1-1-2015

Secure delegation of signing power from factorization

Yong Yu
University of Wollongong, yyong@uow.edu.au

Man Ho Au
University of Wollongong, aau@uow.edu.au

Yi Mu
University of Wollongong, ymu@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Huai Wu
University of Electronic Science and Technology of China, huai@uow.edu.au

Secure delegation of signing power from factorization

Abstract

Delegation of signing is a working way common in oce automation work, and is also an important approach to establish trust. Proxy signature is an important cryptographic primitive for delegating the signing powers and it has found many real world applications. The existing proxy signature schemes from factorization assumption are either insecure or inecient. In this paper, we propose a novel, ecient and provably secure proxy signature scheme from factorization. Our construction makes use of a factorization based key-exposure free chameleon hash function in the delegation phase and the proxy signer needs only to nd a collision to a chameleon hash value to generate a valid proxy signature. As a result, our scheme is highly ecient in terms of the computation of a proxy signature. We also provide a formal security proof by classifying the adversaries into three categories. Comparisons demonstrate that the new scheme outperforms the known ones in terms of security, computational eciency and the length of the public key.

Disciplines

Engineering | Science and Technology Studies

Publication Details

Yu, Y., Au, M. Ho., Mu, Y., Susilo, W. & Wu, H. (2015). Secure delegation of signing power from factorization. The Computer Journal, 58 (4), 867-877.

Secure Delegation of Signing Power from Factorization

Yong Yu[1,2], Man Ho Au[2,*], Yi Mu[2], Willy Susilo[2], Huai Wu[1]

1. School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 610054, PR China
2. Centre for Information and Computer Security Research, School of Computer Science and Software Engineering, University of Wollongong, Wollongong, NSW 2522, Australia
Email: aau@uow.edu.au

Delegation of signing is a working way common in office automation work, and is also an important approach to establish trust. Proxy signature is an important cryptographic primitive for delegating the signing powers and it has found many real world applications. The existing proxy signature schemes from factorization assumption are either insecure or inefficient. In this paper, we propose a novel, efficient and provably secure proxy signature scheme from factorization. Our construction makes use of a factorization based key-exposure free chameleon hash function in the delegation phase and the proxy signer needs only to find a collision to a chameleon hash value to generate a valid proxy signature. As a result, our scheme is highly efficient in terms of the computation of a proxy signature. We also provide a formal security proof by classifying the adversaries into three categories. Comparisons demonstrate that the new scheme outperforms the known ones in terms of security, computational efficiency and the length of the public key.

Keywords: Digital signature, Proxy signature, Factorization, Provable security

Received 00 January 2013; revised 00 Month 2013

1. INTRODUCTION

The notion of proxy signature [1] was put forth by Mambo et al., which enables a proxy signer to sign on behalf of an original signer in the case where the original signer is unavailable (eg. temporal absence, lack of computational power etc.). Subsequently, many proxy signature schemes with their variants, analysis, improvements and applications have been proposed in the literature [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]. Proxy signatures have been suggested for extensive usage in numerous practical applications such as in distributed computing, e-commerce, e-cash, and grid computing where delegation of rights is quite common [5, 6, 7]. Mambo et al. [1] classified proxy signatures into two categories: proxy-unprotected and proxy-protected. A proxy-protected scheme, where only the proxy signer is able to generate valid proxy signatures, is more practical since it accommodates some highly desirable properties such as fairness and signature ownership. In an orthogonal dimension, proxy signatures can also be classified according to their delegation types, including full delegation, partial delegation, delegation with warrant and threshold delegation. At present, delegation with warrant is the most popular choice because of its high security and flexible delegation policy. Thus, we limit our attention to proxy-protected proxy signature authorized by using delegation with warrant in this paper.

The security of the majority existing proxy signature schemes depends on the discrete logarithm assumption and some intractable assumptions on elliptic curves. To avoid putting all eggs in one basket, Shao [19] proposed two proxy signature schemes based on the factorization assumption. However, no formal security analysis was provided. Subsequently, Shao [20] designed a new proxy signature scheme from factorization and proved the security in the random oracle model. Zhou et al.[21] described two efficient proxy-protected signature schemes based on RSA assumption and the factoring assumption respectively. Unfortunately, Park et al. [22] showed that both schemes are vulnerable to outsider attacks. In 2007, Liu et al. [23] proposed an improvement of Zhou et al.'s scheme and claimed that the new construction satisfies all the desirable security requirements of a secure proxy signature. However, Hu et al. [24] discovered that Liu et al.'s scheme [23] is not secure against a malicious original signer's attack. In 2012, Yu et al. [33] proposed a provably secure proxy signature scheme from factorization. The construction in [33] is conceptually simple and elegant: the original signer issues a standard digital signature on the public key of the proxy signer together with the delegation policy under which the proxy signer is authorized to sign

on behalf of the original signer. This standard signature is viewed as a warrant. A proxy signature from this proxy signer is a standard signature on the message under the proxy's public key together with the warrant. To verify a proxy signature, anyone can check if the warrant is a valid signature from the original signer on the public key of the proxy signer and the delegation policy, followed by the validation of the signature under the proxy's public key. Yu et al. [33] instantiated the above idea based on an improved Rabin-type signature [25]. The scheme was shown secure against both outsider attackers and insider attackers.

It is relatively straightforward to construct proxy signature schemes following the two signature approach discussed above, and the security of the resulting scheme depends on the unforgeability of the underlying digital signature scheme. While this approach is clever, it imposes a lower bound on the time and space complexities of the resulting proxy signature since it can be viewed as a consecutive execution of two digital signature instances. In this paper, our goal is to construct an efficient secure proxy signature scheme that is secure under hardness of factorization, a fundamental and well-studied intractable problem in number theory. By efficient we require that the time and space complexity should be less than the cost of running two standard signature instances and thus another approach is adopted.

1.1. Related Work

Chameleon Hash Our approach in the construction of proxy signatures involves the use of another cryptographic primitive called Chameleon hash function introduced by Krawczyk and Rabin [34]. Roughly speaking, chameleon hash, also known as trapdoor hash, is a trapdoor one-way function, which prevents anyone except the holder of the trapdoor from finding the collisions of a random input. Specifically, to evaluate the hash of an input m, a random number r is chosen and the function takes m, r as input for evaluation. Given the trapdoor of the chameleon hash function, the original input m and r, together with another input m', the trapdoor holder could output another randonness r' such that (m, r) and (m', r') evaluates to the same hash value. This is a useful primitive for constructing chameleon signatures [34], online/offline signatures [35, 36] etc. Ateniese and Mederious [37] discussed the key exposure problem in the original formulation of chameleon hash function due to Krawczyk et al.[34]. Specifically, while the trapdoor holder can generate collisions (two distinct input that "hashes" to the same output), the trapdoor can be computed based on any pair of collisions. They mentioned that this phenomena is undesirable in applications in which the pair of collisions will be released eventually and introduced identity-based chameleon hash [37] to solve the problem. Subsequently, many novel chameleon hash schemes without the key-exposure problem were proposed such as in [38, 37, 39, 40].

The use of Chameleon Hash in Proxy Signatures Mehta and Harn [41] firstly introduced the idea of using chameleon hash to construct novel proxy signature schemes where verifiers need not be aware of the existence of a proxy. However, in their scheme, the proxy signer can only sign one time since two valid proxy signatures will lead to the exposure of his trapdoor. Recently, Chandrasekhar et al. [42] presented a technique of using trapdoor hash functions to build proxy signature that does not restrict the number of signatures a proxy can generate per delegation. We first review their generic construction and highlight some of the subtle issues when we describe their concrete instantiation from the discrete logarithm assumption. In their generic approach, the trapdoor hash function is used as the public key of a proxy while the trapdoor is the secret key. When the delegator wishes to delegate his signing right to this proxy, he creates a signature σ on a value h concatenated with the warrant w, where h is the output of the trapdoor hash function of the proxy on input w and randomness r. When the proxy wishes to create a proxy signature on a message m, it uses his trapdoor to compute randomness c such that (m, c) also 'hashes" to h. A proxy signature then consists of the signature σ from the original signer on the chameleon hash value h concatenated with the warrant w, the warrant w, as well as the randomness r and c. The verification of the proxy signature involves computing h which is the chameleon hash of w with randomness r, validating σ on $h||w$ and that the message m together with randomness c "hashes" to h. Note that this approach is novel in that it does not involve operations on two signatures.

