern [20] proposed another on the fly signature scheme (PS-scheme for short), whose security relies on the difficulty of integer factoring. In 2001, Okamoto, Tada and Miyaji [15] proposed on th fly signature (OTM-scheme for short) by improving PS-scheme in the computational work and transmitted data size.

In this paper, we propose a fast signature scheme, which is derived from a three pass identification scheme. Our idea to reduce the on-line computation, completely differ from that of on the fly signatures. That is, the on-line multiplication is eliminated in our scheme, whereas the modular reduction is done in on the fly signatures. Up to the present, some fast signature schemes which have the property of on the fly signature, have been proposed. However, this paper is the first approach for the fast signature scheme without on-line modular multiplication.

To sum up, in the on-line computation, the modular reduction is used in our schemes, and multiplication is used in on the fly schemes. Now let us compare the computational efficiency between the multiplication and the modular reduction. In the multiplication, a recursive algorithm due to [8] reduces the complexity of the multiplying. On the other hand, in the modular reduction, we can use the efficient technique such as [1, 11].

Those methods are further advantageous than [8] because a *single modulus* can be used in our schemes and many reductions are performed by using such a modulus. That is, in our scheme the modulus s is fixed because s is a secret key.

[2] In this paper, a *fast on-line signature* means the signature scheme which has the on-line computational efficiency.

[3] In this paper, we call a modulus to be a product of two distinct primes an *RSA-modulus*.

This property lead to a good computational efficiency for modular reduction. On the other hand, such an efficiency does not exist for the multiplication.

Consequently, the on-line modular reduction in our schemes is faster than the on-line multiplication in on the fly signatures from implementation point of view. For more detailed treatment of the above methods, see [9].

As for the security, our scheme is based on the integer factoring problem for RSA modulus n and provably secure in the random oracle model. To satisfy the security, our scheme uses a public key g with specific structure, called *asymmetric basis*, and which is a variant of [16]. This property leads to the good efficiency in terms of both size of data and amount of work.

Concrete to say, compared with OTM-scheme, the size of a signature in our signature can be reduced by at least 21%, and computational cost in our scheme for verification can be reduced by 32%, respectively. In the same way, compared with PS-scheme, the size of a secret key and a signature can be reduced by at least 68% and 58%, respectively. Furthermore, the computational cost for pre-computation and verification can be reduced by at least 83% and 78%, respectively.

This paper is organized as follows. In Section 2, we will review on the fly schemes. In Section 3, we will introduce our proposed identification scheme, and will give provable security for ours. In Section 4, we will introduce our signature scheme derived from our identification, and will discuss the security consideration. In Section 5, we will suggest practical values of our identification and the resulting signature scheme and will describe the implementation notes. In Section 6, we will introduce an efficient signature scheme which is optimized in terms of data size, and will evaluate the performance of our signature scheme by comparing with existing on the fly signatures: OTM-scheme, PS-scheme and GPS-scheme. The conclusion will be given in Section 7.

2 Previous Work

In this section, we briefly survey the previous works.

We first introduce some notations. The symbol $\varphi(\cdot)$ denotes Euler totient function, that is, $\varphi(n)$ is the number of the natural numbers less than n and coprime to n. The symbol $\lambda(\cdot)$ denotes so-called Carmichael function, that is, $\lambda(n)$ is the greatest number among the possible orders of elements in $\mathbb{Z}_n^*$. The order of an element $g \in \mathbb{Z}_n^*$ is represented as $\mathrm{Ord}_n(g)$. We denote by $|x|$, the binary length for a positive integer x. We say that p is a strong prime if $p = 2p'+1$ and both p and p' are primes.

Now let us recall the on the fly schemes (identification version).

GPS-scheme [19]: The public key is (n, g, v), where n is an RSA modulus, $g \in \mathbb{Z}_n^*$ is an element with high order, and $v = g^{-s} \bmod n$, where $s \in \mathbb{Z}_{2^k}$ is picked up at random. The secret key is s.

In identification step, the commitment is $x = g^r \bmod n$, where $r \in \mathbb{Z}_A$ is picked up at random, the random challenge is $e \in_R \mathbb{Z}_B$, and the answer is

$y = r + se$ (in $\mathbb{Z}$). A verifier finally checks whether $x = g^y v^e \bmod n$ holds or not. Here each parameter (A, B, k) satisfies $B \ll 2^k \ll A$.

It is known that if one sets up the parameters in GPS-scheme such that the scheme is as secure as the discrete logarithm problem for modulus n under the one key attack scenario [20], the scheme leads to the inefficiency for both the amount of work and transmitted data size. For more details, we refer to [16].

PS-scheme [20]: The public key is (n, g), where n is a product of strong primes p and q, and $g \in \mathbb{Z}_n^*$ is an element such that $\mathrm{Ord}_n(g) \in \{\lambda(n), \lambda(n)/2\}$. The secret key is $s = n - \varphi(n)$ $(= p + q + 1)$, where $|s| = k$.

