Short signature from factoring assumption in the standard model

Zhiwei Wang[*], Guozi Sun, and Danwei Chen
1. College of Computer,
Nanjing University of Posts and Telecommunications
Nanjing, 210046, China
2. State Key Laboratory of Information Security
(Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China)
{zhwwang, sun, chendw}@njupt.edu.cn

Abstract
Programmable hash functions (PHFs) is a new cryptographic primitive, which can mimic certain programmability properties of random oracles. Due to these properties, PHFs are very useful to construct short signatures in standard models. Based on **(m,1)-PHF**, we propose an efficient construction of short signature from factoring problem. Our signature doesn't require the generation of primes at signing, and it can resist the information leakage to some extent.

Keywords: signature, factoring problem, Programmable hash functions, standard model, provable security.

1 Introduction

Digital signatures are one of the most important fundamental cryptographic primitives, which can be used to construct complex cryptographic protocols. However, most practical signatures [2][4] can only be proved secure in random oracles[1][5], while signatures in standard model are often considered as less efficient or based on the stronger assumptions, such as Strong RSA assumption[8][7][10][11], Strong q-Diffie-Hellman assumption[11][3] et al. In Strong q-assumption, an attacker is provided with q random "solved instances" (leaked information) and has to compute a given, fixed instance.

The concept of programmable hash function is proposed by Hofheinz et al.[10], which is an efficiently computable function that maps binary strings into a group $\mathbb{G}$. Programmable hash functions (PHFs) is a keyed group hash function that can behave in two indistinguishable ways, depending on how the key is generated. If the standard key generation algorithm is used, then the hash function fulfills its normal functionality. If the alternative (trapdoor) key generation algorithm is used, the hash function outputs a key that is indistinguishable from the one output by the standard algorithm. The programmability of PHFs is very similar to a scenario we are often confronted with in "provable security", namely, we know some of the hash outputs, but for some we do not. Hofheinz et al. construct a short signature from RSA assumption in the standard model[11]. However, they leave a open problem that *how to design truly practical signatures from RSA or factoring problem without random oracles, especially do not require the generation of primes at signing.*

In this paper, based on **(m, 1)-PHFs** proposed by Hofheinz et al.[11], we propose a simple but very efficient construction of short signature from factoring assumption in the standard model. Our signature scheme doesn't need to generate primes, and it can resist the information leakage to some extent. The rest of this paper is organized as follows: In the next section, we review some preliminaries related to our construction. Then, we propose our short signature scheme in Section 3. The security properties will be analyzed in Section 4. We conclude in Section 5.

Journal of Internet Services and Information Security (JISIS), volume: 2, number: 3/4, pp. 110-117

[*]Corresponding author: Nanjing University of Posts and Telecommunications, Wenyuan Road 9, Xianlin Yadong New Direct, Nanjing, 210023, China, Tel: +86-025-85866427

2 Preliminaries

2.1 Some concepts in number theory

Let $N = p \times q$ be a composite modulus, where p and q are two large prime numbers. Let $\mathbb{Q}_N$ denote the subgroup of squares in $\mathbb{Z}_N^*$. Then, it is well known that $\mathbb{Q}_N$ is a cyclic group with order $\phi(N)/4 = (p-1)(q-1)/4$ [12].

Factoring problem. given a k-bit composite N, which is a multiple of two large primes p and q, to output p or q. Factoring problem is usually considered as a hard problem.

Theorem 2.1 *Let $a \in \mathbb{Q}_N$, $N = p \times q$, where p, q are large primes and $p = 2p' + 1$, $q = 2q' + 1$. p' and q' are also large primes. Then $a^{2d} \equiv a \pmod N$, where $d = (N - p - q + 5)/8$.*

Proof. Since $d = \frac{(N-p-q+5)}{8} = \frac{(p-1)(q-1)+4}{8} = \frac{4p'q'+4}{8}$, then $a^{2d} = a^{p'q'+1} = a \pmod N$.(We note that $\phi(N)/4 = (p-1)(q-1)/4 = p'q'$.)

Indeed, Theorem 2.1 provides a way to compute one square root of a quadratic residue $a \in \mathbb{Q}_N$.