Obviously, the approach requires key exposure-free chameleon hash function since a proxy signatures consist of a collision. Chandrasekhar et al. proposed a discrete-logarithm based chameleon hash function for the purpose. A detailed analysis of their concrete construction reviewed that their DL-based construction is not a straightforward instantiation of the above conceptual idea. Firstly, h is not just the trapdoor hash of the warrant w. Instead, it is the trapdoor hash of another value which is the hash of the public key of the proxy concatenated with the warrant. Secondly, σ is not a signature on $h||w$. Rather, it is a Schnorr signature on the string $h||w||r$ and that the randomness r is also part of the Schnorr signature. In other words, r is part of the signature from the original signer as well as the randomness used in the evaluation of the chameleon hash. On one hand, this modification from the above conceptual idea allows the compression of the proxy signature via safe re-use of randomness. On the other hand, these deviations from the generic construction appeared to be necessary to prevent a malicious proxy from creating another chameleon hash function and randomness r' such that (w, r') also "hashes" to the same value h for this new chameleon hash function. The

use of the same randomness r in the signature and the chameleon hash function evaluation tied the signature from the original signer to the specific hash value being signed. This technique is especially suitable for Schnorr-type signatures [26], where the randomness used can be generated before the message to be signed is known. Thus, it is non-trivial to construct factorization-based proxy signatures following this conceptual approach. It involves subtle modifications in addition to choosing appropriate building blocks that are secure under the factorization assumption.

Our contributions. To our best knowledge, there has not been such a factorization-based proxy signature scheme that achieves provable security, efficiency and novelty simultaneously. Therefore, our main contribution of this paper is to fill this gap by providing a provably secure and non-consecutive proxy signature scheme from factorization. Specifically, we propose a new proxy signature scheme by incorporating factorization-based chameleon hash function and improved Rabin-type digital signatures [27] where $n = pq$ is a Williams integer. We show that our scheme achieves all the desirable properties a secure proxy signature should possess and provide formal security proofs as well under the factorization assumption in the random oracle model. The theoretical analysis and the experimental results demonstrate that the new scheme offers stronger security, higher computational efficiency and shorter public key simultaneously.

Organization: Section 2 reviews some preliminaries used in this paper. Section 3 describes the formal model of proxy signature schemes. Section 4 presents our proxy signature scheme and the comparisons with other schemes. Section 5 provides security analysis of the new scheme. Section 6 concludes the paper.

2. PRELIMINARIES

In this section, we review some basic knowledge used in this paper, including quadratic residue, Legendre symbol, Jacobi symbol and the factorization assumption [28, 29].

2.1. Related definitions and facts

Definition 1. Let $a \in Z_n^*$, the multiplicative group of Z_n. a is said to be a quadratic residue modulo n, if there exists an $x \in Z_n^*$ such that $x^2 \equiv a \pmod{n}$. If no such x exists, a is called a quadratic nonresidue modulo n. The set of all quadratic residues modulo n is denoted by Q_n and the set of all quadratic nonresidues is denoted by $\bar{Q}_n$.

Fact 1. Let $n = pq$, a product of two distinct odd primes p and q. Then $a \in Z_n^*$ is a quadratic residue modulo n if and only if $a \in Q_p$ and $a \in Q_q$. It follows that $|Q_n| = |Q_p| \cdot |Q_q| = \frac{(p-1)(q-1)}{4}$ and $|\bar{Q}_n| = \frac{3(p-1)(q-1)}{4}$.

Definition 2. Let p be an odd prime and a an integer. The Legendre symbol $\left(\frac{a}{p}\right)$ is defined by

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{if } a \in Q_p, \\ -1, & \text{if } a \in \bar{Q}_p, \\ 0, & \text{if } p \mid a. \end{cases}$$

When the modulo n is a composite number, Jacobi symbol $(\frac{a}{n})$, a generalization of the Legendre symbol, will be involved.

Definition 3. Let $n \geq 3$ be odd with prime factorization $n = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$, the Jacobi symbol $\left(\frac{a}{n}\right)$ is defined by the formula

$$(\frac{a}{n}) = (\frac{a}{p_1})^{e_1} (\frac{a}{p_2})^{e_2} \cdots (\frac{a}{p_t})^{e_t}.$$

Definition 4. A Blum integer is a composite integer of the form $n = pq$, where p and q are distinct primes each congruent to 3 modulo 4.

Fact 2. If $n = pq$ is a Blum integer, the function $f : Q_n \to Q_n$ defined by $f(x) \equiv x^2 \pmod{n}$ is a permutation. The inverse function of f is

$$f^{-1}(x) \equiv x^{\frac{(p-1)(q-1)+4}{8}} \pmod{n}.$$

2.2. Complexity assumptions

Integer Factorization Problem. Given n, an odd composite integer with at least two distinct large prime factors, output a prime number p such that $p|n$. The probability that a polynomially bounded algorithm $\mathcal{A}$ can solve the integer factorization problem is defined as $Succ_{\mathcal{A}}^{IF} = \Pr[p \leftarrow \mathcal{A}(n)]$.

Integer Factorization Assumption. Given an odd composite integer n with distinct large prime factors, $Succ_{\mathcal{A}}^{IF}$ is negligible.

3. FORMAL MODELS OF PROXY SIGNATURE SCHEMES

In this section, we briefly review the components and security model of proxy signature schemes [14, 6].

3.1. The compents of proxy signature schemes

Two participants, namely an original signer Alice and a proxy signer Bob, are involved in a proxy signature scheme. The scheme consists of the following five algorithms: Setup, KeyGen, DelGen, ProxySign and Verification.

Setup: On input the system security parameter, this algorithm generates system's parameters.

KeyGen: On input the system's parameters, this algorithm outputs secret-public key pairs for Alice and Bob.

DelGen: On input the system's parameters, Alice's secret key sk_a and the warrant w, this algorithm generates a delegation σ_w of w.

ProxySign: On input the system's parameters, the warrant w, the delegation σ_w, the secret key sk_b of the proxy signer and the message m to be signed, this algorithm generates a proxy signature σ on m under the warrant w.

Verification: On input the system's parameters, original signer's public key pk_a, proxy signer's public key pk_b, the warrant w, the signed message m and the alleged signature σ_m, this algorithm outputs True if σ is a valid proxy signature on the message m and outputs $\perp$ otherwise.

3.2. Security requirements for proxy signature schemes

Since the invention of proxy signature, several security requirements of this cryptographic primitive have been specified to satisfy the different situations. It has been widely accepted that a secure proxy signature scheme should provide the following requirements [6, 5].

Verifiability: From a valid proxy signature, a verifier can be convinced of the original signer's agreement on the signed message.

Strong undeniability: Once a proxy signer generates a valid proxy signature on behalf of the original signer, the proxy signer cannot deny his signature generation against anyone else.

Strong identifiability: For a valid proxy signature, anyone can determine the identity of the corresponding proxy signer.

Strong unforgeability: Only the designated proxy signer can generate valid proxy signatures on behalf of the original singer. Namely, any other third party, including the original signer, can not generate a valid proxy signature in the name of the proxy signer.

Prevention of misuse: The proxy signing key cannot be used for purposes other than generating valid proxy signatures. In the case of misuse, the responsibility of proxy signer should be determined explicitly.

