Security Mediated Certificateless Signatures

Wun-She Yap[1], Sherman S.M. Chow[2], Swee-Huay Heng[3], and Bok-Min Goi[1]

[1] Centre for Cryptography and Information Security, FOE
Multimedia University, 63100 Cyberjaya, Malaysia
{wsyap,bmgoi}@mmu.edu.my
[2] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu
[3] Centre for Cryptography and Information Security, FIST
Multimedia University, Jln Ayer Keroh Lama, 75450 Melaka, Malaysia
shheng@mmu.edu.my

Abstract. In PKC 2006, Chow, Boyd and González Neito introduced the notion of security mediated certificateless (SMC) cryptography. SMC cryptography equips certificateless cryptography with instantaneous revocation. They presented a formal security model with two constructions for SMC encryption. This paper studies SMC signatures. We first present a security analysis of a previous attempt by Ju *et al.* in constructing a SMC signature scheme. We then formalize the notion of SMC signatures and propose the first concrete provable scheme without bilinear pairing. Our scheme is existential unforgeable in the random oracle model based on the intractability of the discrete logarithm problem, has a short public key size, and achieves a trust level which is the same as that of a traditional public key signature.

Keywords: security mediated, certificateless, SMC cryptography.

1 Introduction

Efficient revocation of public key certificates has always been a critical issue in the public key infrastructure (PKI). Several methods such as certificate revocation list (CRL), online certificate status protocol (OCSP) [22] and Novomodo (a scalable and small-bandwidth certificate validation scheme) [21] had been proposed to solve the certificate management issue. However, the search for a satisfactory solution continues. In USENIX 2001, Boneh *et al.* [5] first introduced a method for obtaining instantaneous revocation in RSA-type cryptosystems (more details are given in [4]). Rather than revoking the user's certificate, this new approach revokes the user's ability to perform the cryptographic operations such as signing and decryption. They introduced a new entity, which is an online semi-trusted server called as a security mediator (SEM). To sign or decrypt a message, a client must first obtain a message-specific token from its SEM. Without this token, the client cannot accomplish the intended task. To revoke the user's ability to sign or decrypt, the SEM is instructed to stop issuing tokens for the future request of that user.

J. Katz and M. Yung (Eds.): ACNS 2007, LNCS 4521, pp. 459–477, 2007.

In 2003, Al-Riyami and Paterson [1] introduced an intermediate model between the traditional public key cryptography (TPKC) and the identity-based cryptography (IBC), known as certificateless public key cryptography (CLPKC). In CLPKC, both the partial private key and user secret key are necessary in signing or decryption. On the other hand, the identity (ID) and the user public key are both required in verification and encryption respectively. Without using certificates that are essential in IBC, CLPKC achieves implicit certification (through the partial private key) while does not suffer from the inherent key escrow problem in IBC (through the user secret key). Thus, CLPKC still maintains the advantages enjoyed in TPKC and IBC.

For the key revocation in CLPKC, a pragmatic way to deal with the revocation problem is to concatenate a validity period to the ID. However, this involves the need to re-issue the partial private key periodically for each validity period, thus burdens the trusted key generation center (KGC), and requires users to store different IDs that are concatenated with different validity dates. Besides, this method does not fit an environment when immediate revocation is required.

Recently, Chow, Boyd and González Nieto initiated the study of SMC cryptography from a formal point of view in [9]. SMC cryptography solves the instantaneous key revocation problem in CLPKC while maintaining the merits in CLPKC: implicit certification without key escrow. In short, SMC cryptography achieves a set of features that no previous paradigms can satisfy simultaneously, so it provided a new compromise between the various desirable features.

1.1 Related Work

Chow, Boyd and González Neito discussed various facets of SMC cryptography and formalized the notion of SMC encryption in [9]. We note that Ju *et al.* has also discussed the idea of SMC cryptography briefly in [17]. Based on the Libert and Quisquater revocation mechanism [20] and Al-Riyami and Paterson various certificateless schemes [1], Ju *et al.* introduced a signature scheme, an encryption scheme and a hierarchical variant. Generally, [17] did not provide the necessary details clearly compared with [9]. More details are given in the Appendix. Ju *et al.*'s [17] signature scheme was derived from the Al-Riyami and Paterson certificateless signature (CLS) scheme [1] (the key construction and the `Verify` algorithms are just identical), although the authors did not cite it clearly. Their scheme only achieves level 2.

There are a number of CLS scheme proposals after [1], which are based on either bilinear pairing [8,13,16,18,19,27] or identity-based signatures (IBS) [15,26]. Huang *et al.*'s scheme [16] is the revised version of Al-Riyami and Paterson's insecure scheme [1], and Cao *et al.*'s scheme [8] is revised from the insecure scheme of Gorantla-Saxena [13]. Liu *et al.*'s scheme [18] is the only one proposed without random oracles; however, it is also the least efficient.