Let N be a multiple of two large primes p,q and $a \in \mathbb{Q}_N$. If s_1 and s_2 are two square roots satisfying $s_1 \neq \pm s_2 \pmod N$, then N could be factored by computing $GCD(s_1 + s_2, N)$ or $GCD(s_1 - s_2, N)$ as the non-trivial divisor of N. However, if $s_1 = \pm s_2 \pmod N$, it will be no useful to the factorization of N. Thus, if given two random square roots, the probability of factoring N is $1/2$.

Lemma 2.1. *Given $\omega \neq 0, l > 0, a \in Q_N$ and $X \in \mathbb{Z}_N^*$ such that $a^\omega = X^{2^l} \pmod N, \omega < 2^l$, the square root y of a can be computed efficiently ($a = y^2 \pmod N$).*

Proof. There exists two integers γ, δ such that $\omega = 2^\gamma(2\delta + 1)$. From the conditions, we can derive that $(a^{2\delta+1})^{2^\gamma} = a^\omega = X^{2^l} \pmod N$, so $a^{2\delta+1} = X^{2^{l-\gamma}} \pmod N$. Let $y = X^{2^{l-\gamma-1}}/a^\delta \pmod N$. Obviously, y is the square root of a, since $y^2 = X^{2^{l-\gamma}}/a^{2\delta} = a \pmod N$.

2.2 Generalized Birthday Bound

We will apply the following lemma [11] the security proof of our signature.

Lemma 2.2. *Let A be set with $|A| = a$. Let $X_1, \cdots, X_q$ be q independent random variables, taking uniformly random values from A. Then the probability that there exist $m+1$ pairwise distinct indices $i_1, \cdots, i_{m+1}$ such that $X_{i_1} = \cdots = X_{i_{m+1}}$ is upper bounded by $\frac{q^{m+1}}{a^m}$.*

2.3 Definition of programmable hash functions

In this section, we will review the definition of programmable hash functions by Hofheinz et al.[10][11]. Programmable hash function is a group hash function H over a group $\mathbb{G}$ with input length k($k \in \mathbb{N}$ is security parameter), which consists of two efficient algorithms **PHF.Gen** and **PHF.Eval**. The probabilistic algorithm **PHF.Gen**(1^k) generates a hash key κ. The deterministic algorithm **PHF.Eval** takes as input a hash function κ and $X \in \{0,1\}^l$, and outputting **PHF.Eval**$(\kappa, X) \in \mathbb{G}$.

Definition 2.1. We say $H = ($**PHF.Eval**, **PHF.Eval**$)$ is (m, n, γ, δ)-programmable, if there exist an efficient trapdoor key generation algorithm **PHF.TrapGen** and an efficient trapdoor evaluation algorithm **PHF.TrapEval** satisfying the following properties:

- The probabilistic trapdoor generation algorithm **PHF.TrapGen**$(1^k, g, h)$ takes as input group elements $g, h \in \mathbb{G}$, and produces a hash function key κ and a trapdoor information τ.
- For all generators $g, h \in \mathbb{G}$, the keys κ generated by **PHF.Gen** and κ' produced by **PHF.TrapGen** are statically $\gamma-$close.

- Taking as input $X \in \{0,1\}^l$ and trapdoor information τ, the deterministic trapdoor evaluation algorithm **PHF.Eval** outputs $a_X, b_X \in \mathbb{Z}$, so that for all $X \in \{0,1\}^l$,

$$\textbf{PHF.Eval}(\kappa, X) = g^{a_X} h^{b_X}$$

- For all generators $g, h \in \mathbb{G}$, all κ generated by **PHF.TrapGen**, and all $X_1, \cdots, X_m \in \{0,1\}^l$ and $Z_1, \cdots, Z_n \in \{0,1\}^l$ satisfying $X_i \neq Z_j$ for all i, j, then

$$Pr[a_{X_1} = \cdots = a_{X_m} = 0 \quad and \quad a_{Z_1}, \cdots, a_{Z_n} \neq 0] \geq \delta,$$

  where $(a_{X_i}, b_{X_i}) = \textbf{PHF.Eval}(\tau, X_i)$ and $(a_{Z_j}, b_{Z_j}) = \textbf{PHF.Eval}(\tau, Z_j)$.

