Security Mediated Certificateless Signatures Without Pairing

Zhongmei Wan
College of Science, Hohai University, Nanjing, 210098, P.R. China
zmeiwan@gmail.com

Jian Weng
Department of Computer Science, Jinan University, Guangzhou 510632, P.R. China
cryptjweng@gmail.com

Jiguo Li
College of Computer and Information Engineering, Hohai University, Nanjing, 210098, P.R. China
lijiguo@hhu.edu.cn

***Abstract*—Security-mediated certificateless (SMC) signature provides a method for immediate revocation of security capabilities in certificateless signature (CLS). In addition, SMC signauture maintains the merits in CLS: implicit certification without key escrow. Unfortunately, most of the existing schemes for CLS and SMC signature are based on bilinear pairing. There are schemes without bilinear pairing such as [1] and [2] which is based on discrete logarithm problem. In this paper, based on the schemes in [2] and [3], we propose another SMC signature scheme without bilinear pairing. Our scheme is existential unforgeable in the random oracle model based on the intractability of the factoring problem. In addition, it is also efficient in signing and verifying.**

***Index Terms*—security mediated, SMC cryptography, ID-based cryptography, certificateless signature, factoring**

I. INTRODUCTION

In 1984, Shamir [4] introduced the concept of ID-based cryptography (IBC) where any strings such as email addresses, server names or phone numbers, can be used as public keys. This simplifies certificate management procedures of public key infrastructure (PKI) in traditional public key cryptography. However, an inherent problem of ID-based cryptography is the key escrow problem. To handle this problem, Al-Riyami and Paterson [5] introduced the concept of certificateless public key cryptography (CLPKC) in which the Key Generation Center (KGC) generates a user's partial private key from the master secret key. The user also independently generates a secret value and the corresponding public key, and a user's full private key is a combination of his partial private key and his secret value, in such a way that the key escrow problem can be security capabilities in CLPKC, Chow, Boyd and González Neito [6] introduced in PKC 2006 the notion of SMC cryptography which maintains the merits in CLPKC: implicit certification without key escrow. The notion of SMC cryptography is a generalization of the notion of plain CLPKC.

Since Al-Riyami and Paterson [5] presented the first concrete construction of the certificateless signature (CLS) scheme, a lot of CLS schemes [7-15] based on bilinear pairing [16] have been proposed. Recently, there are CLS schemes without pairing such as [1] and [2] which are pretty efficient. The schemes of [1] and [2] are based on the intractability of the discrete logarithm problem. To avoid putting all our eggs in the same basket, it is common practice in cryptography to try to find alternative constructions of a primitive based on different intractability assumptions. Therefore it is desirable to design a proven secure SMC signature scheme based on factoring.

Our contribution. Based on the schemes in [2] and [3], we present another provable secure SMC signature scheme without bilinear pairing. It is existential unforgeable in the random oracle model. In addition, our scheme is very efficient computationally. Since signing requires only four exponentiations in Z_N^* and verifying requires only three exponentiations in Z_N^*. To the best of our knowledge, this is the first SMC signature scheme based on the intractability of the factoring problem.

The rest of the paper is organized as follows. In section II, we give a brief review of some basic concepts and theorems used in our scheme. In section III we review the formal definition and security model for SMC signature. In section IV, we present our concrete scheme. In section V, we describe the security proof. A conclusion is drawn in section VI.

This paper was partially supported by the National Natural Science Foundation of China (No. 60903178), the Fundamental Research Funds for the Central Universities (No.2010B09614) and the Foundation of HoHai University (2084/409265).

doi:10.4304/jcp.5.12.1862-1869

II. PRELIMINARIES

A. Quadratic Residue

For positive integer N, an integer a is called a quadratic residue modulo N if GCD(a,N) = 1 and $x^2=a$ mod N for some integer a, where x is called a square root of a modulo N.

Let $N = p\,q$ be a composite modulus, where p and q are two large prime numbers, satisfying p=2p′+1, q=2q′+1, with themselves primes. Let N denote the subgroup of squares in Z_N^*. Then, it is well known that Q_N is a cyclic group with order $\varphi(N)/4 = (p-1)(q-1)/4$ [17].

Remark: Following the theorems introduced in [3], a 2^lth root s of a could be efficiently computed as $s = a^{d^l}$, where d^l is computed over modulo p′q′. Note, factorization of N must be known, otherwise it is computationally infeasible to calculate the 2^lth root.

B. the Hardness Assumption of Factoring

Informally, the factoring problem is stated as follows: given a k-bit composite N, which is a multiple of two large primes p and q, to output p or q. The factoring problem is always assumed to be (t, ε)-hard, in the sense that there is no algorithm that can output p or q with probability over ε in polynomial time t (with respect to some security parameter k).

III. FRAMEWORK of SECURITY MEDIATED CERTIFICATELESS SIGNATURE

In this section we review the formal definition and security model for security mediated certificateless signature [2].