In identification step, the commitment is $x = g^r \bmod n$, where $r \in \mathbb{Z}_A$ is picked up at random, the random challenge is $e \in_R \mathbb{Z}_B$, and the answer is $y = r + se$ (in $\mathbb{Z}$). A verifier finally checks whether both $y < A$ and $x = g^{y-ne} \bmod n$ hold or not. Here each parameter (A, B, k) satisfies $B \ll 2^k \ll A$.

PS-scheme has some drawbacks: As you see, a secret key in PS-scheme is $s = n - \varphi(n)$ and the size of s is very large, i.e. about $|n|/2$. This feature lead to inefficiency in view of the amount of work and transmitted data size.

The following OTM-scheme solves the above drawback.

OTM-scheme [15]: The public key is (n, g, z), where $n = \prod_{i=1}^{t} p_i$ for an integer $t \geq 2$, $g \in \mathbb{Z}_n^*$ is an element such that $q = \mathrm{Ord}_n(g)$ and $z \in \mathbb{Z}_{2^c}$ is picked up at random. The secret key is $s = z \bmod q$, where $|s| = k$.

In identification step, the commitment is $x = g^r \bmod n$, where $r \in \mathbb{Z}_{2^a}$ is picked up at random, the random challenge is $e \in_R \mathbb{Z}_{2^b}$, and the answer is $y = r + se$ (in $\mathbb{Z}$). A verifier finally checks whether both $y < 2^{a+1}$ and $x = g^{y-ze} \bmod n$ hold or not. Here each parameter (a, b, c, k) satisfies $2^b \ll 2^k \ll 2^a \ll 2^{bc}$.

The discussion for the above on the fly signatures and our proposed scheme, is given in Section 6.2.

3 Identification Scheme

In this section, we introduce our identification scheme. The security consideration is also discussed.

3.1 Protocol

Our identification scheme uses the parameters k, s, a and b with $2^{k-1} \leq s < 2^k \ll 2^a \ll 2^b$, and also uses slightly generalized asymmetric basis [16], which is defined as follows.

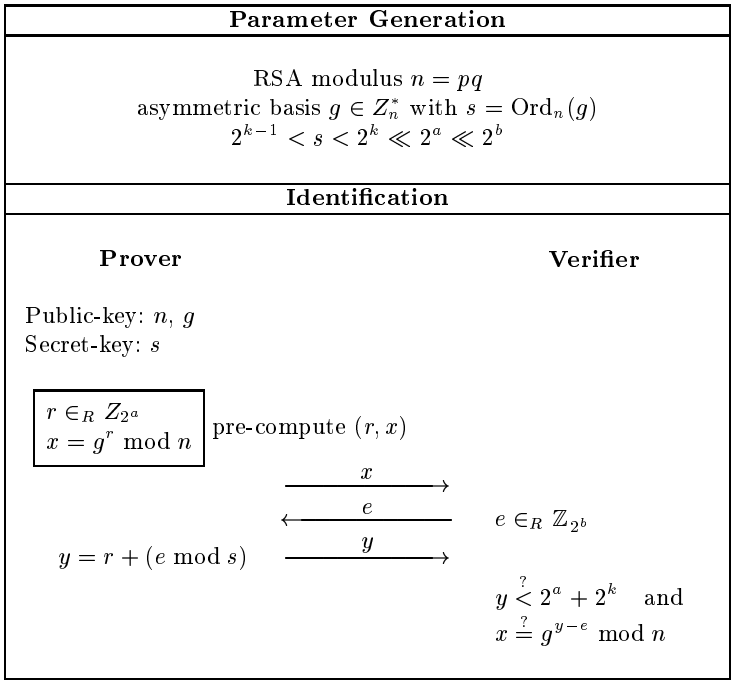

Fig. 1. Proposed identification scheme

Definition 1 (Asymmetric basis) Let n be an RSA modulus such that $n = pq$. Then we say that g is an asymmetric basis in $\mathbb{Z}_n^*$ if the multiplicity of 2 in $\mathrm{Ord}_p(g)$ is not equal to that of 2 in $\mathrm{Ord}_q(g)$. □

We now give our identification protocol.

Key generation step: The following steps are executed:

Step1 Pick up two same-size primes p and q, and compute $n = pq$.

Step2 Choose an element $g \in \mathbb{Z}_n^*$ which is an asymmetric basis in Z_n^*, where $s = \mathrm{Ord}_n(g)$.

Public-key/Secret-key: The public-key is (n, g) and the corresponding secret-key is s.

Identification step: A prover and a verifier execute the following steps:

Step1 The prover picks up a random number $r \in \mathbb{Z}_{2^a}$, computes the commitment $x = g^r \bmod n$ and sends x to the verifier.

Step2 The verifier picks up a random challenge $e \in \mathbb{Z}_{2^b}$ and sends e to the prover.