The trapdoor information τ generated by **PHF.TrapGen** depends on two (user-specified) generators g and h from the group. This trapdoor information makes it possible to relate the output of the hash function H to g, h. For the PHFs to be (m, n, γ, δ)-programmable, we need that for all $X_1, \cdots, X_m \in \{0,1\}^l$ and $Z_1, \cdots, Z_n \in \{0,1\}^l$, it is true that $X_i \neq Z_j$ for all i, j, and it holds that $a_{X_i} = 0$ but $a_{Z_j} \neq 0$ with significant probability. Thus, the number of X such that $H(X) = h^{b_X}$ is controlled by m, while n controls the number of Z such that $H(Z) = g^{a_Z} h^{b_Z}$ for $a_Z \neq 0$.

2.4 Definition of signature

A digital signature scheme consists of three algorithms: **KeyGen**, **Sign**, **Verify**.

KeyGen(1^k) This algorithm generates a key pair (pk, sk).

Sign(m, sk) This algorithm takes as input a message m and the secret key sk, produces a signature σ.

Verify(pk, m, σ) This algorithm takes as input a message m, the public key pk, and the signature σ, outputs a decision $b \in \{accept, reject\}$.

Let's review the standard security definition of signature, which is called *existential unforgeability against chosen message attacks* (EUF-CMA)[9]. This definition can be defined by a game between a challenger $\mathscr{C}$ and an attacker $\mathscr{A}$.

Setup: The challenger $\mathscr{C}$ runs KeyGen(1^k) to get (pk, sk), and sends pk to $\mathscr{A}$.

Queries: The attacker $\mathscr{A}$ may adaptively ask $\mathscr{C}$ to sign a number of messages. For a message m_i, the challenger $\mathscr{C}$ returns a signature σ_i under sk.

Forgery: Finally, the attacker $\mathscr{A}$ outputs a forgery signature σ^* on message m^*. We say that the attacker $\mathscr{A}$ wins the game, if the following conditions are satisfied.

1. $Verify(pk, m^*, \sigma^*) = accept$.
2. $m^* \neq m_i$ for all i.

If the success probability of all probability polynomial-time $\mathscr{A}$ is negligible, then we say that the signature scheme is EUF-CMA secure.

3 Our short signature scheme

Let $N = p \cdot q$, p and q are both safe primes. Let $\mathbb{G} = Q_N$ be the group of quadratic residues modulo N. Let $l = l(k)$ and $\lambda = \lambda(k)$ be polynomials. Let

$$H = (\textbf{PHF.Gen}, \textbf{PHF.Eval})$$

be a group hash functions over $\mathbb{G}$ with input length l. We define the signature scheme $Sig_F[H]$=(**KeyGen**, **Sign**, **Verify**) as follows:

KeyGen(1^k): This algorithm selects two large safe $k/2$-bit primes p and q, and computes $N = pq$. Then it generates a group hash function key κ for the group $\mathbb{G} = Q_N$ from the algorithm **PHF.Gen**. Finally, it computes $d = (N - p - q + 5)/8$. The public key is $pk = (N, \kappa)$, and the secret key is $sk = (p, q, d)$.

Sign(sk, M): On input of the secret key sk, and message $M \in \{0,1\}^l$, the signing algorithm selects a random value $s \in \{0,1\}^\lambda$, and computes

$$\sigma = (H(M)^{d^s} \pmod N).$$

We write $H(M)$ shorthand for **PHF.Eval**(κ, M). The signature is $(\sigma, s) \in \mathbb{Z}_N \times \{0,1\}^\lambda$.

Verify($pk, M, (\sigma, s)$): Taking as input the public key pk, message M and signature (σ, s), this algorithm returns *accept* if

$$H(M) = \sigma^{2^s} \pmod N,$$

otherwise returns *reject*.

Correctness. If $\sigma = H(M)^{d^s} \pmod N$, then $\sigma = H(M)^{\frac{1}{2}^s} \pmod N$. So we have $\sigma^{2^s} = H(M) \pmod N$.

Analysis. Our signature is very simple but efficient, which does not require the generation of primes. The signature only involves one element in $\mathbb{Z}_N$ and a small integer. However, we should note that for a message M, if the random value s has been used to generate the corresponding signature, then any random value such that $\leq s$ cannot be used after that.