1.2 Trust Levels

Trust levels referred to the three different levels of trust placed on the trusted third party as defined by Girault [12], the higher level the more desirable.

- *Level 1*: The authority knows (or can easily compute) the private keys and is capable of impersonating any user without being detected.
- *Level 2*: The authority does not know the private keys, but it can impersonate any user by generating false certificates without being detected.
- *Level 3*: The authority cannot compute the private keys and if it generates false certificates for users, such generation can be detected.

Table 1 summarizes the comparison of various security mediated signature.

Table 1. Properties of Security Mediated Signatures

Schemes in Different Paradigms	Implicit Certification	Escrow Freeness	Trust Level
Security mediated (traditional) signature [4,5,20]	✘	✘	1
Security mediated identity-based signature [11]	✓	✘	1
Security mediated certificateless signature	✓	✓	3

While key escrow is inherent in IBC, the same problem is not necessary inherent in all security mediated traditional signatures (with trust level 1). The above table considers the security mediated traditional signatures in [4,5,20], in which the certification authority is the one who is responsible in generating the user private key, they could only achieve trust level 1.

1.3 Our Contributions

We initiate the formal study of revocation in certificateless signatures paradigm. Our contributions are three-fold.

Attack. We show that the previous attempt by Ju *et al.* in constructing SMC signature scheme is flawed. More precisely, their scheme does not support revocation at all since the user can get the SEM private key after interacting with the SEM, i.e. after receiving a partial signature. Consequently, the user can sign any subsequent messages thereafter without the assistance of the SEM. In short, the complication involved the SEM is redundant and useless.

Recently, the Al-Riyami and Paterson CLS scheme was shown to be insecure by a realistic key replacement attack by Huang *et al.* in [16]. We show that Ju *et al.*'s SMC signature scheme is vulnerable to a similar attack as well.

Security Model. We formalize the security model of SMC signature. We highlight the differences between our model and existing certificateless signature model. We also discuss the subtleties in the definition of "public key" arose from the probabilistic key generation, which introduces a public component selected by the KGC instead of the user.

Efficient Construction. We then present the first concrete provable secure SMC signature scheme without bilinear pairing. Our scheme is existential unforgeable in the random oracle model based on the intractability of the discrete logarithm problem and achieves a trust level which is the same as that of a traditional digital signature scheme. It is worth noting that our proposed scheme can be transformed into an efficient CLS scheme without pairing.

Organization. Section 2 provides the framework and the security model of SMC signatures. Section 3 presents both the insider and outsider attacks on Ju *et al.*'s SMC signature. Section 4 shows the proposed construction of SMC signature together with its security and performance analysis.

2 Security Mediated Certificateless Signatures

Here we present the framework and the security model of SMC signature scheme. We adopt the compact but versatile model of [15] improved from [1,16,26].

2.1 Framework

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

1. `Setup` *is a probabilistic algorithm that takes as input a security parameter in the form of* 1^k *and returns a master key* s *and a parameter list* $params$.
2. `KeyGen` *is a probabilistic algorithm that takes as input a parameter list* $params$. *It picks a secret value at random. The public key is computed based on the selected value. It returns a pair of matching public and private keys* (P_{ID}, x_{ID}).
3. `Register` *is a probabilistic algorithm that takes as input a parameter list* $params$, *the master key* s, *an user identity* ID *and a public key* P_{ID}. *It returns the SEM private signing key* D_{ID}.
4. `Sign` *is an interactive probabilistic protocol between the user and the SEM. Their common inputs include a parameter list* $params$, *a message* m, *and an user identity* ID. *The SEM has an additional input of* D_{ID} *to run the sub-algorithm* `SEM-Sign`*; while the user has an additional input of* x_{ID} *to run the sub-algorithm* `User-Sign`. *The protocol finishes with either a signature* σ, *or* $\perp$ *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. `Verify` *is a deterministic algorithm that takes as input a parameter lists* $params$, *a message* m, *an user identity* ID, *an user public key* P_{ID} *and a signature* σ. *It returns* $true$ *or* $false$.

More precisely, the scheme flows as follows. First, the KGC will take as input a security parameter 1^k to generate the *params* and the master key s by running the `Setup` algorithm. Then, the user who holds identity ID runs the `KeyGen` algorithm to generate the public key P_{ID} and the secret value x_{ID}. The public key P_{ID} is registered with the KGC via authentication and the `Register` algorithm, during which the KGC will use the master key s to generate the SEM private

signing key D_{ID}. This key needs to be transmitted to the SEM authentically and confidentially through a secure channel. The secrets held by the SEM and the user are different. During the running of the `Sign` protocol, the SEM and the user uses their corresponding secret to sign the message cooperatively. Finally, a recipient needs both the public key and ID to verify the validity of the signature by using the `Verify` algorithm.

We keep the following new features of the model introduced in [15]:

1. The SEM private key generation by the KGC is probabilistic.
2. The SEM private key generated and the user private key can be "mixed" together in some randomized way each time during signature generation.
3. A single probabilistic algorithm is used for creating the public/private key pair.