Among all the essential security requirements, the property of **strong unforgeability** is the most desirable, which essentially indicates the undeniability and prevention of misuse. To capture the nature of strong unforgeability well, we adopt the security model of proxy signatures due to Huang et al. [14] to prove the security of our scheme. Besides outside adversaries, two types of inside adversaries are also considered in the model.

Type 1: As an outside adversary, $\mathcal{A}_1$ only has the public keys of Alice and Bob.

Type 2: As a malicious proxy signer, $\mathcal{A}_2$ additionally holds the secret key of the proxy signer Bob.

Type 3: $\mathcal{A}_3$ simulates a malicious original signer and has the secret key of the original signer Alice.

Obviously, if a proxy signature scheme is unforgeable against Type 2 and Type 3 adversaries, it is unforgeable against Type 1 adversary as well.

3.3. Security models

Existential unforgeability against adaptive $\mathcal{A}_2$ adversary: This notion captures that it is hard for an adversary to forge a valid proxy signature on message m under a warrant w, if he does not have the delegation of w.This property is defined using the following game between a challenger $\mathcal{C}$ and a Type 2 adversary $\mathcal{A}_2$.

Setup: The challenger $\mathcal{C}$ runs the Setup algorithm to obtain system's parameters, and runs KeyGen algorithm to obtain key pairs (sk_a, pk_a) and (sk_b, pk_b) of the original signer and the proxy signer. $\mathcal{C}$ then forwards (pk_a, pk_b, sk_b) to $\mathcal{A}_2$.

Delegation queries: $\mathcal{A}_2$ requests the delegations on warrants w_i adaptively. $\mathcal{C}$ runs the DelGen algorithm to obtain σ_{w_i} and returns it to $\mathcal{A}_2$.

ProxySign queries: $\mathcal{A}_2$ requests a proxy signature on a message m_i under a warrant w_i adaptively. In response, $\mathcal{C}$ firstly runs DelGen algorithm to obtain the delegation on w and then, runs the ProxySign algorithm to get a proxy signature σ and returns it to $\mathcal{A}_2$.

Output: Finally, $\mathcal{A}_2$ outputs a triple (m^*, w^*, σ^*), such that

1. σ^* is a valid proxy signature on message m^* under warrant w^*.
2. w^* has not been queried during the Delegation queries.
3. (m^*, w^*) has not been queried during the ProxySign queries phase.

The success probability of an adversary $\mathcal{A}_2$ winning above game is defined as $Succ_{A_2}$. $\mathcal{A}_2$ can $(t, q_w, q_{ps}, \epsilon)$ break a proxy signature scheme if it makes at most q_w Delegation queries, q_{ps} ProxySign queries, and runs in time at most t and $Succ_{A_2}$ is at least ϵ.

Existential unforgeability against adaptive $\mathcal{A}_3$ adversary: This notion says that it is difficult for a malicious original signer to generate a valid proxy signature on a message m^* which was not singed by the proxy signer. It is defined using the following game between an adversary $\mathcal{A}_3$ and a challenger $\mathcal{C}$.

Setup: The challenger $\mathcal{C}$ runs the Setup algorithm to obtain system's parameters, and runs KeyGen algorithm to obtain key pairs (sk_a, pk_a), (sk_b, pk_b) of the original signer and the proxy signer. $\mathcal{C}$ then sends (pk_a, pk_b, sk_a) to $\mathcal{A}_3$.

ProxySign queries: $\mathcal{A}_3$ requests a proxy signature on the pair (m_i, w_i) with the original signer Alice and the proxy signer Bob. In response, $\mathcal{C}$ outputs a valid proxy signature σ_i and returns it to $\mathcal{A}_3$.

Output: Finally, $\mathcal{A}_3$ outputs a new triple (m^*, w^*, σ^*), such that

1. σ^* is a valid proxy signature on message m^* under warrant w^*.
2. (m^*, w^*) has not been queried during the ProxySign queries phase.

The success probability of an adversary $\mathcal{A}_3$ winning above game is defined as $Succ_{A_3}$. We say that $\mathcal{A}_3$ can (t, q_{ps}, ϵ) break a proxy signature scheme if $\mathcal{A}_3$ makes at most q_{ps} proxy signature queries in time at most t and $Succ_{A_3}$ is at least ϵ.

4. OUR CONSTRUCTION

In this section, we first describe our design philosophy and the choice of our building blocks, followed by a detailed description of our chameleon hash-based proxy signature scheme based on factorization. Then, we will compare the performance of the new scheme with the existing ones in the same style.

4.1. Design Philosophy

Our approach is similar to the approach due to Chandrasekhar et al. [42] with some minor modifications. Specifically, we require that the public key of the proxy signer is included in the warrant. A delegation is made possible when the original signer creates a signature on the chameleon hash of the warrant based on the chameleon hash function of the proxy. With this minor modification, it is not necessary for the signature from the original signer to include the randomness of the chameleon hash nor the signature itself (as in the DL-based instantiation of Chandrasekhar et al.) and allows us to use the efficient Rabin-PDH signature due to Gentry [25], a deterministic signature scheme with signature size of one single group element described below.

In STOC 1989, Vallée [30] provided an elegant analysis of the distribution, in Z_n, of integers in $B_{n,h,h'} = \{x \in [1, n) : h \leq x^2 \pmod n \leq h'\}$ for $h' - h \geq 8n^{\frac{2}{3}}$, i.e., integers with modular squares in a narrow interval. In Crypto 2002, based on Vallée's conclusion, Coron[31] investigated the security of partial-domain hash signature schemes and showed that for $e = 2$, partial-domain hash signature schemes are provably secure in the random oracle model, assuming that factoring is hard, if the size of the hash function is larger than 2/3 of the modulus size. In Crypto 2004, Gentry [25] improved Coron's results for Rabin-PDH and proposed a specific variant of the improved Rabin-PDH. We also developed a new factorization-based chameleon hash function by introducing double trapdoor to Shamir's scheme [35] as a building block of our construction.

4.2. Description of the new scheme

As mentioned our construction is inspired by Gentry's signature scheme [25], together with our new Harn-like chameleon hash function [36], and is described as follows.

Setup: The original signer Alice randomly picks two large primes p_0 and q_0 satisfying $p_0 \equiv 3 \bmod 8$, $q_0 \equiv 7 \bmod 8$ as her private key and computes $n_0 = p_0 q_0$ as her public key. The proxy signer picks two safe primes p_1, q_1 and computes $n_1 = p_1 q_1$. Compute $p_1' = \frac{p_1 - 1}{2}$, $q_1' = \frac{q_1 - 1}{2}$ and $\lambda(n_1) = lcm(p_1 - 1, q_1 - 1) = 2p'q'$. Choose an element $g \in Z_{n_1}^*$ of order $\lambda(n_1)$. Let H_1, H_2 and H_3 be three secure hash functions where $H_1 : \{0,1\}^* \to [h, h')$ with $h'' = h' - h \pmod{n_0} \geq 8n_0^{\frac{2}{3}}$, $H_2 : \{0,1\}^* \to Z_{n_1}$ and $H_3 : \{0,1\}^* \to Z_{n_1}$. The public chameleon hash key of the proxy signer is (n_1, g) and the private trapdoor key is (p_1, q_1).

Delegation: The original signer delegates her signing power to a proxy signer, say Bob, as follows.