4 Security proof

From we discussed in Section 2.3, PHFs is useful in groups with hard discrete logarithms and when the trapdoor key generation algorithm does not know the discrete logarithm of h to the basis g. We can program the hash function such that the hash values of all possible choices $X_1, \cdots, X_m$ of m inputs do not depend on g, which is due to $a_X = 0$. At the same time, we can arrange that the hash values of all possible choices $Z_1, \cdots, Z_n$ of n do depend on g, since $a_Z \neq 0$. If $n = 1$, this situation is very similar to the "provable security". The knowledge of discrete logarithms of some hash queries can be used to simulate the leaked information to an attacker, and finally, the attacker produces a signature on its own. If this signature corresponds to a hash query that we do not know the discrete logarithm, we can use it to break an underlying computational assumption. In this section, we will use PHFs to derive proofs of the adaptive security of our signature.

Theorem 4.1 *Let H be a $(m, 1, \gamma, \delta)$-programmable hash function.If there exists a (t, q, ε)-attacker $\mathscr{A}$ breaking the existential forgery under adaptive chosen message attacks of $Sig_F[H]$, then there exists a* *challenger $\mathscr{C}$ that (t',ε')-solves the factoring problem with $t' \approx t$ and $\varepsilon \leq \frac{q(2\varepsilon'+\gamma)}{\delta} + \frac{q^{m+1}}{2^{m\lambda}}$.*

Proof. We first give a brief outline. As [11], two types attackers should be distinguished. For the type I attackers, the forged signature of the form (M^*,σ^*,s^*) with $s^* = s_i$ for some $i \in \{1,\cdots,q\}$. A type II attacker returns a signature with a fresh s^*. We must remark that in the proof of both type I and type II attackers, s^* should not be less than s_i for all $i \in \{1,\cdots,q\}$. Otherwise, $H(M_i)^{(\frac{1}{2})^{s_i}}$ will provide direct help to attacker's eventually forgery, which means that attacker's finally forgery can not be used to solve the factoring problem.

In the proof of adaptively secure, we need to put up a simulation that is able to generate q signatures $(M_i,\sigma_i,s_i)_{i\in\{1,\cdots,q\}}$ for the attacker's adaptively chosen message M_i. However, for a type I attacker, we have to prepare one or more signatures of the form $H(M_i)^{(\frac{1}{2})^{s_i}}$ for the same $s_i = s^*$ that the attacker eventually uses in his forgery. We resolve this complication by using the programmable properties of H. Namely, we first choose all s_i and i such that $s_i = s^*$. Then, we prepare H with generators g and h such that we know all 2^{s_j}th roots of h(for all j), and all 2^{s_j}th roots of g(for all $s_j \neq s_i$). Thus, when a type I attacker asks the signature of the message M_j with $s_j = s_i$, we can make $a_i = 0$, then $H(M_i) \in < h >$, so we can compute $H(M_j)^{(1/2)^{s_j}}$. With this capability, we can simulate the leaked signature information to the attacker. At the same time, $H(M^*) \notin < h >$ has a nontrivial g-factor, so we can get a secure root of g from attacker's forgery, which will be used to factorize N.

Compare the type I attacker, it is easier to treat a type II attacker. To do this, we choose all s_i in advance. Then, we prepare **PHF H** using **PHF.TrapGen**, but relative to generators g and h for which we know 2^{s_i}th roots. This allows to leak signatures for the attacker, and while the attacker outputs a new signature, it essentially outputs a fresh 2^{s^*}th root of g^{a^*}, from which a square root of g can be computed efficiently (lemma 2.1).

Now, we prove **Theorem 4.1** by proving the following two lemmas.

Lemma 4.1 *Let $\mathscr{A}$ be a type I attacker can (t,q,ε)-break the existential forgery under adaptive chosen message attacks of $Sig_F[H]$, then there exists a challenger $\mathscr{C}$ that (t',ε')-solves the factoring problem with $t' \approx t$ and $\varepsilon' \geq \frac{1}{2}(\frac{\delta}{q}(\varepsilon - \frac{q^{m+1}}{2^{m\lambda}}) - \gamma)$.*

Proof. In the interaction game, the attacker $\mathscr{A}$ is only allowed to make q *Sign* oracle queries. If attacker $\mathscr{A}$ can outputs a successful forgery eventually, then we can construct a challenger $\mathscr{C}$ that solves the factoring problem with a non-negligible probability. Given a challenge N, $\mathscr{C}$'s goal is to output the factorization of N.