The first point worths more discussion. In most of the existing concrete CLS schemes (i.e. excluding generic constructions using other signature schemes like IBS as a building block), deterministic algorithm is used such that there is only one partial private key (the SEM private key in our context) corresponding to each identity. If a probabilistic algorithm is used, the randomness introduced may give rise to a new "public component" that should be included in the signature. This gives ambiguity in what is meant by "user public key". No discussion has been made on this subtle point so far. We defer our discussion on this issue to Section 4.4, with the hope that the abstract concept can be better understood based on a concrete scheme.

2.2 Security Model

We consider the security against existential forgery on adaptive chosen message and identity attacks. The security model of SMC signature is a mixture of the models in [9,15]. There are two types of adversary with different capabilities. The Type I adversary $\mathcal{A}_{\mathcal{I}}$ acts as a dishonest user. The Type II adversary $\mathcal{A}_{\mathcal{II}}$ acts as a malicious KGC (we do not consider a rogue SEM explicitly since it is strictly weaker than the Type II adversary).

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

1. `Setup`: $\mathcal{C}$ takes as input 1^k, runs the `Setup` algorithm, and gives $\mathcal{A}$ the resulting *params*. The master key is given to $\mathcal{A}$ if it is a Type II adversary.
2. `Attack`: $\mathcal{A}$ issues a sequence of requests, each request being either a query of `Create`, `Replace`, `SEM-Extract`, `User-Extract`, `SEM-Sign`, `User-Sign` or `Complete-Sign` for a particular entity.
   `Create` queries create users by either registering the user public key when playing against $\mathcal{A}_I$, or executing the `Key-Gen` algorithm for $\mathcal{A}_{II}$. `Replace` queries let $\mathcal{A}_I$ to change user public key at its wish. Two `Extract` queries return the SEM and user private key respectively. `Sign` queries are to be

explained shortly. These queries may be asked adaptively, subjected to the rules on adversary behaviors to be defined below.
3. `Forgery`: $\mathcal{A}$ outputs a signature σ on message m signed by user ID with public key P_{ID}. The only restriction is that (m, ID) does not appear in the set of previous `Sign` queries. $\mathcal{A}$ wins the game if `Verify`$(params, \sigma, m, ID, P_{ID})$ is *true*. The advantage of $\mathcal{A}$ is defined as the probability that it wins.

Differences from the Existing Certificateless Signature Models: We highlight the differences from the existing models on the following types of query.

1. `SEM-Sign`: On input a message m and an identity ID, the adversary is returned with the partial signing result by using D_{ID}.
2. `User-Sign`: On input a message m and a public key P_{ID}, the adversary is returned with the partial signing result by using x_{ID}, *even* if the public key P_{ID} is *previously replaced* by the (Type I) adversary.
3. `Complete-Sign`: On input a message m and a public key P_{ID}, the adversary is returned with the complete signing result by using x_{ID} and D_{ID}, *even* if the user public key P_{ID} is *previously replaced* by the (Type I) adversary.

The first two queries capture the adversary's capabilities to ask for partial signing results, which are not considered in the traditional certificateless setting. The last one appears in a weaker form in the existing certificateless signature model [15,26], such that signing query for a replaced public key requires the adversary to submit the corresponding private key. If an invalid one is submitted (or no key is submitted at all), an invalid signature will be given. This restriction makes the model weaker than its counterpart in encryption [1,9], where the decryption oracle gives valid result even if the public key has been replaced.

It may seem unrealistic to entertain any signing query of the replaced public key from the first glance. We try to give some intuition here. Suppose the adversary has an access of a cryptographic device (e.g. giving standard signatures) with some public/private key pair. It is not impossible that the combination of this particular public key with a certain identity may enable the adversary to forge. In this case, the signing query models some useful knowledge about the replaced public key that the adversary may obtain *outside* our system. Our security formulation guarantees unforgeability in this case.

Now we spell out some restrictions placed on the adversaries.

SMC Signatures Type I Adversary: Adversary $\mathcal{A}_{\mathcal{I}}$ does not have access to the master key. On the other hand, $\mathcal{A}_{\mathcal{I}}$ may request and replace the public keys, extract the SEM private key and the user private key and make the sign queries. Here are several natural restrictions on such a Type I adversary:

1. $\mathcal{A}_{\mathcal{I}}$ cannot extract the SEM private key of the challenge identity ID^*.
2. $\mathcal{A}_{\mathcal{I}}$ has not issued any `SEM-Sign` query on the forged message m for the challenged identity ID^*.
3. $\mathcal{A}_{\mathcal{I}}$ cannot make a `Complete-Sign` query on the forged message m for the challenged identity ID^*.