A. Synatax

Definition 1. A security mediated certificateless signature scheme consists of five tuples of polynomial time algorithms as follows:

1. **Setup:** is a probabilistic polynomial time (PPT) algorithm, run by a Key Generation Center (KGC), given a security parameter 1^k as input, outputs a randomly chosen master key s and a list of public parameters **params**.
2. **KeyGen:** is a PPT algorithm, run by the user, given a list of public parameters **params** as input, outputs a private key sk_{ID} and a public key PK_{ID}.
3. **Register:** is a PPT algorithm that takes as input a list of public parameters **params**, the master key s, a user identity ID and a public key PK_{ID}. It returns the SEM private signing key S_{ID}.
4. **Signing:** is an interactive probabilistic protocol between the user and the SEM. Their common inputs include a list of public parameters **params**, a message m, and a user identity ID. The SEM has an additional input of S_{ID} to run the sub-algorithm SEM-Sign; while the user has an additional input of sk_{ID} to run the sub-algorithm User-Sign. The protocol finishes with either a signature σ or ⊥ when the SEM refuses to give a valid partial signature, for example in the case where the user's signing capability has been revoked.
5. **Verifying:** is a deterministic algorithm that takes as input a list of public parameters **params**, a message m, a user identity ID, a user public key PK_{ID} and a signature σ. It returns true or false.

B. Security Model

As introduced in [5], we discuss the security notions for a SMC signature scheme according to two kinds of adversaries: a type I adversary A_I and a type II adversary A_{II} . A_I acts as a dishonest user. A_I does not have access to the master key but can replace any users public keys at will. A_{II} acts as a malicious KGC. A_{II} has access to the master key but cannot replace any user's public key. We introduce the model in [2] which is extended based on [5].

Definition 2 A SMC signature scheme is secure against the existential forgery on adaptive chosen message and identity attacks (EUF-CMIA) against adversary A=(A_I, A_{II}) if no polynomial time algorithm A has a non-negligible advantage against a challenger C in the following game:

1. **Setup:** C takes as input 1^k, runs the **Setup** algorithm, and gives A the resulting **params**. The master key is given to A if it is a Type II adversary.

   A makes the following requests or queries adaptively:
2. **SEM-Key Extraction Queries:** On input an identity ID, the adversary is returned with S_{ID} held by SEM for doing the partial signing on behalf of the user. (Note that it is only useful to Type I adversary)
3. **Public Key Extraction Queries:** On input an identity ID, the adversary obtains user ID's public key PK_{ID}.
4. **Private Key Extraction Queries:** On input an identity ID, the adversary gets the user's private key sk_{ID}. This query is reasonably disallowed if the public key of user ID has already been replaced by the adversary. (In the case of Type I adversary).
5. **Public Key Replacement Queries:** (For Type I adversary only) On input an identity ID and a valid public key, the public key of user ID is replaced by this new one.
6. **SEM-Sign Queries:** On input a message m and an identity ID, the adversary is returned with the partial signing result by using S_{ID}.
7. **User-Sign Queries:** On input a message m and a public key PK_{ID}, the adversary is returned with the partial signing result by using sk_{ID}, even if the public key PK_{ID} is previously replaced by the (Type I) adversary.
8. **Complete-Sign Queries:** On input a message m and a public key PK_{ID}, the adversary is returned with the complete signing result by using sk_{ID} and S_{ID}, even if the user public key PK_{ID} is previously replaced by the (Type I) adversary.

9. **Forgery:** *A* outputs (m^*, ID^*, PK_{ID^*}, σ^*) such that the following conditions hold:

- (ID^*, m^*) was never appeared in the **Complete-Sign** queries.
- **Verifying** (**params**, $m^*, ID^*, PK_{ID^*}, S^*$)=1.
- If it is Type I, ID^* has not been submitted to **SEM-Key Extraction Queries**. Moreover, (ID^*, m^*) has not been submitted to both **SEM-Sign Queries** and **Complete-Sign Queries**.
- If it is Type II, ID^* has not been submitted to **Private Key Extraction Queries**. Moreover, (ID^*, m^*) has not been submitted to both **User-Sign Queries** and **Complete-Sign Queries**.

We define *A*'s advantage as the probability of winning this game.

IV. CONSTRUCTION of OUR SCHEME

In this section, we propose a security mediated certificateless signature scheme which is built on quadratic residue. The merit of this approach is that no pairing computation is needed at all. Our scheme is motivated from those in [2] and [3].

1. **Setup:** Given the security parameter (k, l), the KGC performs the following.

- Generate two random primes p_1, q_1, such that $p_1 = 2p_1' + 1$, $q_1 = 2q_1' + 1$, with p_1', q_1' being primes too, satisfying $2^{k-1} \le (p_1-1)(q_1-1)$ and $p_1q_1 < 2^k$, then compute $N_1 = p_1q_1$.
- Select $a \in_R Z_{N_1}^*$, such that Jacobi symbol $(a/N_1) = -1$.
- Compute $d=(N_1-p_1-q_1+5)/8$.
- Select four one-way hash function h_0, h_1, h_2, h_3, satisfying $h_1 : \{0,1\}^* \to Z_{N_1}^*$ and h_0, h_2, h_3: {0, 1}*→ {0, 1}^{l}.

  The master key of KGC is set to be $msk = (p_1, q_1, p_1', q_1', d)$ and the public parameter of KGC are **params**=(N_1, $h_0()$, $h_1()$, $h_2()$, $h_3()$, a, l).