Step3 The prover computes $z = e \bmod s$ and an answer $y = r + z$ in $\mathbb{Z}$, and sends y to the verifier.

Step4 The verifier checks whether both $y < 2^a + 2^k$ and $x = g^{y-e} \bmod n$ hold or not. If both equations hold, the verifier accepts. Otherwise she rejects.

In Step3, the on-line multiplication for the prover is eliminated. This is the main idea of our scheme.

Note that, in the conventional identification schemes such as Schnorr [22] or Guillou-Quisquater [7], the challenge e can be a fixed constant. In this case, such schemes whould have some rounds. However, in our identification, e has the condition such that $2^k \ll e$. Therefore, the round of our identification is constant and fixed as one. We can say such a property indeed characterizes our schemes.

3.2 Security analysis

We show that our proposed scheme is a three-pass honest-verifier statically zero-knowledge identification protocol. As a strategy, we would like to indicate that our scheme provides completeness, soundness and the statistical zero-knowledge property, respectively. To estimate the rigorous security, we follow the approach of Feige, Fiat and Shamir in [5].

We say that a positive function $f(k) : \mathbb{N} \to \mathbb{R}$ is said to be *negligible*, if for any c, there exists a k_c such that $f(k) \leq k^{-c}$ for any $k \geq k_c$. Otherwise f is said to be *non-negligible*. $\|x\|$ denotes the absolute value of x.

Lemma 2 Let n be an RSA modulus and g be an asymmetric basis in $\mathbb{Z}_n^*$. Assume that we find $L > 0$ such that $g^L = 1 \bmod n$. Then we can construct a Turing machine M which on input n, g and L outputs a factor of n in time $O(|L||n|^2)$

Proof. This lemma is basically due to [16]. Hereafter, we describe how to construct M.

At first, M extract the odd part b of L, such that $L = 2^a b$. Since g is an asymmetric basis in $\mathbb{Z}_n^*$, it holds $g^{2b} = 1 \bmod p$ and $g^{2b} = 1 \bmod q$, and also holds $g^b = 1 \bmod p$ and $g^b = -1 \bmod q$. Then we have the following results: $p \mid g^b - 1$ and $n \nmid g^b - 1$. Consequently, M can find a factor of n by computing $\gcd(g^b - 1 \bmod n, n)$.

Note that the modular exponentiation algorithm and the extended Euclidean algorithm have a running time of $O(|L||n|^2)$ and $O(|n|^2)$, respectively. Hence M can execute the above steps in time $O(|L||n|^2)$. □

Theorem 3 (Completeness) In the proposed identification scheme, the prover who has a secret-key, and who follows the scheme, is always successfully accepted.

Proof. The answer y correctly computed by the prover, is $r + z$, where $z = e \bmod s$. Since $r < 2^a$ and $z < s < 2^k$, those conditions satisfy $y < 2^a + 2^k$. Furthermore, from the following equations:

$$g^{y-e} = g^{(r+z)-e} = g^{r+(e \bmod s)-e} = g^r = x \bmod n,$$

a correctly generated answer y always passes the verification. □

Theorem 4 (Soundness) If there exists a polynomial-time adversary $\tilde{P}$ which is accepted by honest verifiers with probability $> 1/2^b + \varepsilon, \varepsilon > 0$, where the average running time is T. Then, by using $\tilde{P}$, we can construct a polynomial-time machine M, which can figure out the factorization of public-key n, in expected time $O(T/\varepsilon + |n|^{O(1)})$.

Proof. Hereafter, we illustrate how to construct M. First M input (n, g) to $\tilde{P}$. Here we denote by ST, the current $\tilde{P}$'s state at this stage. *Ver* is a verification function, on input (x, e, y), outputs OK if those parameters are valid. Otherwise it outputs NG.

M executes the following algorithm and tries to obtain forged parameters.

% Algorithm 1
input result := NG, counter := 0
while (result = NG $\wedge$ counter $< |n|/\varepsilon$) do
pick up a random tape ω at random
$x := \tilde{P}(\omega)$
pick up a random number $e \in \mathbb{Z}_{2^b}$ at random
$y := \tilde{P}(e)$
result := $Ver(x, e, y)$
counter := counter + 1
end while
output (ω, x, e, y)
end

If the forgery of Algorithm 1 is successful, M executes the next steps. Otherwise it stops (this time the attempt is failure). The probability with which M can obtain the forged parameters is estimated as:

$$1 - (1 - \varepsilon)^{\frac{\lfloor n \rfloor}{\varepsilon}} \geq 1 - \left(\frac{1}{\tilde{e}}\right)^{|n|},$$

where $\tilde{e}$ is the natural logarithm.

Next, M initializes the $\tilde{P}$'s condition: set as ST. By using the same (ω, x) obtained by Algorithm 1, M tries to obtain forged parameters (x, e', y'). In this case, we denote by ε' the probability to obtain those parameters by one iteration.

M executes the following algorithm.