SMC Signatures Type II Adversary: Adversary $\mathcal{A}_{\mathcal{II}}$ does have access to the master key, but cannot replace the public keys of entities. Adversary $\mathcal{A}_{\mathcal{II}}$ can compute the SEM private key itself, request the public keys, extract users' private key and make the sign queries, all for the identities of its choice. The restrictions on this type of adversary are:

1. $\mathcal{A}_{\mathcal{II}}$ cannot replace the public keys of the challenge identity ID^*.
2. $\mathcal{A}_{\mathcal{II}}$ cannot extract the user private key for ID^* at any point.
3. $\mathcal{A}_{\mathcal{II}}$ cannot make an `User-Sign` query on the forged message m for the challenged identity ID^*.
4. $\mathcal{A}_{\mathcal{II}}$ cannot make a `Complete-Sign` query on the forged message m for the challenged identity ID^*.

Definition 2. *A SMC signature scheme is secure against EUF-CMIA if there is no efficient adversary in the above game with a non-negligible advantage in the security parameter k for both types of adversary.*

3 Critical Review on Ju *et al.*'s Scheme

First, we review the construction of Ju *et al.*'s scheme.

- `Setup`: Given a security parameter k, the KGC performs the steps below:
  1. Run the Bilinear Diffie-Hellman parameter generator $\mathcal{IG}$ [6] with input k in order to generate output $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of some prime order q and a *bilinear map* $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
  2. The system parameters are $params = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}(\cdot,\cdot), P, P_0, H_1(\cdot), H_2(\cdot) \rangle$ where P is an arbitrary generator in $\mathbb{G}_1$, $P_0 = sP$ where s is picked uniformly at random from Z_q^*, H_1 and H_2 are cryptographic hash functions where $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \{0,1\}^* \times \mathbb{G}_2 \to Z_q^*$.
  3. The master key is s.
- `Key Generation`: Given an user with identity $ID_A \in \{0,1\}^*$,
  1. The KGC computes $Q_A = H_1(ID_A) \in \mathbb{G}_1$ and $D_{ID_A} = sQ_A$.
  2. The KGC randomly chooses $D^U_{ID_A} \in \mathbb{G}_1$.
  3. The KGC sends the SEM private key $D^U_{ID_A}$ to the user A and $D^S_{ID_A} = D_{ID_A} - D^U_{ID_A}$ to the SEM, over a confidential and authentic channel.
  4. The user A selects $x_A \in Z_q^*$ as his private key and constructs his public key as $\langle X_A, Y_A \rangle = \langle x_A P, x_A P_0 \rangle$.
- `Sign`: Now user A with identity ID_A wants to get a signature on $m \in \{0,1\}^*$.
  1. A chooses a random element $a \in Z_q^*$, computes the following values: $r = \hat{e}(aP, P) \in \mathbb{G}_2$, $\upsilon = H_2(m, r) \in Z_q^*$, and $T^U = \upsilon D^U_{ID_A}$.
  2. A sends υ to the SEM, who checks if A's public key has been revoked.
  3. If not, the SEM computes $T^S = \upsilon D^S_{ID_A}$ and sends it to A.
  4. Upon receiving T^S, A computes $T = x_A(T^S + T^U) + aP$.
  5. The final signature is $\langle T, \upsilon \rangle$.
- `Verify`: When receiving $\langle T, \upsilon \rangle$ on message $m \in \{0,1\}^*$ for identity ID_A and public key $\langle X_A, Y_A \rangle$, the verifier performs the following steps:
  1. If $\hat{e}(X_A, P_0) \neq \hat{e}(Y_A, P)$, return $\perp$ and abort.
  2. Compute $r' = \hat{e}(T, P) \cdot \hat{e}(Q_A, -Y_A)^{\upsilon}$.
  3. Accept the signature if and only if $\upsilon = H_2(m, r')$ holds.

3.1 Insiders Attack Against Revocation

The whole point of introducing a SEM is for the instant revocation of the user's ability to sign. The user cannot issue signature without the cooperation of the SEM since the SEM holds a SEM private key of each user, which is essential to compute a signature. In other words, the user must seek help from the SEM every time he/she wants to issue a new signature. It is thus natural to assume that a malicious user may interact with the SEM in anyway to gain knowledge about the partial private key and later produce a signature autonomously.

Now we show that it is easy for a malicious user to get his/her SEM private key by only one interaction with the SEM. After getting hold of the partial private key, the user can sign any subsequent messages thereafter at will without the assistance of the SEM anymore. This flaw is very disastrous since the scheme is no longer having the merit of security mediated cryptography, i.e. it does not provide the instantaneous key revocation property as claimed.

The flaw is that in their `Sign` algorithm, the SEM computes $T^S = \upsilon D^S_{ID_A}$ and sends it to the user A. The user A can easily compute the value of $D^S_{ID_A}$ by using the equation $D^S_{ID_A} = \upsilon^{-1} \cdot T^S$ as υ is a value to be included in the final signature anyway. Then, the user A can generate signature on any message with the complete knowledge of $D^S_{ID_A}$, $D^U_{ID_A}$ and x_A.