1. Alice makes a warrant w specifying her public key, the public chameleon hash key of the proxy signer, a period of validity, and restrictions on the messages the proxy signer is allowed to sign.
2. Bob picks a random integer k_1, computes $r_1 \equiv g^{k_1} \pmod{n_1}$ and forwards r_1 to Alice.
3. Alice chooses a random integer t_0, computes $h_w = H_1(r_1 g^{H_2(w, r_1)||t_0} \pmod{n_1}, w)$ and a_0, b_0 as follows.

$$a_0 = \begin{cases} 0, & \text{if } (\frac{h_w}{n_0}) = 1, \\ 1, & \text{if } (\frac{h_w}{n_0}) = -1. \end{cases}$$

$$b_0 = \begin{cases} 0, & \text{if } (\frac{2^{-a_0} h_w}{p_0}) = (\frac{2^{-a_0} h_w}{q_0}) = 1, \\ 1, & \text{if } (\frac{2^{-a_0} h_w}{p_0}) = (\frac{2^{-a_0} h_w}{q_0}) = -1. \end{cases}$$

4. Alice computes s_0 using her private key p_0, q_0,

$$s_0 \equiv (2^{-a_0} h_w)^{\frac{n_0 - p_0 - q_0 + 5}{8}} \pmod{n_0}$$

and sends (w, t_0, s_0, a_0, b_0) to Bob via a secure channel.

5. Receiving $(w,\ t_0,\ s_0,\ a_0,\ b_0)$, Bob computes $h'_w = H_1(r_1 g^{H_2(w,r_1)||t_0} \pmod{n_1}, w)$ and verifies the delegation by checking if $s_0^2 \equiv 2^{-a_0} \cdot h'_w \cdot (-1)^{b_0} \pmod{n_0}$ holds.

ProxySign: Bob signs a message $m \in \{0,1\}^*$ on behalf of Alice according to the warrant with his chameleon hash trapdoor as follows.

1. Check whether m conforms to the warrant w. If the check fails, output "invalid" and abort. Otherwise, go to the next step.
2. Pick a random $k_2 \in Z_{\lambda(n_1)}$ and compute $r_2 \equiv g^{k_2} \pmod{n_1}$.
3. Compute $t_1 \equiv t_0 + (k_1 - k_2) + 2^{\lambda(n_1)}(H_2(w,r_1) - H_3(m,r_2)) \bmod \lambda(n_1)$.
4. The final proxy signature on m under the warrant w is $\sigma = (a_0,\ b_0,\ s_0,\ t_0,\ r_1,\ r_2,\ t_1)$.

Verification: To check whether $\sigma = (a_0, b_0, s_0, t_0, r_1, r_2, t_1)$ is a valid proxy signature on m under the warrant w, a verifier performs the following operations.

1. Check whether m conforms to w. If the check fails, output "invalid" and abort. Otherwise, go to the next step.
2. Check whether $r_1 g^{H_2(w,r_1)||t_0} \stackrel{?}{\equiv} r_2 g^{H_3(m,r_2)||t_1} \pmod{n_1}$.
3. Compute $h'_m = H_1(r_2 g^{H_3(m,r_2)||t_1} \bmod n_1, w)$ and verify whether $s_0^2 \stackrel{?}{\equiv} 2^{-a_0} \cdot h'_m \cdot (-1)^{b_0} \pmod{n_0}$. If it holds, the proxy signature is valid; Otherwise, invalid.

4.3. Correctness

The correctness of the scheme is not self-evident and we interpret it in detail. The integer $n_0 = p_0 q_0$ where $p_0 \equiv 3 \bmod 8$, $q_0 \equiv 7 \bmod 8$ is called Williams integer. Obviously, a Williams integer is also a Blum integer. Assume $p_0 = 8t_1 + 3$ and $q_0 = 8t_2 + 7$ where $t_1, t_2 \in Z$, we can observe that

$$(\frac{2}{p_0}) = (-1)^{\frac{p_0^2-1}{8}} = (-1)^{\frac{64t_1^2+48t_1+8}{8}} = -1,$$

$$(\frac{2}{q_0}) = (-1)^{\frac{q_0^2-1}{8}} = (-1)^{\frac{64t_2^2+112t_2+48}{8}} = 1,$$

thus, $(\frac{2}{n_0}) = (\frac{2}{p_0}) \cdot (\frac{2}{q_0}) = -1$. Therefore, if $h'_w = H_1(r_1 g^{H_2(w,r_1)||t_0} \bmod n_1, w)$ equals h_w and (w, t_0, s_0, a_0, b_0) is a valid delegation, then

$$((2^{-a_0} h_w)^{\frac{n_0-p_0-q_0+5}{8}})^2 = \begin{cases} 2^{-a_0} h_w, & \text{if } 2^{-a_0} h_w \in Q_n, \\ -2^{-a_0} h_w, & \text{if } 2^{-a_0} h_w \in \bar{Q}_n. \end{cases}$$

Therefore, the verification of delegation $s_0^2 \equiv 2^{-a_0} \cdot h'_w \cdot (-1)^{b_0} \pmod{n_0}$ always holds.

Regarding the correctness of the proxy signature, since **Delegation** and **ProxySign** share the same signature algorithm, we just show the chameleon hash of the message m is identical to that of the warrant w.

$$\begin{aligned} & r_2 g^{H_3(m,r_2)||t_1} \\ \equiv\ & g^{k_2} g^{H_3(m,r_2)||t_0 + (k_1-k_2) + 2^{\lambda(n_1)}(H_2(w,r_1) - H_3(m,r_2))} \\ \equiv\ & g^{k_2 + 2^{\lambda(n_1)} H_3(m,r_2) + t_0 + k_1 - k_2 + 2^{\lambda(n_1)}(H_2(w,r_1) - H_3(m,r_2))} \\ \equiv\ & g^{k_1 + t_0 + 2^{\lambda(n_1)} H_2(w,r_1)} \\ \equiv\ & r_1 g^{H_2(w,r_1)||t_0} \pmod{n_1} \end{aligned}$$

We can conclude that $h'_w = h'_m$, and as a consequence, $s_0^2 \equiv 2^{-a_0} \cdot h'_m \cdot (-1)^{b_0} \pmod{n_0}$ also holds. ◇

4.4. Performance comparisons and experimental results

We compare the security and efficiency of the new scheme with those of existing ones in the same style[21, 20, 23, 33], i.e., whose security is related to the integer factorization problem or the RSA problem. Several expensive cryptographic operations are considered. We denote by *mul*, *exp*, and *inv* the multiplication, the modular exponentiation and the inversion computation respectively. *root* is denoted as the operation to find the solutions of $x^2 \equiv a \pmod{n}$ when the factor of n is given. The comparisons of six known schemes are summarized in Table 1. The **Delegation** column, **Proxysign** column and **Verify** column demonstrate the computation costs of each scheme. **Security** column denotes the intractable assumptions that the schemes rely on. **Against** $\mathcal{A}_2$ and **Against** $\mathcal{A}_3$ columns show whether the scheme is secure against the attacks mounted by malicious proxy signers and malicious original signers respectively. The symbol N means that the scheme is vulnerable to this kind of attack while Y indicates that it is secure against the attack. The schemes in [21] are insecure because they can resist neither a malicious proxy signer's attack nor a malicious original signer's attack. Although the scheme in [23] is secure against a malicious proxy signer's attack, it suffers a malicious original signer's attack. Thus, this scheme is also insecure. The constructions in [20, 33] and our new proposal are secure against all kinds of adversaries. Our scheme outperforms the one in [20] in terms of security, computational efficiency and length of the public key. Specifically, the unforgeability of Shao's proxy signature [20] depends on RSA problem while the unforgeability of our scheme relies on factorization. Consequently, our scheme is at least as secure as Shao's scheme, but possibly more secure [27]. Furthermore, our scheme is a bit more efficient than the scheme in [20] because 1 exponentiation, the most expensive operation in the schemes, is less required. The employment of Williams integer helps to avoid searching an integer a satisfying Jacobi symbol $(\frac{a}{n}) = -1$ and including it in

TABLE 1. Comparisons of six proxy signature schemes from factorization.