% Algorithm 2
input ω, result := NG, counter := 0
$x := \mathcal{A}(\omega)$
while (result = NG $\wedge$ counter $< |n|/\varepsilon'$) do
pick up a random number $e' \in \mathbb{Z}_{2^b}$ at random
$y' := \tilde{P}(e')$
result := $Ver(x, e', y')$
counter := counter + 1
end while
output (e', y')
end

It can be formally proved that $\varepsilon' \geq \varepsilon/2$. For more details, see *Splitting Lemma* in [17]. In the same way as before, the probability with which M can obtain the forged parameters is $\geq 1 - \left(\frac{1}{e}\right)^{|n|}$.

As a consequence, the running time by executing Algorithm 1 and Algorithm 2 is $O(|n|t/\varepsilon)$ and the probability with which M can obtain (x, e, y) and (x, e', y') is estimated as $\geq \frac{1}{2}\left(1 - \left(\frac{1}{e}\right)^{|n|}\right)^2$.

Finally, M computes $L = \|(y - y') - (e - e')\|$. Here $L > 1$ is a multiple of $\mathrm{Ord}_n(g)$, that is, $g^L = 1 \bmod n$, and g is an asymmetric basis. Therefore, as in the consequence of Theorem 2, M can obtain the factorization of public in expected time $O(kT/\varepsilon + |n|^{O(1)})$. □

Theorem 5 (Zero-knowledge property) The proposed identification scheme provides honest-verifier statistical zero-knowledge property, if $2s/2^a$ is negligible.

Proof. We prove the above theorem along the line of Theorem 6 in [19]. To prove the theorem, we show that we can construct a polynomial-time machine (simulator) M which simulates the communication between the honest prover and a honest verifier.

In this case, M executes the following steps:

Step1 Pick up two random numbers: $e' \in \mathbb{Z}_{2^b}$ and $y' \in \mathbb{Z}_{2^a}$.
Step2 Compute $x' = g^{y'-e'} \bmod n$.
Step3 Output (x', e', y').

We denote by π, the probability in the communication between a honest prover and a honest verifier, that is, $\pi = \mathtt{Pr}\left[(x, e, y) = (\alpha, \beta, \gamma)\right]$. Then we have the following:

$$
\begin{aligned}
\pi &= \mathtt{Pr}\begin{bmatrix} \alpha = g^r \bmod n, \\ \beta = e, \\ \gamma = r + \Omega \end{bmatrix} \\
&= \frac{1}{2^{a+b}} \cdot \sum_{\substack{0 \leq r < 2^a \\ 0 \leq e < 2^b}} \mathtt{Pr}\begin{bmatrix} \alpha = g^{\gamma - \Omega'} \bmod n, \\ \beta = e, \\ r = \gamma - \Omega' \end{bmatrix} \\
&= \frac{1}{2^{a+b}} \cdot \mathtt{Pr}\begin{bmatrix} \alpha = g^{\gamma-\beta} \bmod n, \\ 0 \leq \beta < 2^b, \\ 0 \leq \gamma - \Omega' < 2^a \end{bmatrix} \\
&= \frac{1}{2^{a+b}} \cdot \chi(\alpha = g^{\gamma-\beta} \bmod n) \\
&\quad \cdot \chi(0 \leq \beta < 2^b) \cdot \chi(\Omega' \leq \gamma < 2^a + \Omega'),
\end{aligned}
$$

where $(e \bmod q)$ and $(\beta \bmod q)$ are denoted by Ω and Ω', respectively. For a predicate Q, $\chi(Q)$ means the characteristic function of Q, that is, if Q is true, then $\chi(Q) = 1$, otherwise $\chi(Q) = 0$.

In the same way as above, the probability by M, that is, $\mathtt{Pr}\left[(x', e', y') = (\alpha, \beta, \gamma)\right]$ is denoted by π'. In this case, we have the following:

$$\begin{aligned}
\pi' &= \mathtt{Pr}\left[\begin{array}{c}\alpha = g^{y'-e'} \bmod n,\\ \beta = e',\\ \gamma = y'\end{array}\right] \\
&= \frac{1}{2^{a+b}} \cdot \sum_{\substack{0 \le r < 2^a \\ 0 \le e < 2^b}} \mathtt{Pr}\left[\begin{array}{c}\alpha = g^{\gamma-\beta} \bmod n,\\ \beta = e',\\ \gamma = y'\end{array}\right] \\
&= \frac{1}{2^{a+b}} \cdot \chi(\alpha = g^{\gamma-\beta} \bmod n) \\
&\qquad \cdot \chi(0 \le \beta < 2^b) \cdot \chi(0 \le \gamma < 2^a),
\end{aligned}$$