Setup: Challenger $\mathscr{C}$ chooses $\hat{g},\hat{h} \in \mathbb{Z}_N^*$, and sets $g = \hat{g}^{2^{\sum_{t\in E^*} t}}$ and $h = \hat{h}^{2^{\sum_{t\in E} t}}$, where $E = \bigcup_{i=1}^{q}\{s_i\}$ and $E^* = E \setminus \{s^*\}$. Then it runs **PHF.TrapGen**$(1^k,g,h)$ to generate hash key κ and the trapdoor information τ. Since H is a $(m,1,\gamma,\delta)$-programmable hash function, hash key generated by **PHF.TrapGen** and **PHF.Gen** are statically γ-close. Challenger $\mathscr{C}$ sends (N,κ) to $\mathscr{A}$ as the public key.

Queries: When $\mathscr{A}$ makes a signing queries on message M_i, $\mathscr{C}$ selects a random value $s_i \in \{0,1\}^\lambda$, and runs **PHF.TrapEval**(τ,M_i) to get (a_i,b_i). If $a_i \neq 0$ for some $i \in \{1,\cdots,q\}$ with $s_i = s^*$, then $\mathscr{C}$ aborts. Otherwise, $\mathscr{C}$ computes

$$\sigma_i = \hat{g}^{a_i \cdot 2^{\sum_{t\in E_i^*} t}} \hat{h}^{b_i \cdot 2^{\sum_{t\in E_i} t}} = H(M_i)^{(1/2)^{s_i}},$$

where $E_i = E \setminus \{s_i\}$ and $E_i^* = E^* \setminus \{s_i\}$. Then $\mathscr{C}$ returns (σ_i,s_i) as the answer to $\mathscr{A}$.

Outputs: Eventually, $\mathscr{A}$ outputs a valid forged signature (σ^*, s^*) on message M^*. If $s^* < s_i$ for some $i \in \{1, \cdots, q\}$, then $\mathscr{C}$ aborts. Otherwise, $\mathscr{C}$ extracts the solution to the factoring problem as follows. $\mathscr{C}$ first runs **PHF.TrapEval**(τ, M^*) to get (a^*, b^*), and computes $z = \frac{\sigma^*}{\hat{h}^{b^* \cdot 2^{\Sigma_{t \in E^*} t}}}$. Observe here that

$$\begin{aligned} z^{2^{s^*}} &= \left(\frac{\sigma^*}{\hat{h}^{b^* \cdot 2^{\Sigma_{t \in E^*} t}}}\right)^{2^{s^*}} \\ &= \frac{H(M^*)}{h^{b^*}} \\ &= \frac{g^{a^*} h^{b^*}}{h^{b^*}} \\ &= g^{a^*} \pmod N. \end{aligned}$$

Due to lemma 2.1, we can compute a square root y of g from $z^{2^{s^*}} = g^{a^*} \pmod N$. As discussed above, $\hat{g}^{2^{\Sigma_{t \in E^*} t-1}}$ is another square root of g. If $y \neq \pm \hat{g}^{2^{\Sigma_{t \in E^*} t-1}}$, $\mathscr{C}$ can factorize N by computing $GCD(y - \hat{g}^{2^{\Sigma_{t \in E^*} t-1}}, N)$ or $GCD(y + \hat{g}^{2^{\Sigma_{t \in E^*} t-1}}, N)$. Since $\hat{g}$ is randomly chosen from $\mathbb{Z}_N^*$, then the probability of $y \neq \hat{g}^{2^{\Sigma_{t \in E^*} t-1}}$ is $1/2$.

Probability analysis: We assume that the attacker $\mathscr{A}$ can win the above game with the probability of ε. Since H is a $(m, 1, \gamma, \delta)$-programmable hash function, from lemma 2.2 we can deduce that challenger $\mathscr{C}$ can solve the factoring problem through $\mathscr{A}$'s forgery with the probability of $\frac{1}{2}(\frac{\delta}{q}(\varepsilon - \frac{q^{m+1}}{2^{m\lambda}}) - \gamma)$.