Here, we point out some considerations in proposing a provable secure SMC signature scheme. Firstly, we should ensure that the adversary does not gain any advantage from the above queries. The SEM private key and the user private key must be protected when issuing a partial signature. Leaking either one part of the private keys will result in an insecure scheme.

3.2 Key Replacement Attack by Any Malicious Outsider

Ju *et al.* scheme [17] follows the approach used in the Al-Riyami and Paterson CLS scheme [1] while the latter was derived from the Hess IBS scheme [14]. The Al-Riyami and Paterson scheme was proven insecure in the defined model of [16]. The insecurity lies in that [1] adopted a similar approach as [14] without considering the adversary's ability in CLPKC. Caution must be taken considering the ability of the adversary in replacing any public key, as no certificate is needed in authenticating the user public key.

Now we show that Ju *et al.*'s scheme [17] is vulnerable to the public key replacement attack by Type I adversary, similar to the technique in [16]. As defined in [1], Type I adversary represents a dishonest user who can replace user public key at will (as there is no certificate to authenticate the public key).

Our break is a strong one, which is universal forgery against no message attack, i.e. no signing oracle is required in the Type I adversarial model (the attack does not need any help from the SEM and the user in signing any message for any identity ID) and the forger can sign any message. Details are given below:

`Sign`: To sign a message m with identity ID_i, where $Q_i = H_1(ID_i)$:

1. Select a random $T \in \mathbb{G}_1$, compute $r = \hat{e}(T, P)\hat{e}(Q_i, -P_0)$ and $\upsilon = H_2(m, r)$.
2. Set $x_i = \upsilon^{-1} \in Z^*_q$, compute $X'_i = x_i P$ and $Y'_i = x_i P_0$.

3. Replace the user public key with $\langle X_i', Y_i' \rangle$, a clearly valid one.
4. Return $\langle T, \upsilon \rangle$ as a signature of m.

The forged signature is valid since

$$r' = \hat{e}(T, P) \cdot \hat{e}(Q_i, -Y_i')^{\upsilon} = \hat{e}(T, P) \cdot \hat{e}(Q_i, -x_i P_0)^{\upsilon} = \hat{e}(T, P) \cdot \hat{e}(Q_i, -P_0) = r.$$

Although it may be possible to fix the scheme by the countermeasures in [16], it still fails to provide instantaneous revocation capability. Instead of fixing, we propose a much more efficient provably secure scheme in the next section.

4 Our Proposed Construction

This section presents our proposed scheme derived from the Schnorr signature scheme [24]. The similar partial private key construction was used in [2]. The merit of this approach is that no pairing computation is needed at all.

- `Setup`. Given the security parameter k, the KGC performs the following.
  1. Generate two primes p and q such that $q|p-1$.
  2. Pick a generator g of Z_p^*.
  3. Pick $s \in Z_q^*$ uniformly at random and compute $Y = g^s$.
  4. Choose hash functions $H_0 : \{0,1\}^* \rightarrow \{0,1\}^{\ell}$, $H_1 : \{0,1\}^* \rightarrow Z_q^*$ and $H_2 : \{0,1\}^* \rightarrow Z_q^*$.
  5. Return $params = (p, q, g, Y, H_0, H_1, H_2)$ and master key is s.
- `KeyGen`: The user performs the following.
  1. Randomly select $x_{ID} \in Z_q^*$ as the user private key.
  2. Compute $P_{ID} = g^{x_{ID}} \in Z_q^*$ as the user public key.
- `Register`: Now user ID wants to register a public key P_{ID}:
  1. The KGC authenticates and registers (ID, P_{ID}), randomly picks $w \in Z_q^*$.
  2. The KGC computes $W = g^w$ and $d = w + sH_1(ID||W)$.
  3. The KGC sends the SEM private key $D_{ID} = \langle W, d \rangle$ and the (ID, P_{ID}) pair to the SEM over a confidential and authentic channel.
- `Sign`: 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 (I denotes the first part and II denotes the second):
  - `SEM-Sign` (I): randomly chooses $r_S \in Z_q^*$, computes $R_S = g^{r_S}$, sends $c = H_0(R_S)$ to the user.
  - `User-Sign` (I): randomly chooses $r_U \in Z_q^*$, sends $R_U = g^{r_U}$ to the SEM.
  - `SEM-Sign` (II): computes

$$R = R_S \cdot R_U, h_S = H_2(ID||W||0||P_{ID}||R||m), t = r_S + d \cdot h_S;$$

and sends back $\langle R_S, t \rangle$ to the user.

Note that the partial signing result generated by the SEM does not need to transmit to the user through a secure channel since any party other than the SEM and the user does not gain any advantage from it.

Depending on the scenarios, the user may want to verify the correctness of the partial signature given by the SEM by checking whether the equality $R_S = g^t (W \cdot Y^{H_1(ID||W)})^{-h_S}$ holds.