We whould like to estimate the distance between actual system and M. We define $\Delta = \sum_{\alpha,\ \beta,\ \gamma} \|\pi - \pi'\|$. From the results of π and π', we obtain

$$\Delta = \sum_{\substack{\alpha = g^{\gamma-\beta} \bmod n \\ 0 < \beta < 2^b \\ 0 \le \gamma < \Omega'}} \pi' + \sum_{\substack{\alpha \\ \beta \\ \Omega' \le \gamma < 2^a}} \|\pi - \pi'\| + \sum_{\substack{\alpha \\ \beta \\ 2^a \le \gamma < 2^a + \Omega'}} \pi.$$

Since

$$\left(\sum_{\substack{\alpha,\ \beta \\ 0 \le \gamma < \Omega'}} \pi'\right), \left(\sum_{\substack{\alpha,\ \beta \\ 2^a \le \gamma < 2^a + \Omega'}} \pi\right) \le \frac{\Omega'}{2^a} < \frac{q}{2^a}$$

and

$$\sum_{\substack{\alpha,\ \beta \\ \Omega' \le \gamma < 2^a}} \|\pi - \pi'\| = 0,$$

we can conclude $\Delta < 2q/2^a$. This proves the theorem. □

4 Signature Scheme

We introduce our signature scheme which is derived from the identification scheme proposed in Section 3.

4.1 Protocol

As well as conventional identification schemes such as Schnorr [22] or Guillou-Quisquater [7], our identification scheme in Section 3 can be turned into a signature scheme by using the technique in [4]. Then the challenge in the identification is replaced with an appropriate hash function. Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^b$ be a hash function.

Key generation step: This is the same as our identification scheme.

Public-key/Secret-key: This is the same as our identification scheme.

Signature generation step: Suppose a signer which has a public-key and the corresponding secret-key, generates the signature of her message $m \in \{0,1\}^*$. Then she executes the following steps:

Step1 Pick up a random number $r \in \mathbb{Z}_{2^a}$ and compute $x = g^r \bmod n$.

Step2 Compute $e = \mathcal{H}(x, m)$.

Step3 Compute $z = e \bmod s$ and $y = r + z$ in $\mathbb{Z}$.

Signature: The signature for a message m is (x, e, y).

Verification step: Suppose a verifier which has the signer's public-key and the corresponding message, checks the validity of the signature for m. Then she executes the following steps:

Step1 Check whether $y < 2^a + 2^k$ holds or not. If the equation does not hold, then reject the signature and stop this protocol.

Step2 Compute $e' = \mathcal{H}(x, m)$ and $x' = g^{y-e} \bmod n$.

Step3 Check whether both $e = e'$ and $x = x'$ hold or not. If both equations hold, accept the signature. Otherwise reject it.

4.2 Security analysis

In Section 3.2, we have proved that our identification scheme is three-pass honest-verifier statistically zero-knowledge identification protocol. This result includes all the properties to apply the technique of *the forking lemma* in [17]. Therefore, we can say that, in the random oracle model [2, 17], our signature scheme is existentially unforgeable under the adaptive chosen message attack

Theorem 6 Let Q be the number of queries which a polynomial-time adversary $\mathcal{A}$ executing the adaptive chosen-message attack, can ask to the random oracle, and let R be the number of queries which $\mathcal{A}$ can ask to the actual signer. Assume that $2s/2^a$ is negligible. Also assume that, $\mathcal{A}$ can forge a signature with non-negligible probability $\varepsilon \geq 10(R+1)(R+q)/2^b$, where the average running time of $\mathcal{A}$ is T. Then we can construct a polynomial-time machine M which can factor n with non-negligible probability in expected time $O(QT/\varepsilon + |n|^{O(1)})$.

Proof. By the result of Theorem 5, the signature oracle in our signature scheme can be statistically simulated by a probabilistic polynomial-time machine M which works according to the protocol like [17].

When M uses $\mathcal{A}$, she can obtain two distinct signatures (x, e, y) and (x', e', y') such that $x = x'$, but $e \neq e'$. Then, we can get a multiple of $\mathrm{Ord}_n(g)$ such that $L > 1$ and $g^L = 1 \bmod n$. Here g is an asymmetric basis in $\mathbb{Z}_n^*$, therefore by the result of Lemma 2 we can find a factor of n in expected time $O(QT/\varepsilon + |n|^{O(1)})$. □

5 Parameter Generation

In this section, we describe the conditions of the parameters to keep the security in our identification and the corresponding signature scheme. We also show that how to implement the parameters.

5.1 Choice of parameters

Parameters a and b: For the security reason, the values of a and b shall satisfy: $a = k + \kappa_1$ and $b = a + \kappa_2$, where k is a security parameter satisfying $|s| = k$ with $\mathrm{Ord}_n(g) = s$, and where both κ_1 and κ_2 are information leak parameters.

Parameters κ_1 and κ_2: By Theorem 5, we must set such that $1/2^{\kappa_1}$ is intractable. As for κ_2, let us first consider the following problem: given (n, g, α, κ_2), find β such that $g^\alpha = g^\beta \bmod n$, where $|\beta|$ is at least κ_2-bits smaller than $|\alpha|$. In our schemes, we must set κ_2 on the condition that the problem is hard to solve for the security parameter k. For implementation, we should take κ_1 and κ_2 greater than 64 and 24 bits, respectively.