2. **KeyGen:** The user performs the following.

- Generate two random primes p_2, q_2, such that $p_2 = 2p_2' + 1$, $q_2 = 2q_2' + 1$ witt p_2', q_2' being primes too, satisfying $2^{k-1} \le (p_2-1)(q_2-1)$ and $p_2q_2 < 2^k$, then compute $N_2 = p_2\, q_2$.
- Randomly select $x_{ID} \in Z_{N_2}^*$, set $sk_{ID} = (p_2, q_2, p_2', q_2', x_{ID})$ as the user private key.
- Compute $P_{ID} = x_{ID}^{2^l} \bmod N_2$, set $PK_{ID}=(N_2, P_{ID})$ as the user public key.

3. **Register:** Now the user *ID* wants to register a public key, the KGC performs the following:

- Authenticate and register (ID, PK_{ID}).
- Compute $c_1 = \begin{cases} 0 & (h_1(ID)/N_1) = 1 \\ 1 & (h_1(ID)/N_1) = -1 \end{cases}$.
- Compute $h = a^{c_1} h_1(ID)$ and $c_2 = \begin{cases} 0 & (h/p_1) = (h/q_1) = 1 \\ 1 & (h/p_1) = (h/q_1) = -1 \end{cases}$.

Let $H(ID) = (-1)^{c_2} a^{c_1} h_1(ID)$, then $H(ID) \in Q_{N_1}$.

- Compute S_{ID} as a 2^lth root of $H(ID)$, $S_{ID} = H(ID)^{d^l} \bmod N_1$.

  Note $S_{ID}^{2^l} = H(ID) \bmod N_1$
- Send the SEM private key S_{ID}, two tags (c_1, c_2) and the (ID, PK_{ID}) to the SEM over a confidential and authentic channel.

4. **Signing:** Suppose the user *ID* wants to get the signature of message m, the SEM checks whether *ID* is revoked; if not, the interaction between the signer and the SEM is as follows:

- **SEM-Sign(I):** Chooses $r_s \in_R Z_{N_1}^*$, computes $R_s = r_s^{2^l} \bmod N_1$, sends $c_s=h_0(R_s)$ to the user.
- **User-Sign(I):** Chooses $r_u \in_R Z_{N_2}^*$, computes $R_u = r_u^{2^l} \bmod N_2$, sends R_u to the SEM.
- **SEM-Sign(II):** Computes R mod N_1N_2 by using the Chinese Remainder Theorem such that $R=R_s$ mod N_1, $R=R_u$ mod N_2, then computes $h_s=h_2(ID, PK_{ID}, R, m\|0)$, $t_s = r_s S_{ID}^{h_s} \bmod N_1$ and sends back (R_s, t_s) to the user.
- **User-Sign(II):** Checks if $c_s=h_0(R_s)$; if they are equal, computes R mod N_1N_2 by using the Chinese Remainder Theorem such that $R=R_s$ mod N_1, $R=R_u$ mod N_2, then computes $h_u=h_3(ID, PK_{ID}, R, m\|1)$, $t_u = r_u x_{ID}^{h_u} \bmod N_2$, finally computes t mod N_1N_2 by using the Chinese Remainder Theorem such that $t=t_s$ mod N_1, $t=t_u$ mod N_2.

  The final signature is σ=(R, t, c_1, c_2).

5. **Verifying:** Given a signature $\sigma=(R, t, c_1, c_2)$ on a message m, any party could verify the validity by the following operations:

- Compute $H(ID) = (-1)^{c_2} a^{c_1} h_1(ID)$, $h_s=h_2(ID, PK_{ID}, R, m\|0)$, $h_u=h_3(ID, PK_{ID}, R, m\|1)$.
- Compute $R_s' = t^{2^l} H(ID)^{-h_s} \bmod N_1$, and $R_u' = t^{2^l} H(ID)^{-h_u} \bmod N_2$, compute R' mod N_1N_2 by using the Chinese Remainder Theorem such that $R' = R_s' \bmod N_1$, $R' = R_u' \bmod N_2$.
- Check whether $R=R'$ mod N_1N_2 holds or not. If the equation holds, output "valid'', otherwise, output "invalid".

The correctness of the proposed scheme can be easily verified as follows.

$$R' = R'_s = t^{2^l} H(ID)^{-h_s} = t_s^{2^l} S_{ID}^{-2^l h_s} = r_s^{2^l} = R_s = R \bmod N_1$$

$$R' = R'_u = t^{2^l} H(ID)^{-h_u} = t_u^{2^l} S_{ID}^{-2^l h_u} = r_u^{2^l} = R_u = R \bmod N_2$$

V. SECURITY ANALYSIS

Theorem 1. If the factoring problem is (t', ε')-hard, then our scheme is (t, ε)-existential unforgeable against Type I adversary,satisfying

$$\varepsilon' \geq \frac{1}{2q_{h_1}}\left(\frac{\Delta^2}{q_{h_2}+1} - 2^{-l}\Delta\right),\ t' = t + O(k^2 l + k^3),$$

where

$$\Delta = \varepsilon - q_s(q_{h_0}+q_s)2^{3-k} - q_s(q_{h_2}+q_s)2^{6-2k} - (q_s + q_{h_0}+1)^2 2^{6-2k} - q_s / 2^l,$$

(k, l) are security parameters introduced in section 4, q_s denotes the number of queries made to the SEM-Sign oracle and User-Sign oracle altogether, q_k denotes the number of queries made to the oracle public key extraction and private key extraction altogether, $q_{h_0}, q_{h_1}, q_{h_2}, q_{h_3}$ denote the number of queries made to the oracle h_0, h_1, h_2, and h_3 respectively.