- `User-Sign` (II): checks if $c = H_0(R_s)$; if they are equal, computes

$$R = R_S \cdot R_U, h_U = H_2(ID||P_{ID}||1||W||R||m), v = r_U + x_{ID} \cdot h_U + t.$$

  The final signature is $\sigma = \langle R, V, (ID||P_{ID}||W)\rangle$.

– `Verify`: Given σ, accepts if and only if the following equality holds:

$$R = g^V {P_{ID}}^{-H_2(ID||P_{ID}||1||W||R||m)}(W \cdot Y^{H_1(ID||W)})^{-H_2(ID||W||0||P_{ID}||R||m)}.$$

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

$$\begin{aligned} R' &= g^V {P_{ID}}^{-h_U}(W \cdot Y^{H_1(ID||W)})^{(-h_S)} \\ &= g^{r_S+r_U+d\cdot h_S+x_{ID}\cdot h_U} g^{-x_{ID}\cdot h_U}(g^w \cdot g^{sH_1(ID||W)})^{(-h_S)} \\ &= g^{r_S+r_U+d\cdot h_S} g^{(w+sH_1(ID||W))(-h_S)} \\ &= g^{r_S+r_U+d\cdot h_S} g^{(d)(-h_S)} \\ &= g^{r_S+r_U} = R \end{aligned}$$

4.1 Generic Construction

One of the major characteristics of SMC signature is that the SEM private key generated by the KGC and the user private key are required in different partial signing algorithm; which is different with

1. the traditional CLS model [1,16,26], where the partial private key generated by the KGC and the user private key are combined into a single private key,
2. the new CLS model [15], where the partial private key generated by the KGC and the user private key are both inputs of the same signing algorithm.

The signing algorithm of the generic CLS construction in [15] can actually be partitioned into two parts – the first part takes the partial private key (but not the user private key) for partial signature generation, and the second part which takes the user private key and the partial signature to give the final one.

Based on these observations, it is tempting to use the above partition to give a generic construction of SMC signature scheme. Nevertheless, similar to the weakness of the generic construction of SMC encryption in [9] (where decryption oracle may not work if the public key has been replaced), such approach cannot give us signing oracles which still work after the public key has been replaced.

4.2 Extension from CLS

Another possible approach to construct SMC signature scheme is to extend from existing concrete CLS scheme. We note that Zhang *et al.*'s scheme [27] can be partitioned in the way as described above, and the resulting scheme supports partial signing queries even if the public key has been replaced. However, such level of security is achieved at the expense of high computational complexity.

4.3 Efficiency Analysis

We denote "Exp" the exponentiation in Z_p^*, "MtP" the `MapToPoint` encoding function in [6] and $\hat{e}(\cdot,\cdot)$ the pairing operation. The "n" letters appear in the "MtP" columns denote n multiplications in Z_p^* are required in encoding an identity or a message, where n denotes the bit-length of an identity or a message to be signed. Generally speaking, the operations are listed according to their efficiency in descending order.

Regarding our scheme's performance, the SEM private key generation requires only 1 Exp. To verify, 4 Exp's are needed. For signing, each of the user and the SEM compute 1 Exp. Table 2 compares the performance of our proposal with existing CLS schemes. Note that we include the total computational requirement of both the SEM and user in signing, and ignore the `MapToPoint` operation to derive the public key during the signing and the verification. These considerations are unfavorable to our proposal.

It is clear that our proposal outperforms all existing ones.

Table 2. Efficiency Analysis of Certificateless Signature Schemes

	Key Generation			Signing			Verification		
	Exp	MtP	$\hat{e}(\cdot,\cdot)$	Exp	MtP	$\hat{e}(\cdot,\cdot)$	Exp	MtP	$\hat{e}(\cdot,\cdot)$
Cao *et al.* [8]	1	1	0	3	0	1	1	0	4
Huang *et al.* [16]	1	1	0	3	0	1	1	0	4
Li *et al.* [19]	1	1	0	2	0	0	1	0	4
Zhang *et al.*[27]	1	1	0	3	2	0	0	2	4
Liu *et al.*[18]	2	n	0	4	n	0	0	n	6
Our Proposed Scheme	1	0	0	2	0	0	4	0	0

Generic constructions of certificateless signature are proposed in [26] and [15]. ([26] is later shown to be insecure by [15]). Comparing our proposed scheme with the construction in [15], our scheme produces signatures of shorter length, as the signature in [15] is a concatenation of an IBS and a traditional signature.

4.4 Subtleties of the Public Component Selected by the KGC

In our scheme, there is a public component $W = g^w$ where w is randomly chosen by the KGC. W is public since it is in the final signature. No existing certificateless signature schemes have such component. Subtleties appear such that whether this element can be considered as part of the public key.

This component W appears is in the generation and extraction of the SEM key. It also appears in the `SEM-Sign` query to get signatures for some purported public key as well as in the forgery returned by the adversary.