Parameter s: Let us consider the attack which an adversary computes s only from the information of the public-key (n, g). We can see the algorithms to extract s, such as *Pollard lambda method* in [18] and *the baby-step giant-step method* in [10]. One may say that the former is better than the latter since it has same computational complexity (exponential-time: $O(\sqrt{s})$) but does not need large memory. The size of s shall be set up such that the above algorithms cannot apply for the security parameter k. For implementation, we should take $|s| = k$ greater than 160 bits for the security reason.

5.2 Implementation notes

We first describe how to find p, q and an asymmetric basis g in $\mathbb{Z}_n^*$.

Step1 Pick up two primes $p = 2p'p'' + 1$ and $q = 2q'q'' + 1$ such that p' and q' are also primes, and p'' and q'' are odd numbers.

Step2 Choose $\alpha_p \in \mathbb{Z}_p^*$ satisfying $g_p = \alpha_p^{(p-1)/p'} \neq 1 \bmod p$. In the same way, choose $\alpha_q \in \mathbb{Z}_q^*$ satisfying $\alpha_q \neq q - 1 \bmod q$, $\alpha_q^{(q-1)/2} \neq 1 \bmod q$ and $g_q = \alpha_q^{(q-1)/2q'} \neq 1 \bmod q$.

Step3 Compute $n = pq$ and $g = q(q^{-1} \bmod p)g_p + p(p^{-1} \bmod q)g_q \bmod n$.

In Step3, g is computed by using the technique of Chinese Reminder Theorem (CRT). Note that $\mathrm{Ord}_p(g) = p'$ and $\mathrm{Ord}_q(g) = 2q'$. Therefore $\mathrm{Ord}_n(g) = \mathrm{lcm}(p', 2q') = 2p'q'$.

Next, we discuss secure hash algorithm which we should adopt. If $\mathcal{H}$ were an *ideal* hash function, then the proposed signature scheme would be *secure* under the meaning of the description in Section 4.2. Since such a random function does

not exist in the real world, in implementation, we are recommended to adopt MD5 by [21] or SHA-1 by [13], each of which is designed so that the algorithm can be a collision intractable hash function [3].

6 Efficiency for Signature Schemes

In this section, we first introduce the optimized signature scheme. We next evaluate the performance.

6.1 Optimized scheme of data size

With respect to the signature scheme in Section 4, we can diminish the size of the signature. Consequently, communication lord is more efficient than before. We now focus on the following two parts:

Elimination of x: When we have two parameters e and y, the parameter x can be generated by computing $x = g^{y-e} \bmod n$. Therefore, the signature x is eliminated like conventional generic signature schemes such as Schnorr [22] or Guillou-Quisquater [7]. In this case, the signature for m consists of (e, y).

Using a short-size e: As for the signature scheme in Section 4, signature e is, in a certain sense, verbose. In our signature scheme, large-size e such as $2^a \ll e$ is actually needed in verification. Hence, we can use the following technique: we regard short-size e satisfying $2^k \ll e \ll 2^b$ as signature, and in verification, we extend the size of e by using appropriate hash function.

We use two hash functions $\mathcal{F} : \{0,1\}^* \to \{0,1\}^c$ and $\mathcal{G} : \{0,1\}^c \to \{0,1\}^b$, where c has the condition $2^c \ll 2^b$. The optimized signature scheme, which apply the above techniques, are as follows:

Key generation step: This is the same as our identification scheme.

Public-key/Secret-key This is the same as our identification scheme.

Signature generation step: A singer executes the following steps:
Step1 Pick up a random number $r \in \mathbb{Z}_{2^a}$ and compute $x = g^r \bmod n$.
Step2 Compute $e = \mathcal{F}(x, m)$.
Step3 Compute $\epsilon = \mathcal{G}(e)$, $z = \epsilon \bmod s$ and $y = r + z$ in $\mathbb{Z}$.

Signature: The signature for a message m is (e, y).

Verification step: A verifier executes the following steps:
Step1 Check whether $y < 2^a + 2^k$ holds or not. If the equation does not hold, then reject the signature and stop this protocol.
Step2 Compute $\epsilon' = \mathcal{G}(e)$ and $e' = \mathcal{F}(g^{y-\epsilon'} \bmod n, m)$.
Step4 Check whether $e' = e$ holds or not. If the equation holds, accept the signature. Otherwise reject it.

As for the hash function $\mathcal{G}$, we should take c greater than 160 bits for implementation.