Proof: Here we follow the idea from the schemes in [2] and [3]. Assume there exists a Type I adversary A_I. We start by describing how an adversary A_I can be used by a probabilistic polynomial time algorithm B to solve the factoring problem with probability at least ε' and in time at most t'.

Now, we will show how to build an algorithm B that on input of a given instance of factoring problem $N_1 = p_1 q_1$ for some unknown p_1 and q_1, outputs $p_1$or q_1 with non negligible probability.

Firstly B chooses $a \in Z_{N_1}^*$ satisfying Jacobi symbol (a/N_1)= −1, chooses a secure parameter $l \geq 160$, and sends (N_1, a, l) to A_I as public parameters. B picks an identity ID* at random as the challenged ID in this game. Then, B responds to A_I 's queries as follows:

***h_0*-Queries:** When A_I queries h_0, B checks the corresponding h_0-list and outputs c if such query has already been made. Otherwise, B picks $c \in \{0, 1\}^l$ at random, updates the h_0-list and outputs c as answer.

***h_1*-Queries:** When A_I queries h_1 on input ID, B checks the corresponding h_1-list and outputs h_1 if such value is defined. Otherwise, B chooses a random number $s \in Z_{N_1}^*$ and two random bits (c_1, c_2)$\in \{0, 1\}^2$, and returns $h_1 = \frac{s^{2^l}}{(-1)^{c_2} a^{c_1}} \bmod N_1$ as the answer, then adds the entry (ID, h_1, s, c_1, c_2) to the h_1-list.

***h_2*-Queries:** When A_I queries h_2 on input (ID, PK_{ID}, R, m||0), B checks the corresponding h_2-list and outputs h_s if such value is defined. Otherwise, B picks a random $h_s \in \{0, 1\}^l$, returns h_s as the answer, then adds the entry (ID, PK_{ID}, R, m||0, h_s) to the h_2-list.

***h_3*-Queries:** When A_I queries h_3 on input (ID, PK_{ID}, R, m||1), B checks the corresponding h_3-list and outputs h_u if such value is defined. Otherwise, B picks a random $h_u \in \{0, 1\}^l$, returns h_u as the answer, then adds the entry (ID, PK_{ID} , R, m||1, h_u) to the h_3-list.

SEM-Key Extraction Queries: For queries on input ID, if ID= ID*, B aborts. Otherwise, B acts as follows:

if ID already exists on the h_1-list in the tuple (ID, h_1, s, c_1, c_2), B returns S_{ID}=s, as well as (c_1, c_2) as the answer. Otherwise, B adds a new tuple containing ID in the same way as handling h_1-query, then returns s and (c_1, c_2) as the answer.

Public Key Extraction Queries: B keeps the list List of user public/private key. It first puts the public key $PK_{ID^*} = (N_2, P_{ID^*})$ of the identity ID* into List. Upon receiving a public key extraction query on ID. B looks up its list List to find out the corresponding entry. If it does not exist, B runs KeyGen to generate a private and public key pair. It stores the key pair in the list List and returns the public key as the query output.

Private Key Extraction Queries: For queries on input ID, if ID=ID*, B aborts. Otherwise, B looks up the list List to find out the corresponding entry. If it does not exist, B runs KeyGen to generate a private and public key pair. It stores the key pair in the list List and returns the private key as the query output.

Public Key Replacement Queries: Upon receiving a public key replacement query on ID, B searches the list List to replace the corresponding entry. If it does not exist, B creates a new entry for this identity.

SEM-Sign Queries: Upon receiving a query for a SEM signature of (m, ID, PK_{ID}). if $ID \neq ID$*, B can extract the SEM private key and run the algorithm SEM-Sign. Otherwise, B acts as follows:

- Recover the tuple (ID*, h_1, s, c_1, c_2) from the list h_1-list.
- Compute $H(ID^*) = (-1)^{c_2} a^{c_1} h_1$.
- Pick $t_s \in Z_{N_1}^*$, $h_s \in \{0,1\}^l$ and compute $R_s = \frac{t_s^{2^l}}{H(ID^*)^{h_s}}$
- Execute h_0 oracle simulation, get $c = h_0(Rs)$ and send it to A_I.
- After getting R_u from A_I, compute R mod N_1N_2, where R=R_s mod N_1, R=R_u mod N_2, set h_2(ID*, PK_{ID^*}, R, m||0)=h_s.
- Send (R_s, t_s, c_1, c_2) to A_I.