Schemes	Delegation	Proxysign	Verify	Security	Against $\mathcal{A}_2$	Against $\mathcal{A}_3$
ZCL1[21]	1 *exp*	2 *exp*+1 *mul*	2 *exp*+1 *inv*+1 *mul*	RSA	N	N
ZCL2[21]	1 *root*	1 *exp*+2 *mul*+1 *root*	5 *mul*+1 *inv*	Factorization	N	N
LWLH[23]	2 *mul*+1 *root*	2 *exp*+2 *mul*+1 *root*	2 *exp*+5 *mul*+2 *inv*	Factorization	Y	N
Shao[20]	2 *exp*+1 *inv*	3 *exp*+1 *mul*	3 *exp*+1 *mul*	RSA	Y	Y
Yu[33]	1 *exp*+1 *mul*	3 *exp*+1 *mul*	1 *exp*+4 *mul*+1 *inv*	Factorization	Y	Y
Ours	4 *exp*+1 *mul*	1 *exp*+1 *mul*	2 *exp*+2 *mul*	Factorization	Y	Y

TABLE 2. Experimental results of the new proxy signature scheme

Setup	Delegation	ProxySign	Verification
512.658 ms	90.297 ms	25.633 ms	28.462 ms

the public key because 2 is just such an integer. As a result, the length of the public key of the new scheme is shorter than the existing constructions. Although three more exponentiations are needed in delegation generation than that of the scheme in [33], the new scheme achieves higher efficiency in proxy signature and proxy signature verification, which is highly essential for two reasons. Firstly, in many applications, delegation is a one-time process while proxy signing and verification will be executed frequently. Secondly, in some real-world scenarios, proxy signatures are often deployed in the environments such as agent-based computing systems [5], smart card applications [7], where the proxy has limited processing capability.

Experimental results: We also evaluate the performance of our new construction by conducting an experiment to assess the running time of our scheme on personal computers. An implementation of the algorithms was realized on a computer with CPU T8100 @ 2.10 GHz and 2.0 GB memory, and the operating system is Windows 7.0. All the public key operations were built with Miracle Library [44] which implements multiprecision integer and rational data-types, and provides the routines to perform basic arithmetic on them. According to the security requirements of factoring, we employed 1024-bit modulus n_0 and n_1, and the corresponding large prime factors p_0, q_0, p_1, q_1 are 512 bits. We preformed the experiment 1000 times and the average running time of each sub-algorithm is shown in Table 2. We can observe that it takes 512.658 ms to setup the system, which is the most expensive operation but runs only one time. The delegation phase costs 90.297 ms, while the proxysign and verification cost 25.633 ms and 28.462 ms respectively. The implementing results demonstrate that the new scheme is highly efficient.

5. SECURITY PROOFS

The new proxy signature is warrant-based, and a delegation is the original signer's signature on the warrant which contains information regarding proxy signer's identity, a period of validity, and restrictions on the messages for which the warrant is valid, etc. Therefore, the properties of identifiability, undeniability, verifiability and prevention of misuse can follow naturally due to the security of improved Rabin signature scheme. In this section, we will focus on the unforgeability of the new construction and demonstrate the security against adaptive chosen message attacks of Type 2 and Type 3 adversaries in the random oracle model.

Existential unforgeability against type 2 adversary

THEOREM 5.1. *If there exists a Type 2 adversary $\mathcal{A}_2$ that can $(t, q_w, q_{ps}, \epsilon)$ breaks the unforgeability of the new proxy signature scheme, then there exists another algorithm $\mathcal{B}$ that can use $\mathcal{A}_2$ to solve an instance of the factoring problem with probability*

$$Succ_{\mathcal{B}}^{IF} \geq \frac{\epsilon^2 \cdot (1 - 1/8n_0^{\frac{2}{3}})^2}{8 \cdot q_w^2}.$$

`Proof.` The security proof is by contradiction. Assume there exists a polynomial-time adversary $\mathcal{A}_2$ that is able to produce a forgery with non-negligible success probability, we then use $\mathcal{A}_2$ to build an algorithm $\mathcal{B}$ to factorize a Williams integer. Assume $\mathcal{B}$ receives a random instance of factorizing a Williams integer: given a large Williams number N with $|N| \geq$ 1024 bits, output a factor of N.

Setup: $\mathcal{B}$ sets N, the input of the factorization problem, as n_0, the public key of the original signer. Then, $\mathcal{B}$ picks two large safe primes p_1, q_1, computes $n_1 = p_1 q_1$, $\lambda(n_1) = lcm(p_1 - 1, q_1 - 1)$ and choose an element $g \in Z_{n_1}^*$ of order $\lambda(n_1)$. The hash functions $H_1 : \{0,1\}^* \rightarrow [h, h')$ with $h' - h \pmod{n_0} \geq 8n_0^{\frac{2}{3}}$, $H_2 : \{0,1\}^* \rightarrow Z_{n_1}$ and $H_3 : \{0,1\}^* \rightarrow Z_{n_1}$ are treated as random oracles. The public chameleon hash key of the proxy signer is (n_1, g) and the private trapdoor key is (p_1, q_1). $\mathcal{B}$ returns (N, p_1, q_1) to $\mathcal{A}_2$.

H_2 queries: $\mathcal{A}_2$ can make at most q_{H_2} queries on the hash function H_2 for the input (w_i, r_i) adaptively.

$\mathcal{B}$ maintains a hash list called $\mathcal{L}_2$ to store the answers for the consistency of a hash function. If $\mathcal{B}$ has received an identical query before, it responds as it did before. Otherwise, $\mathcal{B}$ picks a random value $h_i \in Z_{n_1}$, returns h_i to $\mathcal{A}_2$ and logs the tuple (w_i, r_i, h_i) into $\mathcal{L}_2$.

H_3 queries: $\mathcal{A}_2$ can make at most q_{H_3} queries on the hash function H_3 for the input (m_j, r_j) adaptively. $\mathcal{B}$ maintains a hash list called $\mathcal{L}_3$ to store the answers for the consistency of a hash function. If $\mathcal{B}$ has received an identical query before, it responds as it did before. Otherwise, $\mathcal{B}$ picks a random $h'_j \in Z_{n_1}$, returns h'_j to $\mathcal{A}_2$ and adds the tuple (m_j, r_j, h'_j) into $\mathcal{L}_3$.

H_1 queries: $\mathcal{A}_2$ can make at most q_{H_1} queries on the hash function H_1 for the input (v_i, w_i) adaptively. $\mathcal{B}$ maintains a hash list $\mathcal{L}_1$ to store the answers for the consistency of a hash function. For each request on (v_i, w_i), if $\mathcal{B}$ has received the same query before, it returns the previous answer to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ picks a random $l_i \in Z_{n_1}$, returns l_i to $\mathcal{A}_2$ and adds $(\perp, \perp, \perp, \perp, \perp, v_i, w_i, l_i)$ into $\mathcal{L}_1$ list.

Delegation queries: $\mathcal{A}_2$ can make at most q_w delegation queries adaptively. When $\mathcal{A}_2$ queries on a warrant w_i, $\mathcal{B}$ picks a random k_i and computes $r_i = g^{k_i}$. $\mathcal{B}$ then searches the $\mathcal{L}_2$ list to check whether there is an item (w_i, r_i, h_i). If there does not exist such an item, $\mathcal{B}$ picks a random $h_i \in Z_{n_1}$ and logs (w_i, r_i, h_i) into $\mathcal{L}_2$ list. Then $\mathcal{B}$ picks a random t_i, computes $r_i g^{h_i||t_i} \pmod{n_1} \rightarrow v_i$, picks $a_i, b_i \in \{0,1\}, s_i \in B_{N,h,h'}$ with uniform distribution, sets $H(v_i, w_i) = s_i^2 \cdot 2^{a_i} \cdot (-1)^{-b_i} \pmod{N}$ and logs $(a_i, b_i, k_i, r_i, t_i, v_i, w_i, l_i)$ into $\mathcal{L}_1$ list. Finally, $\mathcal{B}$ responds (t_i, s_i, a_i, b_i) to $\mathcal{A}_2$.