Table 1. Performance of signature schemes

Scheme	UMP	CPC ($\times M$)	CSG	CVF ($\times M$)	SPK (bits)	SSK (bits)	SSig (bits)
Our scheme	Integer factoring with g	61	248 mod 160	372	2048	160	304
OTM-scheme	Integer factoring with g	61	80×160	552	2048	160	384
PS-scheme	Integer factoring	381	80×512	1656	1024	513	736
GPS-scheme	Discrete log. for modulo n	381	80×1024	1796	3072	1024	1264

Abbreviation:

- UMP means the underlying mathematical problem that the signature scheme relies on for its security.
- CPC, CSG and CVF mean the computational cost for pre-computation, signature generation and verification, respectively
- SPK, SSK and SSig means the size of a public-key, a secret-key and a signature, respectively.
- M represents the computational cost for one multiplication under a 1024-bit modulus.
- α mod β represents the computational cost for modular reduction of an α-bit number and a β-bit number modulus.
- $\gamma \times \delta$ represents the computational cost for multiplication of an γ-bit number and a δ-bit number on $\mathbb{Z}$.

Notes:

- For all schemes in the table, we set up the parameter under the line of the one-key attack scenario [20].
- For respective computational cost, a primitive arithmetic of binary methods [9] are used, e.g. amount of work for $g^{\alpha} \bmod n$ is $\frac{3}{2}|\alpha|M$ if $|n| = 1024$. Of course there exist more sophisticated techniques which reduce the amount of computational work. However we think they estimate the concrete performance without loss of generosity.
- In UMP, integer factoring with g means that it is a variant of integer factoring problem on input RSA modulus n and the asymmetric basis g, outputs the factor of n.
- In CPC, the signer uses the technique of CRT. In this case, the signer must secretly have the factors of n, p and q.

6.2 Performance

We evaluate the efficiency of our signature scheme by comparing existing on the fly signatures. Table 1 gives the performance of various schemes, such as OTM-scheme, PS-scheme and GPS-scheme, including ours.

The parameters in the schemes are set up as follows.

- Our scheme has $a = 224$ and $b = 248$ by taking $k = 160$, $\kappa_1 = 64$ and $\kappa_2 = 24$. Our scheme also use the technique of Section 6.1.
- OTM-scheme has $a = 104$, $b = 80$ and $c = 288$ by taking $k = 160$. To keep the same security as our scheme, in OTM scheme, n is a RSA modulus and g is an asymmetric basis in $\mathbb{Z}_n^*$. Furthermore, the size of public-key is optimized as follows. We regard actual public-key as (n, g), and z is computed by $z = \mathcal{H}'(n, g)$, where $\mathcal{H}'$ is a hash function $\mathcal{H}' : \{0,1\}^* \to \{0,1\}^c$.
- PS-scheme has $|A| = 656$ and $|B| = 80$ by taking $k = 513$.
- GPS-scheme has $|A| = 1184$ and $|B| = 80$ by taking $k = 1024$.

Additionally, we set $|n| = 1024$ for all schemes.

Next, we show the comparison between OTM-scheme (resp. PS-scheme and resp. GPS-scheme) and our signature.

OTM-scheme: One of the verifications in OTM-scheme is $x = g^{y-ze} \bmod n$, where in the index of g, we can see the multiplication of two parameters z and e. On the contrary, the verification in our scheme is $x = g^{y-e} \bmod n$, hence the multiplication in the index does not exist. The large-size index involved by the multiplication, lead to the inefficiency from both the amount of work and data size point of view. Nowadays, all the existing on the fly signatures have the same drawbacks.

Consequently, `CVF` and `SSig` in our scheme is superior to those in OTM-scheme. Since the public key in our scheme and that in OTM-scheme are (n, g) and (n, g, z), respectively, the number of the parameters in our scheme is smaller than that in OTM-scheme.

PS-scheme: The size of secret key in PS-scheme is only dependent on the modulus n, and that is considerably large (about $|n|/2$). This drawback leads to inefficient results with respect to the computational work (`CPC`, `CSG` and `CVF`) and the data size (`SSK` and `SSig`).

On the other hand, since PS-scheme is intended to be used with a modulus product of two strong primes, $g = 2$ is a correct basis and do not have to be included in the public key. Consequently, we can set $\texttt{SPC} = 1024$ for PS-scheme. Therefore, one may say that PS-scheme is more efficient than our scheme in terms of size of public key.

GPS-scheme: Since the public key in GPS-scheme consists of three parameters such as (n, g, v), the size of the public key is $\texttt{SPK} = 3072$. Hence, GPS-scheme has the largest size for public-key of all the schemes in Table 1.

Note that, in the table, all the schemes are based on the one-key attack scenario. Consequently, GPS-scheme has provable security such that the scheme is as secure as the discrete problem for modulo n. However, the size of secret-key in GPS-scheme is considerably large: $\mathtt{SSK} = 1024$. In the same reason as in PS-scheme, this result leads to the inefficiency mentioned above.

Table 1 show that our signature scheme is quite efficient from both the computational cost and the data size point of view.