User-Sign Queries: Upon receiving a query for a user signature of (m, ID, PK_{ID}). if $ID \neq ID$*, B can extract the user private key and run the algorithm User-Sign. Otherwise, B acts as follows:

- An l-bit commitment c is obtained from the adversary, which models the first message that SEM should be sent to initiate the User-Sign process.

- Parse $PK_{ID^*} = (N_2, P_{ID^*})$, pick $t_u \in Z_{N_2}^*$, $h_u \in \{0, 1\}^l$, and compute $R_u = \frac{t_u^{2^l}}{P_{ID^*}^{h_u}}$.
- After obtained (R_s, t_s, c_1, c_2), compute R mod N_1N_2, where $R=R_s$ mod N_1, $R=R_u$ mod N_2.
- Search for h_0-list to find R_s' such that $h_0(R_s') = c$.
- If found, set $h_3(ID^*, PK_{ID^*}, R, m\|1)=h_u$.
- If $h_0(R_s)=c$, but no R_s' can be found or $R_s \neq R_s'$, stop User-Sign.
- Otherwise, compute t mod N_1N_2, where $t=t_s$ mod N_1, $t=t_u$ mod N_2, return (R, t, c_1, c_2) as an answer.

Complete-Sign Queries: If both the SEM private key and the user private key are available, signing is trivial. If either one of them is unavailable, this request can be simulated faithfully as a combination of the above two simulation.

Forgery: The next step of the simulation is to apply the general forking lemma in [18]: Let (t^*, R^*, c_1^*, c_2^*) be a forgery of a signature on a message m^* with respect to (ID^*, PK_{ID^*}) that is output by A_I at the end of the attack. If A_I does not output ID^* as a part of the forgery then B aborts.

B then replays A_I with the same random tape but different h_2 after the point (ID^*, PK_{ID^*}, R^*, $m^*\|0$). Suppose h_2 outputs h_s and h_s' in the first round and the second round respectively, where $h_s \neq h_s'$. So we get another valid forgery (t', R^*, c_1^*, c_2^*), i.e.

$$t^{*2^l} = R^* H(ID^*)^{h_s} \bmod N_1;$$
$$t'^{2^l} = R^* H(ID^*)^{h_s'} \bmod N_1$$

B thus gets $H(ID^*)^{h_s - h_s'} = (t^*/t')^{2^l} \bmod N_1$. Then B can easily compute a square root s' of $H(ID^*)$, setting $a=H(ID^*)$, $w = h_s - h_s'$, $X = (t^*/t')^{2^l} \bmod N_1$. Now, B searches in the h_1-list to find the entry <ID, h_1, s, c_1, c_2>, and if $s' \neq \pm s^{2^{l-1}} \bmod N_1$, N_1 could be factored. Otherwise, B reports failure. Note, s was some random 2^l root of $H(ID^*)$ chosen by B, which is independent of A_I's view. Thus, the probability that $s' \neq \pm s^{2^{l-1}} m \bmod N_1$ is 1/2. Since B has to run A_I twice, and takes some additional operations to factor N_1, such as exponentiations over modulo N_1, GCD operations for division and factoring N_1, the time B needs to factor N_1 is estimated as $t' = 2t + O(k^2 l + k^3)$.

Let ε' be the probability that N_1 could be factored, ε be the probability that A_I forges a signature in the real attack, and ε^* be the probability that A_I forges a signature in a single run in our simulation.

In the simulation of SEM-Sign oracle, embedding h_s as the response of h_2 is not possible if A_I has queried the h_2 value of (ID, PK_{ID}, R, $m\|0$) beforehand. We consider the case that h_2 has previously queried and the case that it was not. In the first case, A_I probably knows R_s and may have deliberately queried such value. However, since t_s is chosen randomly by B independent of A_I's view, the probability that A_I made such h_0 query is at most $(q_{h_0} + q_s)/|Q_{N_1}|$. In the second case, the view of A_I is completely independent of R_s. The probability that R appeared (by chance) in a previous q_{h_2} query is against at most $(q_{h_2} + q_s)/|Q_{N_1N_2}|$.

The User-Sign query simulation fails if $h_0(R_s)=c$, but no R_s' can be found or $R_s \neq R_s'$. For the first case, the probability that A_I can predict $h_0(R_s)=c$ without asking the random oracle is at most $q_s/2^l$. For the second case, collision must have occurred and the probability for this is at most $(q_{h_0} + q_s + 1)^2/|Q_{N_1}|$. We just assume A_I asked for $h_3(ID, PK_{ID}, R, m\|1)$ if R_s was not found since A_I knew the value of R_s before B.