Proxy signature queries: $\mathcal{A}_2$ can make at most q_{ps} proxy signature queries. For each query on (w_i, m_i), $\mathcal{B}$ searches the $\mathcal{L}_1$ list to check whether the tuple $(a_i, b_i, k_i, r_i, t_i, v_i, w_i, s_i)$ exists in the list. If there is no such a tuple, $\mathcal{B}$ picks a random k_i and computes $r_i = g^{k_i}$. Then $\mathcal{B}$ picks a random t_i, computes $r_i g^{h_i||t_i} \pmod{n_1} \rightarrow v_i$, picks $a_i, b_i \in \{0,1\}, s_i \in B_{N,h,h'}$ with uniform distribution, sets $H(v_i, w_i) = s_i^2 \cdot 2^{a_i} \cdot (-1)^{-b_i} \pmod{N}$ and logs $(a_i, b_i, k_i, r_i, t_i, v_i, w_i, s_i)$ into $\mathcal{L}_1$ list. $\mathcal{B}$ searches the $\mathcal{L}_2$ list to find if there is an item (w_i, r_i, h_i). If there is no such an item, $\mathcal{B}$ picks a random value $h_i \in Z_{n_1}$, and logs the tuple (w_i, r_i, h_i) into $\mathcal{L}_2$. Then, $\mathcal{B}$ chooses a random $k_{pi} \in Z_{\lambda(n_1)}$, computes $r_{pi} \equiv g^{k_{pi}} \pmod{n_1}$. $\mathcal{B}$ searches if there exists an item (m_i, r_{pi}, h'_i) in $\mathcal{L}_3$ list. If there is no such an item, $\mathcal{B}$ selects a random $h'_j \in Z_{n_1}$, and adds the tuple (m_i, r_{pi}, h'_i) into $\mathcal{L}_3$. Finally, $\mathcal{B}$ computes $t_{pi} \equiv t_i + (k_i - k_{pi}) + 2^{\lambda(n_1)}(h_i - h'_i) \bmod \lambda(n_1)$ and response $(a_i, b_i, s_i, t_i, r_i, r_{pi}, t_{pi})$ as a proxy signature of m_i under the warrant w_i to $\mathcal{A}_2$.

Solving IF problem: It's easy to check that in the random oracle model, the outputs of the simulated proof are indistinguishable from those in the real attacks. Finally, $\mathcal{A}_2$ produces a proxy signature forgery $\sigma^* = (a_0^*, b_0^*, s_0^*, t_0^*, r_1^*, r_2^*, t_1^*)$ on a message m^* under warrant w^* with probability ϵ. $\mathcal{A}_2$ outputs a valid proxy signature after querying H_1 oracle with probability $\epsilon' = \epsilon \cdot (1 - 1/8n_0^{\frac{2}{3}})$ because s_i is randomly picked from $[h, h')$. We can solve the factorization problem instance by using the oracle replay technique[32]. We firstly guess a fixed index $1 \leq j \leq q_w$ and expect that w^* happens to be the warrant of which $\mathcal{A}_2$ forges a valid proxy signature on m^*. The probability that we make a correct guess is denoted by $\Pr[\text{luck}] = \frac{1}{q_w}$.

Equipped with the same system parameters and public key N, n_1, the game is played again. For the same sequences of random bits to the two copies of $\mathcal{A}_2$, $\mathcal{B}$ responds with the same answers to the oracle queries until $\mathcal{A}_2$ asks the H_1 query for $w_j = w^*$. At that point, $\mathcal{B}$ answers two random answers $(a_i, b_i, k_i, r_i, t_i, v_i, w_i, s_i)$ and $(a'_i, b'_i, k'_i, r'_i, t'_i, v'_i, w'_i, s'_i)$. Thus, we have

$$s_i^2 \cdot 2^{a_i} \cdot (-1)^{-b_i} \equiv s_i'^2 \cdot 2^{a'_i} \cdot (-1)^{-b'_i} \pmod{N}.$$

Since a_i, b_i are randomly chosen from $\{0,1\}$, $a_i = a'_i$ and $b_i = b'_i$ with probability $1/4$. As a result, $s_i^2 \equiv s_i'^2 \pmod{N}$ holds with probability $1/4$. Now $\text{GCD}(s_i - s'_i, N)$ or $\text{GCD}(s_i + s'_i, N)$ is a nontrivial factor of N [29].

The simulator's probability of success can be assessed using the technique of splitting lemma[32]. Let X be the set of possible sequences of random answers up to the point that $\mathcal{A}_2$ queries the H_1 oracle for (v^*, w^*) and Y be the set of possible sequences of random answers after that point. For any $x \in X$ and $y \in Y$, given the sequences of random answers of random oracles $(x||y)$, the probability that $\mathcal{B}$ obtains a valid proxy signature is $\epsilon \cdot (1 - 1/8n_0^{\frac{2}{3}})/q_w$. Following the splitting lemma, there exists such a "good" subset $\Omega \in X$ such that the probability that the probability that $x \in \Omega$ is at least $\epsilon \cdot (1 - 1/8n_0^{\frac{2}{3}})/2q_w$, and for any $s \in \Omega$, $y \in Y$, the probability that $\mathcal{B}$ gets a valid proxy signature is at least $\epsilon \cdot (1 - 1/8n_0^{\frac{2}{3}})/2q_w$ with the sequences of random answers $(s||y)$. Therefore, the probability that $\mathcal{B}$ can solve the factorization problem is

$$\begin{aligned} Succ_{\mathcal{B}}^{IF} &\geq \frac{1}{4} \cdot \epsilon \cdot (1 - 1/8n_0^{\frac{2}{3}})/q_w \cdot \epsilon \cdot (1 - 1/8n_0^{\frac{2}{3}})/2q_w \\ &= \frac{\epsilon^2 \cdot (1 - 1/8n_0^{\frac{2}{3}})^2}{8 \cdot q_w^2}. \end{aligned}$$

◇

Existential unforgeability against Type 3 Adversary

THEOREM 5.2. *If there exists a type 3 adversary $\mathcal{A}_3$ that can (t, q_{ps}, ϵ) break the proxy signature scheme, there exists another algorithm $\mathcal{B}$ that can use $\mathcal{A}_3$ to solve an instance of factoring problem with probability*

$$Succ_{\mathcal{B}}^{IF} \geq \frac{1}{4q_{H_1}^2} \cdot \epsilon^3 \cdot (1 - \frac{1}{n})^3.$$

`Proof.` The adversary $\mathcal{A}_3$ possesses the public keys of the original signer and the proxy signer and the private key of the original signer, thus, $\mathcal{A}_3$ can generate delegations on any warrant. As a result, delegation queries are not required here. Assume $\mathcal{B}$ is given a large composite number N with $|N| \geq 1024$ bits, $\mathcal{B}$'s goal is factoring N. $\mathcal{B}$ will simulate the challenger and response $\mathcal{A}_3$'s queries in the following way.

Setup: $\mathcal{B}$ sets N, the input of the factorization problem, as n_1, the public key of the proxy signer, retaining $H_1 : \{0,1\}^* \rightarrow [h, h')$ with $h' - h \pmod{n_0} \geq 8n_0^{\frac{2}{3}}$, $H_2 : \{0,1\}^* \rightarrow Z_{n_1}$ and $H_3 : \{0,1\}^* \rightarrow Z_{n_1}$ for use as random oracles. Then, $\mathcal{B}$ picks two large primes p_0, q_0 as the original signer's secret key and the corresponding public key is $n_0 = p_0 q_0$. $\mathcal{B}$ delivers (N, p_0, q_0, n_0) to $\mathcal{A}_3$.

H_2 queries: $\mathcal{A}_3$ can make at most q_{H_2} queries on the hash function H_2 for the input (w_i, r_i') adaptively. $\mathcal{B}$ maintains a hash list called $\mathcal{L}_2$ to store the answers for the consistency of a hash function. If $\mathcal{B}$ has received an identical query before, it responds as it did before. Otherwise, $\mathcal{B}$ picks a random value $h_i \in Z_{n_1}$, returns h_i to $\mathcal{A}_2$ and logs the tuple (w_i, r_i', h_i) into $\mathcal{L}_2$.