We now give the concrete evaluation measured by comparing the integer factoring based scheme, OTM-scheme (resp. PS-scheme) and our scheme. Compared with OTM-scheme, our scheme enables the computational cost to be reduced by 32% for verification. For the data size, the signature size is reduced by 21%. In the following, compared with PS-scheme, our scheme enables the computational cost to be reduced by 83% for pre-computation and by 78% for verification. For the data size, the secret-key size is reduced by 68% and the signature size is 58%.

7 Conclusion

In this paper, we have proposed an efficient and fast signature scheme, which is derived from a three-pass identification scheme. Our proposed signature is, in a sense, a counterpart for on the fly signature schemes.

Compared between two types of signatures, the on-line computation in our scheme is smaller than that in on the fly schemes, because modular reduction is faster than multiplication from implementation point of view.

We have shown that our signature schemes are existentially unforgeable against any polynomial-time adversaries that can execute adaptive chosen message attack in the random oracle model. In this case, the underlying number theoretic problem is the integer factoring problem for an RSA modulus n.

We also have shown that our scheme is more efficient than on the fly signature schemes, in view of both the computational cost and the transmitted data size.

References

1. P. Barrett: "*Implementing of the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor*", Advances in cryptology - Crypto'86, Lecture Notes in Computer Science 263, Springer-Verlag, pp.311-323, 1987.
2. M. Bellare and P. Rogaway: "*Random oracles are practical: a paradigm for designing efficient protocols*", Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS), 1993.
3. I. Damgård: "*Collision free hash functions and public key signature schemes*", Advances in cryptology - Eurocrypt'87, Lecture Notes in Computer Science 304, Springer-Verlag, pp.203-216, 1988.
4. A. Fiat and A. Shamir: "*How to Prove Yourself: practical solutions of identification and signature problems*", Advances in cryptology - Crypto'86, Lecture Notes in Computer Science 263, Springer-Verlag, pp.186-194, 1987.

5. U. Feige, A. Fiat and A. Shamir: "*Zero-knowledge proofs of identity*", Journal of cryptology, vol.1, pp.77-95, 1988.
6. M. Girault: "*Self-certified public keys*", Advances in cryptology - Eurocrypt'91, Lecture Notes in Computer Science 547, Springer-Verlag, pp.490-497, 1992.
7. L. C. Guillou and J. J. Quisquater: "*A 'paradoxal' identity-based signature scheme resulting from zero-knowledge*", Advances in cryptology - Crypto'88, Lecture Notes in Computer Science 403, Springer-Verlag, pp.216-231, 1989.
8. A. Karatsuba and Yu. Ofman: "*Multiplication of multidigit numbers on automata*", Soviet Physics - Koklady, vol.7, pp.595-596, 1963.
9. D. E. Knuth: "*Seminumerical Algorithms*", The art of computer programming, vol.2, Second edition, Addison-Wesley, 1981.
10. D. E. Knuth: "*Sorting and Searching*", The art of computer programming, vol.3, Second edition, Addison-Wesley, 1998.
11. P. Montgomery : "*Modular multiplication without trial division*", Mathematics of computation, vol.44, pp.519-521, 1985.
12. D. Naccache, D. M'raihi, S. Vaudenay and D. Raphaeli : "*Can DSA be improved ?*", Advances in cryptology - Eurocrypt'94, Lecture Notes in Computer Science 950, 1995.
13. National Institute of Standards and Technology (NIST): "*Secure hash standards (SHS)*", Federal Information Processing Standards, 1995.
14. K. Ohta and T. Okamoto: "*On Concrete Security Treatment of Signatures Derived from Identification*", Advances in cryptology - Crypto '98, Lecture Notes in Computer Science 1462, pp.354–369, 1998.
15. T. Okamoto, M. Tada and A. Miyaji "*Proposal of Efficient Signature Schemes based on Factoring*", Trans. IPSJ, Vol.42 No. 8, pp.2123–2133, 2001.
16. D. Pointcheval: "*The Composite Discrete Logarithm and Secure Authentication*", Advances in cryptology - PKC'00, Lecture Notes in Computer Science 1751, 2000.
17. D. Pointcheval and J. Stern: "*Security arguments for digital signatures and blind signatures*", Journal of cryptology, vol.13, no.3, Springer-Verlag, pp.361-396, 2000.
18. J. Pollard: "*Monte Carlo methods for index computation mod p*", Mathematics of Computation, vol 32, pp.918-924, 1978.
19. G. Poupard and J. Stern: "*Security analysis of a practical 'on the fly' authentication and signature generation*", Advances in cryptology - Eurocrypt'98, Lecture Notes in Computer Science 1403, Springer-Verlag, pp.422-436, 1998.
20. G. Poupard and J. Stern: "*On the fly signatures based on factoring*", Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS), pp.48-57, 1999.
21. R. L. Rivest: "*The MD5 message-digest algorithm*", Internet Request for Comments, RFC 1321, 1992.
22. C. P. Schnorr: "*Efficient signature generation by smart cards*", Journal of cryptology, vol.4, Springer-Verlag, pp.161-174, 1991.