So in our simulation, the probability of A_I winning the game is reduced to

$$\varepsilon^* \geq \varepsilon - q_s(q_{h_0} + q_s)2^{3-k} - q_s(q_{h_2} + q_s)2^{6-2k} - (q_s + q_{h_0} + 1)^2 2^{6-2k} - q_s / 2^l$$

Let p_h be the probability that the forgery was based on the h-th h_2-query in a single run. It could be easily calculated that

$$\varepsilon^* = \sum_{h=1}^{q_{h_2}+1} p_h$$

Let $p_{h,s}$ be the probability that the forgery was based on the h-th h_2-query in a single run, given a specific string s of length m, which determines the random tape of A_I and responses to all the queries. Therefore,

$$2^m p_h = \sum_{s \in \{0,1\}^m} p_{h,s}$$

For a specific string s, the probability that a forgery was based on the h-th h_2-query in both two runs is $p_{h,s}$ ($p_{h,s}$-2^l), since the answer of the h-th h_2-query in the second run should be different from the first run. Let P_h be the probability that a forgery was based on the h-th h_2-query in both two runs. Then

$$P_h = \sum_{s \in \{0,1\}^m} 2^{-m}(p_{h,s} - 2^{-l}) = 2^{-m}\Big(\sum_{s \in \{0,1\}^m} p_{h,s}^2 - 2^{-l} \sum_{s \in \{0,1\}^m} p_{h,s}\Big) \geq p_h^2 - 2^{-l} p_h$$

Therefore, the probability that A_I outputs two forgeries that are based on the same h_2-queries in both runs is estimated as

$$\sum_{h=1}^{q_{h_2}+1} P_h \geq \sum_{h=1}^{q_{h_2}+1} p_h^2 - \sum_{h=1}^{q_{h_2}+1} 2^{-l} p_h \geq \frac{\varepsilon^{*2}}{q_{h_2}+1} - 2^{-l}\varepsilon^* \geq \frac{\Delta^2}{q_{h_2}+1} - 2^{-l}\Delta$$

where

$$\Delta = \varepsilon - q_s(q_{h_0} + q_s)2^{3-k} - q_s(q_{h_2} + q_s)2^{6-2k} - (q_s + q_{h_0} + 1)^2 2^{6-2k} - q_s / 2^l$$

Moreover, outputting two forgeries on the same h_2-queries means that B has a probability of 1/2 to factor N_1. In addition, B needs to guess which identity A_I is going to forge the signature. The probability of guessing correctly is $1/q_{h_1}$, so we have $\varepsilon' \geq \sum_{h=1}^{q_{h_2}+1} P_h / 2q_{h_1}$.

Theorem 2. If the factoring problem is (t', ε')-hard, then our scheme is (t, ε)-existential unforgeable against Type II adversary,satisfying

$$\varepsilon' \geq \frac{1}{2q_k}\left(\frac{\Delta^2}{q_{h_3}+1} - 2^{-l}\Delta\right),\ t' = t + O(k^2 l + k^3),$$

where

$$\Delta = \varepsilon - (q_s + q_{h_0} + 1)^2 2^{6-2k} - q_s / 2^l,$$

(k, l) are security parameters introduced in section 4, $q_s, q_k, q_{h_0}, q_{h_2}, q_{h_3}$ have the same meaning as those in Theorem 1.

Proof: Assume there exists a Type II adversary A_{II} against our scheme. We are going to construct another PPT algorithm B that makes use of A_{II} to solve the factoring problem with probability at least ε' and in time at most t'.

Now, we will show how to build an algorithm B that on input of a given instance of factoring problem N_2= p_2q_2 for some unknown p_2 and q_2, outputs p_2 or q_2 with non negligible probability.

Firstly, B generates two random primes p_1, q_1, such that $p_1 = 2p_1' + 1$, $q_1 = 2q_1' + 1$, with p_1', q_1' being primes too, satisfying $2^{k-1} \leq (p_1-1)(q_1-1)$ and $p_1q_1 < 2^k$, then computes $N_1 = p_1q_1$ and sets (p_1, q_1, p_1', q_1') as the KGC's master-key, B chooses $a \in_R Z_{N_1}^*$, satisfying Jacobi symbol $(a/N_1)=-1$, chooses a secure parameter $l \geq 160$, and sets (N_1, a, l) as public parameters. Finally B sends public parameters and master-key to A_{II}. Since A_{II} has access to the master-key, he can do SEM-Extract and SEM-Sign himself, thus the hash function h_1() is not modelled as a random oracle in this case.

Suppose that A_{II} can forge a valid signature on message m* for identity ID* under public key PK_{ID*}. B picks a random $x_{ID*} \in Z_{N_2}^*$, computes $P_{ID*} = x_{ID*}^{2^l} \bmod N_2$ and sets ID*'s public key as PK_{ID*}=(N_2, P_{ID*}).

B responds to A_{II}'s queries as follows:

$\boldsymbol{h_0}$-Qeries, $\boldsymbol{h_2}$-Queries, $\boldsymbol{h_3}$-Queries: Random oracles for these queries are modeled the same as described in Theorem 1.

Public Key Extraction Queries: Upon receiving a query for a public key of an identity ID, if $ID \neq ID$*, B randomly picks two random primes p_3, q_3, such that $p_3 = 2p_3' + 1, q_3 = 2q_3' + 1$, with p_3', q_3' being primes too, satisfying $2^{k-1} \leq (p_3-1)(q_3-1)$ and $p_3q_3 < 2^k$, then compute $N_3 = p_3q_3$, picks a random $x_{ID} \in Z_{N_3}^*$, sets sk_{ID}=(x_{ID}, p_3, q_3) as ID's private value, computes $P_{ID} = x_{ID}^{2^l} \bmod N_3$, returns PK_{ID}=(N_3, P_{ID}) as answer and adds the tuple (ID, sk_{ID}, PK_{ID}) to K-list which is initially empty. Otherwise, returns PK_{ID*}.