For the SEM key generation, W is generated together with the SEM private key $d = w + sH_1(ID||W)$, which both are sent to the KGC. The KGC only

knows the correct value of d corresponding to the value of W given by the KGC. Since the KGC is responsible for checking whether the user's signing right has been revoked anyway, it is actually natural to assume the KGC will *not* sign for whatever purported W. In our security analysis, we also impose such restriction.

Similarly, no extraction of the SEM key according to an adversarially chosen W, since (W, d) is considered as a SEM key *as a whole*. However, considering the fact that the verifier does not hold a legitimate copy of W (that the SEM is holding), i.e. any verifier can only use the W included in the signature during the verification, then forgery with W *replaced* is considered as *valid*.

4.5 The Importance of the Key Binding

In order to achieve trust level 3, we can use the binding technique that ensures only one public key, for which the user knows the corresponding private key can be created. This technique was first employed in [1,26] in order to prevent the KGC from issuing two valid partial private keys (in certificateless context, or the SEM private keys in our terms) for a single user.

With the binding technique in place, the existence of two working public keys for an ID can only result from the existence of the SEM private keys binding that identity to two different public keys; only the KGC could have created these two SEM private keys since only the KGC has the master key.

This technique is useful when higher security level is required, especially in corporate and military environment. It is acceptable that only one permanent public key is available for one identity. When the user private key found compromised or the user is removed from his/her duty, the ID will be revoked permanently. Nevertheless, we note a drawback of the binding technique, which makes the user who holds ID can no longer use a new public key P'_{ID}.

If such permanent revocation is undesirable, when either the SEM private key or the user private key is compromised, some "out-of-band" non-cryptographic signature should be given by the party concerned, so no one can claim later that something malicious is happening even there exist two "working" public keys.

4.6 Security

Before the security proof, we review the definition of discrete logarithm problem.

Definition 3. ***Discrete Logarithm Problem (DLP).*** *The DLP is the problem of finding a given* (p, q, g, g^a) *with uniformly random choices of* $a \in Z_q^*$ *and* $g \in Z_p^*$. *The DL assumption states that there is no polynomial time algorithm with a non-negligible advantage in solving the DLP.*

The security proofs below borrow some proof ideas from [7]. The first one is to ensure certain random oracle responses to be the same in two executions of the adversary, irrespective of when the corresponding queries are made by the adversary. This is done by letting one oracle query to trigger the determination of a number of related oracle responses. Another one is the use of commitment to help the correct programming of the random oracle. This trick has been adopted elsewhere, for example, in Nicolosi *et al.*'s proactive two-party signatures [23].

Theorem 1. *Our SMC signature scheme is existential unforgeable against the Type I adversary in the random oracle model assuming the DLP in $\mathbb{G}$ is hard.*

Proof. Let $\mathcal{A}_{\mathcal{I}}$ be a forger that breaks the proposed signature scheme under adaptive chosen message and identity attack. We show that how $\mathcal{B}$ can use $\mathcal{A}_{\mathcal{I}}$ to solve the DLP instance (p, q, g, g^a).

$\mathcal{B}$ sets $Y = g^a$ and system parameters as (p, q, g, Y). $\mathcal{B}$ then gives the system parameters to $\mathcal{A}_{\mathcal{I}}$. It also maintains a list of tuples $(ID_i, W_i, e_i, d_i, P_i, x_i)$ which is denoted as H_1^{list}. Next, $\mathcal{B}$ randomly selects an index I such that $1 \leq I \leq q_{H_1}$, where q_{H_1} denotes the maximum number of queries to the random oracle H_1. $\mathcal{B}$ also picks $e_I \in Z_q^*$ at random and sets $W_I = g^a Y^{-e_I}$.

Adversary $\mathcal{B}$ interacts with $\mathcal{A}_{\mathcal{I}}$ in the `Attack` phase of the game as follows:

H_0 `Queries`: When $\mathcal{A}_{\mathcal{I}}$ queries H_0, $\mathcal{B}$ checks the corresponding H_0^{list} and outputs c_i if such query has already been made. Otherwise, $\mathcal{B}$ picks $c_i \in \{0,1\}^{\ell}$ at random, updates the H_0^{list} and outputs c_i as answer.

H_1 `Queries`: When $\mathcal{A}_{\mathcal{I}}$ queries H_1 on input $(ID_i||W_i)$, $\mathcal{B}$ checks the corresponding H_1^{list} and outputs e_i if such value is defined. Otherwise, $\mathcal{B}$ picks $e_i \in Z_q^*$ at random and outputs e_i as answer. H_1^{list} is also updated, such that $(ID_i, W_i, e_i, \perp, \tilde{\Delta}, \tilde{\Delta})$ is stored, where $\tilde{\Delta}$ means the old value (if exists) is kept.