H_3 queries: $\mathcal{A}_3$ can make at most q_{H_3} queries on the hash function H_3 for the input (m_j, r_j) adaptively. $\mathcal{B}$ maintains a hash list called $\mathcal{L}_3$ to store the answers for the consistency of a hash function. If $\mathcal{B}$ has received an identical query before, it responds as it did before. Otherwise, $\mathcal{B}$ picks a random $h_j' \in Z_{n_1}$, returns h_j' to $\mathcal{A}_3$ and adds the tuple (m_j, r_j, h_j') into $\mathcal{L}_3$.

H_1 queries: $\mathcal{A}_3$ can make at most q_{H_1} queries on the hash function H_1 for the input (v_i, w_i) adaptively. $\mathcal{B}$ maintains a hash list $\mathcal{L}_1$ to store the answers for the consistency of a hash function. For each request on (v_i, w_i), if $\mathcal{B}$ has received the same query before, it returns the previous answer to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ picks a random $l_i \in Z_{n_1}$, returns h_i to $\mathcal{A}_2$ and logs $(\perp, \perp, \perp, v_i, w_i, l_i)$ into $\mathcal{L}_1$ list.

Proxy signature queries: $\mathcal{A}_3$ can make at most q_{ps} proxy signature queries. For each proxy signature query on (w_i, m_i), $\mathcal{B}$ firstly generates a delegation (a_i, b_i, s_i, t_i) for w_i since he has original signer's secret key (p_0, q_0). $\mathcal{B}$ searches the $\mathcal{L}_2$ list to check if there exists an item (w_i, r_i', h_i). If there is no such an item, $\mathcal{B}$ picks $r_i', h_i \in Z_N$ randomly and logs (w_i, r_i', h_i) into $\mathcal{L}_2$ list. Then, $\mathcal{B}$ picks a random $r_i \in Z_{n_1}$ and checks whether there exists an item (m_i, r_i, h_i') in $\mathcal{L}_3$ list. If there is no such an item, $\mathcal{B}$ picks a random $h_i' \in Z_{n_1}$, and adds the tuple (m_i, r_i, h_i') into $\mathcal{L}_3$. $\mathcal{B}$ selects a random t_i', computes $v_i = r_i g^{h_i' || t_i'}$ and sets $H_1(v_i, w_i) \rightarrow s_i^2 \cdot 2^{a_i} \cdot (-1)^{-b_i}$. Finally, $\mathcal{B}$ logs $(\perp, \perp, \perp, v_i, w_i, s_i^2 \cdot 2^{a_i} \cdot (-1)^{-b_i})$ into $\mathcal{L}_1$ list and transmits $(a_i, b_i, s_i, r_i, r_i', t_i, t_i')$ to $\mathcal{A}_3$ as the answer to his proxy signature query.

Solving IF problem: The simulated outputs are indistinguishable from those in the real attacks in the random oracle model. Therefore, after a series of queries, $\mathcal{A}_3$ will output a new proxy signature $\sigma = (a_0^*, b_0^*, s_0^*, t_0^*, r_1^*, r_2^*, t_1^*)$ on the message m^* under the warrant w^* with probability ϵ. If $\mathcal{A}_3$ has not queried $H_3(m^*, r_2^*)$, the probability that σ is valid is less than $1/n_1$ since the answers to the H_3 hash functions are picked randomly from Z_{n_1}. Thus, $\mathcal{A}_3$ will produce a valid forgery after querying the H_3 oracle with probability $\epsilon' = 1 - 1/n_1$ and the answer to $H_3(m^*, r_2^*)$ is h'. Then, based on the well-known oracle replay attack and the forking lemma [32], $\mathcal{A}_3$ uses the oracle replay technique by a polynomial replay of the attack with the same random salts and a different oracle called H_3' to get another forgery $\sigma' = (a_0', b_0', s_0', t_0', r_1', r_2^*, t_1')$ on the message m^* under the warrant w^*, and the answer to $H_3'(m^*, r_2^*)$ is h''. Now, we have

$$r_2^* g^{h' || t_1^*} \equiv r_2^* g^{h'' || t_1'} \pmod{n_1}.$$

Thus, $t_1^* + 2^k h' \equiv t_1' + 2^k h'' \bmod (\lambda(n_1))$ where $k = |\lambda(n_1)|$ denotes the bit length of $\lambda(n_1)$. Therefore, $\lambda(n_1)$ divides $t_1^* - t_1' + 2^k(h' - h'')$, which indicates that $\phi(n_1)$ divides $2(t_1^* - t_1' + 2^k(h' - h''))$. Following the conclusion due to Miller [43], knowing such a multiple of $\phi(n_1)$ is equivalent to factoring n_1, which contracts the factoring assumption. The probability that $\mathcal{B}$ can solve the factorization problem instance is

$$Succ_{\mathcal{B}}^{IF} \geq \frac{1}{4q_{H_1}^2} \cdot \epsilon^3 \cdot (1 - \frac{1}{n})^3.$$

$\diamond$

6. CONCLUSION

In this paper, we put forward a novel construction of proxy signature whose unforgeability can be reduced to the integer factorization problem. The new scheme employed improved Rabin-type digital signature and a factorization based key-exposure free chameleon hash function. We proved that the scheme is secure against all adversaries, in the random oracle model. Our scheme compares favorably with existing schemes based on the

RSA assumption or the factorization assumption. How to construct a proxy signature scheme with tighter reductions to factorization without using forking lemma is our future work.

Acknowledgement: The first author is supported by the University of Wollongong Vice Chancellor Fellowship and the fourth author is supported by Australian Research Council Future Fellowship FT0991397. This work is supported by the NSFC of China under Grants 61003232, 61103207, U1233108, the National Research Foundation for the Doctoral Program of Higher Education of China under Grant 20100185120012, the NSFC of China for International Young Scientists under Grant 61250110543, and the Fundamental Research Funds for the Central Universities under Grants ZYGX2010J066 and ZYGX2011J067.

REFERENCES

[1] Mambo, M., Usuda, K., Okamoto, E. (1996) Proxy signature: delegation of the power to sign messages. IEICE Trans. Fundamentals, E79-A (9), 1338–1353.

[2] Tan, Z. (2011) An off-line electronic cash scheme based on proxy blind signature. The Computer Journal, 54 (4), 505–512.

[3] Yu, Y., Mu, Y., Wang, G., Sun, Y. (2011) Cryptanalysis of an off-line electronic cash scheme based on proxy blind signature. The Computer Journal, 54 (10), 1645–1651.

[4] Zhang, J., Liu, C., Yang, Y. (2010) An efficient secure proxy verifiably encrypted signature scheme, Journal of Network and Computer Applications, 33(1), 29–34.

[5] Lee, B., Kim, H., Kim, K. (2011) Secure mobile agent using strong non-designated proxy signature. In Mu, Y., Varadharajan, Vijay.(eds), Proc. of ACISP 2001, Sydney, Australia, July 11–13, Lecture Note in Computer Science 2119, pp. 474–486. Springer, Berlin.

[6] Lee, B., Kim, H., Kim, K. (2001) Strong proxy signature and its applications, in: The 2001 Symposium on Cryptography and Information Security Oiso, Japan, January 23-26, 603–08.

[7] Okamoto, T., Tada, M., Okamoto, E. (1999) Extended proxy signatures for smart cards. In: Mambo, M., Zheng Y. (eds.), Information Security, Second International Workshop, ISW'99, Kuala Lumpur, Malaysia, Nov 1999, Lecture Note in Computer Science 1729, pp. 247–258. Springer, Berlin.

[8] Boldyreva, A., Palacio, A., Warinschi, B. (2012) Secure proxy signature scheme for delegation of signing rights. Journal of Cryptology. 25(1), 57–115.