Private Key Extraction Queries: Upon receiving a query for a private key of an identity ID, if $ID = ID$*, B aborts. Otherwise, B searches K-list for the entry (ID, sk_{ID}, PK_{ID}), generating a new key pair if this does not exist, and returns sk_{ID}.

User-Sign Queries: Note that at any time during the simulation, equipped with those user private keys for any $ID \neq ID$*, A_{II} is able to generate partial signatures on any message m. For ID= ID*, the simulation is as follows.

- An l-bit commitment c is obtained from the adversary, which models the first message that SEM should be sent to initiate the User-Sign process.
- Pick $t_u \in Z_{N_2}^*$, $h_u \in \{0, 1\}^l$, and compute $R_u = \frac{t_u^{2^l}}{P_{ID*}^{h_u}}$.
- After obtained (R_s, t_s, c_1, c_2), compute R mod N_1N_2, where R=R_s mod N_1, R=R_u mod N_2.
- Search for h_0-list to find R_s' such that $h_0(R_s') = c$.
- If found, set h_3(ID*, PK_{ID*}, R, m||1)=h_u.
- If $h_0(R_s)$=c, but no R_s' can be found or $R_s \neq R_s'$, stop User-Sign.
- Otherwise, compute t mod N_1N_2, where t=t_s mod N_1, t=t_u mod N_2, return (R, t, c_1, c_2) as an answer.

Complete-Sign Queries: If the user private key is available, signing is trivial. Otherwise, this request can be simulated faithfully similar to above.

Forgery: The next step of the simulation is to apply the general forking lemma in [18]: Let (t^*, R^*, c_1^*, c_2^*) be a forgery of a signature on a message m* with respect to (ID*, PK_{ID*}) that is output by A_{II} at the end of the attack. If A_{II} does not output ID* as a part of the forgery then B aborts.

B then replays A_{II} with the same random tape but different h_3 after the point (ID*, PK_{ID*}, R*, m*||1). Suppose h_3 outputs h_u and h_u' in the first round and the second round respectively, where $h_u \neq h_u'$. So we get another valid forgery (t', R^*, c_1^*, c_2^*), i.e.

$$t^{*2^l} = R^* P_{ID*}^{h_u} \bmod N_2;$$

$$t'^{2^l} = R^* P_{ID*}^{h_u'} \bmod N_2$$

B thus gets $P_{ID*}^{h_u - h_u'} = (t^*/t')^{2^l} \bmod N_2$. By applying Theorem 2, B can easily compute a square root s' of P_{ID*}, a=P_{ID*}, $w = h_u - h_u'$ and $X = (t^*/t')^{2^l}$.

Let ε' be the probability that N_2 could be factored, ε be the probability that A_{II} forges a signature in the real attack, and ε^* be the probability that A_{II} forges a signature in a single run in our simulation.

The User-Sign query simulation fails if $h_0(R_s)=c$, but no R_s' can be found or $R_s \neq R_s'$. For the first case, the probability that A_{II} can predict $h_0(R_s)=c$ without asking the random oracle is at most $q_s/2^l$. For the second case, collision must have occurred and the probability for this is at most $(q_{h_0}+q_s+1)^2/|Q_{N_1}|$.

So in our simulation, the probability of A_{II} winning the game is reduced to

$$\varepsilon^* \geq \varepsilon-(q_s+q_{h_0}+1)^2 2^{6-2k}-q_s/2^l$$

Let p_h be the probability that the forgery was based on the h-th h_3-query in a single run. It could be easily calculated that

$$\varepsilon^* = \sum_{h=1}^{q_{h_3}+1} p_h$$

Let $p_{h,s}$ be the probability that the forgery was based on the h-th h_3-query in a single run, given a specific string s of length m, which determines the random tape of A_{II} and responses to all the queries. Therefore,

$$2^m p_h = \sum_{s\in\{0,1\}^m} p_{h,s}$$

For a specific string s, the probability that a forgery was based on the h-th h_3-query in both two runs is $p_{h,s}$ ($p_{h,s}-2^l$), since the answer of the h-th h_3-query in the second run should be different from the first run. Let P_h be the probability that a forgery was based on the h-th h_3-query in both two runs. Then

$$\begin{aligned} P_h &= \sum_{s\in\{0,1\}^m} 2^{-m}(p_{h,s}-2^{-l}) \\ &= 2^{-m}\Big(\sum_{s\in\{0,1\}^m} p_{h,s}^2 - 2^{-l}\sum_{s\in\{0,1\}^m} p_{h,s}\Big) \geq p_h^2 - 2^{-l}p_h \end{aligned}$$

Therefore, the probability that A_{II} outputs two forgeries that are based on the same h_3-queries in both runs is estimated as

$$\begin{aligned} \sum_{h=1}^{q_{h_3}+1} P_h &\geq \sum_{h=1}^{q_{h_3}+1} p_h^2 - \sum_{h=1}^{q_{h_3}+1} 2^{-l}p_h \geq \frac{\varepsilon^{*2}}{q_{h_3}+1} - 2^{-l}\varepsilon^* \\ &\geq \frac{\Delta^2}{q_{h_3}+1} - 2^{-l}\Delta \end{aligned}$$

where $\Delta = \varepsilon-(q_s+q_{h_0}+1)^2 2^{6-2k}-q_s/2^l$. Moreover, outputting two forgeries on the same h_3-queries means that B has a probability of 1/2 to factor N_2. In addition, B needs to guess which identity A_{II} is going to forge the signature, and assign the problem instance element as the public key of this identity. The probability of guessing correctly is $1/q_k$, so we have $\varepsilon' \geq \sum_{h=1}^{q_{h_3}+1} P_h / 2q_k$.