Lemma 4.2 *Let $\mathscr{A}$ be a type II attacker can (t, q, ε)-break the existential forgery under adaptive chosen message attacks of $Sig_F[H]$, then there exists a challenger $\mathscr{C}$ that (t', ε')-solves the factoring problem with $t' \approx t$ and $\varepsilon' \geq \frac{\delta(\varepsilon - \gamma)}{2}$.*

Proof. The proof for type II attackers proceeds similarly. If attacker $\mathscr{A}$ can outputs a successful forgery eventually, then we can construct a challenger $\mathscr{C}$ that solves the factoring problem. $\mathscr{C}$'s goal is to output the factorization of N.

Setup: Challenger $\mathscr{C}$ chooses $\hat{g}, \hat{h} \in \mathbb{Z}_N^*$, and sets $g = \hat{g}^{2^{\Sigma_{t \in E} t}}$ and $h = \hat{h}^{2^{\Sigma_{t \in E} t}}$, where $E = \bigcup_{i=1}^{q} \{s_i\}$. Then it runs **PHF.TrapGen**$(1^k, g, h)$ to generate hash key κ and the trapdoor information τ. Since H is a $(m, 1, \gamma, \delta)$-programmable hash function, hash key generated by **PHF.TrapGen** and **PHF.Gen** are statically γ-close. Challenger $\mathscr{C}$ sends (N, κ) to $\mathscr{A}$ as the public key.

Queries: When $\mathscr{A}$ makes a signing queries on message M_i, $\mathscr{C}$ selects a random value $s_i \in \{0,1\}^\lambda$, and runs **PHF.TrapEval**(τ, M_i) to get (a_i, b_i). $\mathscr{C}$ computes

$$\sigma_i = \hat{g}^{a_i \cdot 2^{\Sigma_{t \in E_i} t}} \hat{h}^{b_i \cdot 2^{\Sigma_{t \in E_i} t}} = H(M_i)^{(1/2)^{s_i}},$$

where $E_i = E \setminus \{s_i\}$. Then $\mathscr{C}$ returns (σ_i, s_i) as the answer to $\mathscr{A}$.

Outputs: Eventually, $\mathscr{A}$ outputs a valid forged signature (σ^*, s^*) on message M^*. If $s^* < s_i$ for some $i \in \{1, \cdots, q\}$, then $\mathscr{C}$ aborts. Otherwise, $\mathscr{C}$ extracts the solution to the factoring problem as follows. $\mathscr{C}$ first runs **PHF.TrapEval**(τ, M^*) to get (a^*, b^*), and computes $z = \frac{\sigma^*}{\hat{h}^{b^* \cdot 2^{\Sigma_{t \in E^*} t}}}$. Obviously, $z^{2^{s^*}} = g^{a^*} \pmod N$. As lemma 2.1, we can compute a square root y of g, and $\hat{g}^{2^{\Sigma_{t \in E} t-1}}$ is another square root of g. If $y \neq \pm \hat{g}^{2^{\Sigma_{t \in E} t-1}}$, $\mathscr{C}$ can factorize N by computing $GCD(y - \hat{g}^{2^{\Sigma_{t \in E} t-1}}, N)$ or $GCD(y + \hat{g}^{2^{\Sigma_{t \in E} t-1}}, N)$. Since $\hat{g}$ is randomly chosen from $\mathbb{Z}_N^*$, then the probability of $y \neq \hat{g}^{2^{\Sigma_{t \in E} t-1}}$ is $1/2$.

Probability analysis: We assume that the attacker $\mathscr{A}$ can win the above game with the probability of ε. Since H is a $(m,1,\gamma,\delta)$-programmable hash function, we can deduce that challenger $\mathscr{C}$ can solve the factoring problem through $\mathscr{A}$'s forgery with the probability of $\frac{\delta(\varepsilon-\gamma)}{2}$.

This completes our proof.