[9] Lee, J. Y., Cheon, J. H., Kim, S. (2003) An analysis of proxy signatures: is a secure channel necessary? In Joye, M. (eds.), Topics in Cryptology - CT-RSA 2003, San Francisco, CA, USA, April 13-17, 2003, Lecture Note in Computer Science 2612, pp. 68–79. Springer, Berlin.

[10] Kim, S., Park, S., Won, D. (1997) Proxy signatures, revisited. In Han, Y., Okamoto, T., Qing, S. (eds.), Information and Communication Security, First International Conference, ICICS'97, Beijing, China, Nov 11–14, 1997. Lecture Note in Computer Science 1334, pp. 223–232. Springer, Berlin.

[11] Wang, G., Bao, F., Zhou, J., Deng, R. H. (2003) Security analysis of some proxy signatures. In Qing, S., Gollmann, D., Zhou J. (eds.), Information and Communications Security, 5th International Conference, ICICS 2003, Huhehaote, China, Oct 10-13, 2003. Lecture Note in Computer Science 2971, pp. 305–319. Springer, Berlin.

[12] Shao,Z. (2009) Improvement of identity-based proxy multi-signature scheme. Journal of Systems and Software. 82(5), 794–800.

[13] Sun, Y. , Xu, C., Yu, Y., Mu, Y. (2011) Strongly unforgeable proxy signature scheme secure in the standard model. Journal of Systems and Software. 84(9), 1471–1479.

[14] Huang, X., Mu, Y., Susilo, W., Zhang, F. (2005) A short proxy signature scheme: efficient authentication in the ubiquitous world. In Callaghan, V., Enokido, T., Jin, H. (eds), Proc. of UISW2005, Nagasaki, Japan, Dec 6–7, 2005. Lecture Note in Computer Science 3823, pp. 480–489. Springer, Berlin.

[15] Zhang, F., Naini, R., Lin, C. (2004) Some new proxy signature schemes from bilinear pairings. In Progress on Cryptography: 25 years of Cryptography in China, Kluwer International Series in Engineering and Computer Science, 769, 59–66.

[16] Lu, R., Cao, Z. (2005) Designated verifier proxy signature scheme with message recovery. Applied Mathematics and Computation. 169(2), 1237–1246.

[17] Yu, Y., Xu, C., Zhang, X., Liao, Y. (2009) Designated verifier proxy signature scheme without random oracles, Computers and Mathematics with Applications. 57(8), 1352–1364.

[18] Schdult, J., Matsuura, K., Paterson, K. (2008) Proxy signatures secure against proxy key exposure. In Cramer R. (ed.), Proc. of PKC 2008, Barcelona, Spain, March 9–12. Lecture Note in Computer Science 4939, pp. 344–359. Springer, Berlin.

[19] Shao, Z. (2003) Proxy siganture schemes based on factoring. Information Processing Letters, 85, 137–143.

[20] Shao, Z. (2009) Provably secure proxy-protected signature schemes based on RSA. Computers and Electrical Engineering. 35(3), 497–505.

[21] Zhou, Y., Cao, Z., Lu, R. (2005) Provably secure proxy-protected signature schemes based on factoring. Applied Mathematics and Computations. 164, 83–98.

[22] Park,J. H. B., Kang, G., Han, J. W. (2005) Cryptanalysis of Zhou et al.'s proxy-protected signature schemes, Applied Mathematics and Computations. 169, 192–197.

[23] Liu, Y. C., Wen, H. A., Lin,C. L., Hwang, T. (2007) Proxy-protected signature secure against the undelegated proxy signature attack. Computers and Electrical Engineering. 33, 177–185.

[24] Hu, X., Xu, H., Si, T. (2010) Analysis and improvement of proxy-protected signature secure against the undelegated proxy signature attack. Journal of Computational Information System. 6(9), 2997–3002.

[25] Gentry, C. (2004) How to compress Rabin ciphertexts and signatures. In. Franklin M. K.(ed.), Advances in Cryptology - CRYPTO 2004, Santa Barbara, California, USA, August 15–19, 2004, Lecture Note in Computer Science 3152, pp. 179–200. Springer, Berlin.

[26] Schnorr, C. P. (1989) Efficient identification and signatures for smart cards. In Brassard G. (Ed.), Proc. of CRYPTO 1989, Santa Barbara, California, USA, August 20-24, 1989, 239-252

[27] Rabin, M. O. (1979) Digitalized signatures and public key functions as intractable as factorization, MIT/LCS/TR-212, MIT Laboratory for computer Science, 1979.

[28] Hoffstein, J., Pipher, J., Silverman, J. H. (2008) An introduction to mathematical cryptography, Springer Science+Business Media.

[29] Shoup, V. (2008) A computational introduction to number theory and algebra, Cambridge University Press.

[30] B, Valle'e (1998) Provably fast integer factoring with quasi-uniform small quadratic residues. in: Proc. of STOC 1989, 98–106.

[31] Coron,J. S. (2002) Security proof for partial-domain hash signature schemes. In. Yung M. (ed.), Proc. of CRYPTO 2002, Santa Barbara, California, USA, August 18-22, 2002, Lecture Notes in Computer Science 2442, pp. 613–626. Springer, Berlin.

[32] Pointcheval, D., Stern, J. (2000) Security arguments for digital signatures and blind signatures. Journal of Cryptology. 13(3), 361–396.

[33] Yu, Y., Mu, Y., Susilo, W., Sun, Y., Ji, Y. (2012) Provably secure proxy siganture scheme from factorization, Mathematical and Computer Modelling. 55, 1160–1168.

[34] Krawczyk, H., Rabin, T. (2000) Chameleon hashing and signatures. Network and distributed system security symposium, 143–54.

[35] Shamir, A., Tauman, Y. (2001) Improved On-line/Off-line Signature Schemes. In Kilian J. (ed.), Proc. of CRYPTO 2001, Santa Barbara, California, USA, August 19–23, 2001, Lecture Notes in Computer Science 2139, pp. 355–367. Springer, Berlin.

[36] Harn, L., Hsin, W., Lin, C. (2010) Efficient On-line/Off-line Signature Schemes Based on Multiple-Collision Trapdoor Hash Families. Comput. J., 53(9), 1478–1484.

[37] Ateniese, G., Medeiros, B. (2004) Identity-based chameleon hash and applications. In Juels A. (Ed.), Proc. of Financial Cryptography 2004, Key West, FL, USA, February 9–12, 2004 Lecture Notes in Computer Science 3110, pp. 164–180. Springer, Berlin.

[38] Ateniese, G., Medeiros, B. (2004) On the key exposure problem in chameleon hashes. In Blundo, C., Cimato S. (eds.), Proc. of SCN 2004, Amalfi, Italy, Sep 8–10, 2004, Lecture Notes in Computer Science 3352, pp. 165–179. Springer, Berlin.

[39] Chen, X., Zhang, F., Tian H. et al.(2008), Efficient generic on-line/off-line (threshold) signatures without key exposure, Information Sciences. 178 (21), 4192–4203.

[40] Chen, X., Wu, Q., Zhang, F., Tian H. et al. (2011) New receipt-free voting scheme using double-trapdoor commitment, Information Sciences. 181(8), 1493–1502.

[41] Mehta, M., Harn, L.(2005) Efficient one-time proxy signatures, IEE Proc.-Commun. 152 (2), 129–133.

[42] Chandrasekhar,S., Chakrabarti, S., Singhal, M., Calvert, K. L. (2010) Efficient proxy signatures based on trapdoor hash functions. IET Information Security. 4(4). 322–332.

[43] Miller, G. (1976) Reimann's hypothesis and tests for primality. Journal of Computer System Science. 13, 300–317.

[44] Shamus Software Ltd., Miracle library. Available from: http://www.shamus.ie/index.php?page=home. (accessed April 10, 2010)