Time complexity analysis is similar to the proof of Type I Adversary.

VI. CONCLUSION

SMC cryptography solves the instantaneous key revocation problem in CLPKC while eliminating key escrow problem inherited in IBC. SMC signature is one of the most important security primitives in SMC cryptography. In this paper, we propose a provable secure SMC signature scheme without bilinear pairing. It is provable secure in the random oracle model based on the intractability of the factoring problem. Our scheme is efficient in the sense that no bilinear pairing is involved.

REFERENCES

[1] L. Harn, J. Ren, C. Lin. Design of DL-based certificateless digital signatures. Journal of Systems and Software(2009). pp.789-793.

[2] W. S. Yap, S. S.M. Chow, S. H. Heng, and B.M. Goi. Security Mediated Certificateless Signatures. ACNS 2007, LNCS 4521, pp. 459-477, 2007.

[3] Z. C. Cai, Z. F. Cao, and X. L. Dong. Identity-based signature scheme based on quadratic residues. Science in China Series F: Information Sciences. Springer-Verlag, 2007. 50:373-380

[4] A. Shamir. Identity-Based Cryptosystems and Signature Schemes, In Proc. of Crypto1984, LNCS 196, pp. 47-53, Springer-Verlag, 1984.

[5] S. S. Al-Riyami and K. G. Paterson. Certificateless Public Key Cryptography. Advances in Cryptology CAsiaCrypt, LNCS, vol. 2894. Springer-Verlag, pp. 452-473.

[6] S. Chow, C. Boyd, and J. Gonz´alez. Security-mediated certificateless cryptography. In PKC 2006, volume 3958 of LNCS, pages 508-524. Springer-Verlag, 2006.

[7] H. Du, Q. Wen. Efficient and provably-secure certificateless short signature scheme from bilinear pairings. Computer Standards and Interfaces 31 (2), 390-394, 2009.

[8] M. Gorantla, R. Gangishetti, M. Das, and A. Saxena. An effective certificateless signature scheme based on bilinear pairings. In WOSIS 2005, pages 31-39. INSTICC Press, 2005.

[9] X. Huang, W. Susilo, Y. Mu, and F. Zhang. On the security of certificateless signature schemes from Asiacrypt 2003. In CANS 2005, pages 13-25. Springer-Verlag, 2005. LNCS No. 3810.

[10] X. Huang, W. Susilo, Y. Mu, and F. Zhang . Certificateless designated verifier signature schemes. In AINA 2006, pages 15-19. IEEE Computer Society, 2006.

[11] X. Hu, Z.G. Qin, F.G. Li, An improved certificateless signature scheme secure in the standard model. Fundamenta Informaticae(2008) 193-206.

[12] H. S. Ju, D. Y. Kim, D. H. Lee, J. Lim, and K. Chun. Efficient Revocation of Security Capability in Certificateless Public Key Cryptography. In Knowledge-Based Intelligent Information and Engineering Systems 2005, LNAI 3682, pp. 453-459.

[13] J. K. Liu, M. H. Au, and W. Susilo. Self-Generated -Certificate Public Key Cryptography and Certificateless Signature / Encryption Scheme in the Standard Model. In Proceedings of ASIACCS 2007.

[14] Z. F. Zhang, D. S. Wong, J. Xu, and D. G. Feng. Certificateless Public-Key Signature: Security Model and Efficient Construction. In Proceedings of ACNS 2006, LNCS 3989, pp. 293-308. Journal version appeared in Designs, Codes and Cryptography, 42/2, pp. 109-126, 2007.

[15] D. H. Yum and P. J. Lee. Generic construction of certificateless signature. In ACISP 04, pages 200-211. Springer-Verlag, 2004. LNCS No. 3108.

[16] D. Boneh and M. Franklin. Identity-based encryption from the Weil Pairing. Advances in Cryptology-CRYPTO 2001, Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2001, 2193: 213-229

[17] V. Shoup. A Computational Introduction to Number Theory and Algebra, Cambridge University Press, 2005. 534

[18] M. Bellare and G. Neven. Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma In Proceedings of ACM-CCS 2006, pp. 390-399.

Zhongmei Wan was born in Jiangsu Province, China, on May 4, 1973. She received her Bachelor degree from Yancheng Normal University, China in 1994. She received her Master degree from Zhejiang University, China in 2001. She is currently lecturer at the College of Science, Hohai University, Nanjing, China. Her major interests are cryptography theory and technology, cryptography protocol , network and information security.