Note: With $(m,1)$-programmable hash functions, we can simulate the leaked signature information to the attacker. The above proof shows that our signature scheme can resist the information leakage to some extent. Furthermore, our signature scheme is proved secure under factoring problem, while Hofheinz et al.'s construction is proved secure under RSA. However, the hardness of RSA problem and factoring problem are not identical. It is generally believed that RSA assumption is stronger than factoring assumption[6].

5 Conclusion

Programmable hash function is a new cryptographic primitive, which can be plugged into the construction of signatures. The $(m,1)$-programmable hash functions are very similar to the random oracle, by which some short signature schemes have been proposed. We propose a simple but efficient short signature scheme from factoring problem based on **(m, 1)-PHF**, which does not need to generate primes, and can resist the insider information leakage to some extent. Compared with the previous schemes, it may be a truly practical signature scheme in the standard model.

Acknowledgments.

This research is supported by State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093), the Natural science fund for colleges and universities in Jiangsu Province No. 11KJB520015, and Program for Excellent Talents in Nanjing University of Posts and Telecommunications under Grant No.NY209014.

References

[1] M. Bellare and T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In *Proc. of Advances in Cryptology - EUROCRYPT 2009, LNCS*, volume 5479, pages 407–424. Springer-Verlag, December 2009.

[2] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *Proc. of Advances in Cryptology - EUROCRYPT'96, Saragossa, Spain, LNCS*, volume 1070, pages 399–416. Springer-Verlag, May 1996.

[3] D. Boneh and X. Boyen. Short signatures without random oracles. In *Proc. of Advances in Cryptology - EUROCRYPT 2004, Interlaken, Switzerland, LNCS*, volume 3027, pages 56–73. Springer-Verlag, May 2004.

[4] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Proc. of Advances in Cryptology - ASIACRYPT 2001, Gold Coast, Australia, LNCS*, volume 2248, pages 514–532. Springer-Verlag, December 2001.

[5] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *Proc of the 30th Annual ACM Symposium on Theory of Computing, Dallas, Texas, USA*, pages 209–218. ACM, May 1998.

[6] Z. Cao, H. Zhu, and R. Lu. Provably secure robust threshold partial blind signature. *Science in China Series F: Information Sciences*, 49(5):604–615, May 2006.

[7] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *Proc of the 6th ACM Conference on Computer and Communications Security (CCS'99), Kent Ridge Digital Labs, Singapore*, pages 46–51. ACM, November 1999.

[8] M. Fischlin. The Cramer-Shoup strong-RSA signature scheme revisited. In *Proc. of the 6th International Workshop on Theory and Practice in Public Key Cryptography (PKC'03), Miami, USA, LNCS*, volume 2567, pages 6–8. Springer-Verlag, December 2003.

[9] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[10] D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In *Proc. of Advances in Cryptology - CRYPTO 2008, Santa Barbara, California, USA, LNCS*, volume 5157, pages 21–38. Springer-Verlag, August 2008.

[11] D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. http://eprint.iacr.org/2011/296, October 2011.

[12] S. V. *A Computational Introduction to Number Theory and Algebra.* Cambridge University Press, 2005.

Zhiwei Wang is an associate professor in the college of computer, Nanjing University of Posts and Telecommunications. Zhiwei Wang received his BS. degree in the Computer Science from East China University of Science and Technology, Shanghai and the Ph.D. degree in the Cryptogaphy from Beijing University of Posts and Telecommunications. His recent research interests include: Digital signature, Provably Security, Multivariate public key cryptography, Network security, Cloud security. He has published more than 20 papers in the area of information security. He was the TPC member for MobiPST 2011, MobiPST 2012, ACSA-Summer 2012, ICITIS 2012, MIST 2012.

Guozi Sun is a Professor in the College of Computer, Nanjing University of Posts and Telecommunications. He received his Ph.D. degree in Computer Engineering from Nanjing University of Aeronautics and Astronautics. His recent research interests include: Wireless sensor network security, Cryptography protocol, Software security etc, He has published many papers in these area.

Danwei Chen is a Chairman and Professor in the College of Computer, Nanjing University of Posts and Telecommunications. He received his PhD. degree in the Electrical Engineering from Nanjing University of Aeronautics and Astronautics. His recent research interests include: Network security, Software security, Internet of Things security etc, He was a editor of Journal of Nanjing University of Posts and Telecommunications.