H_2 `Queries`: When $\mathcal{A}_{\mathcal{I}}$ issues a query on these hash values, $\mathcal{B}$ parses the input as $(ID||\mu||b||\nu||R_i||m_i)$. If the query cannot be parsed as this form, simply returns a random value $h_i \in Z_q^*$. If $b = 0$, $\mathcal{B}$ sets $W_i = \mu$ and $P_i = \nu$. If $b = 1$, $\mathcal{B}$ sets $W_i = \nu$ and $P_i = \mu$. After that, $\mathcal{B}$ checks whether the value $H_1(ID||W_i)$ has been defined. If no, $\mathcal{B}$ runs the simulation of H_1 as usual to define it.

$\mathcal{B}$ then checks the corresponding H_2^{list}. If an entry for the query is found, the same answer will be given to $\mathcal{A}$. Otherwise, $\mathcal{B}$ randomly selects h_i and h_i' from Z_q^*, assigns $h_i = H_2(ID||\mu||0||\nu||R_i||m_i)$ and $h_i' = H_2(ID||\nu||1||\mu||R_i||m_i)$. By symmetry, $H_2(ID||\nu||1||\mu||R_i||m_i)$ must have not been defined beforehand if $H_2(ID||\mu||0||\nu||R_i||m_i)$ has not. Finally, all queries and the corresponding answer will be stored in the H_2^{list}.

`SEM-Extract`: Suppose the query is on ID_i.

1. If $i \neq I$, then $\mathcal{B}$ picks $d_i, e_i \in Z_q^*$ at random and computes $W_i = g^{d_i} Y^{-e_i}$. $\mathcal{B}$ updates the corresponding record in the H_1^{list} to $(ID_i, W_i, e_i, d_i, \tilde{\Delta}, \tilde{\Delta})$, where $\tilde{\Delta}$ means the old value is kept. (W_i, d_i) is returned as the answer.
2. Else if $i = I$, $\mathcal{B}$ aborts.

The above simulation is faithful since $W_i \cdot Y^{H_1(ID_i||W_i)} = g^{d_i} Y^{-e_i} Y^{e_i} = g^{d_i}$.

`User-Extract`: Suppose the query is on ID_i. Assume the public key for ID_i has not been replaced (no such query is allowed otherwise), $\mathcal{B}$ responds as follow.

1. $i \neq I$: If the public key does not exist, $\mathcal{B}$ picks $x_i \in Z_q^*$ at random and updates the H_1^{list} (i.e. stores x_i and g^{x_i}). Otherwise, returns the x_i stored.
2. $i = I$: $\mathcal{B}$ aborts.

`Replace`: Suppose the public key for ID_i is replaced with value P_i', $\mathcal{B}$ just updates H_1^{list} accordingly (i.e. stores $(P_i', \perp)$ for user public/private key).

`SEM-Sign`: Note that at any time during the simulation, equipped with those SEM private keys for any $ID_i \neq ID_I$, $\mathcal{A}_{\mathcal{I}}$ is able to generate partial signatures on any message if the public key had not been replaced. We assume that the $\mathcal{B}$ would not help the user to sign for any purported W value, instead, SEM uses the stored one. For $ID_i = ID_I$, assume that $\mathcal{A}_{\mathcal{I}}$ issues a query (m_i, P_I) where m_i denotes a message and P_I denotes a current public key chosen by $\mathcal{A}_{\mathcal{I}}$ that is associated with ID_I. Given $W_I = g^a Y^{-e_I}$ where $e_I = H_1(ID_I||W_I)$, $\mathcal{B}$ creates a partial signature as follows:

1. Pick $t_i, h_i \in Z_q^*$ at random and set $R_S = g^{t_i} g^{a(-h_i)}$.
2. Execute H_0 oracle simulation, get $c_i = H_0(R_s)$ and send it to $\mathcal{A}$.
3. After getting R_U from $\mathcal{A}$, set $h_i = H_2(ID_I||W_I||0||P_I||R_S \cdot R_U||m)$.
4. Send $\langle R_S, t_i \rangle$ to $\mathcal{A}$.

Embedding h_i as the response of H_2 is not possible if $\mathcal{A}_{\mathcal{I}}$ has queried the H_2 value of $(ID_I||W_I||0||P_I||R_S \cdot R_U||m)$ beforehand. We consider the case that $H_0(R_S)$ has previously queried and the case that it was not. In the first case, $\mathcal{A}_{\mathcal{I}}$ probably knows R_S and may have deliberately queried such value. However, since t_i is chosen randomly by $\mathcal{B}$ independent of $\mathcal{A}_{\mathcal{I}}$'s view, the probability that $\mathcal{A}_{\mathcal{I}}$ made such H_0 query is at most $(q_H + q_S)/2^k$. In the second case, the view of $\mathcal{A}_{\mathcal{I}}$ is completely independent of R_S. The probability that $R_S \cdot R_U$ appeared (by chance) in a previous H_2 query is against at most $(q_H + q_S)/2^k$.

`User-Sign`: Note that at any time during the simulation, if equipped with those user private keys for any ID, $\mathcal{A}_{\mathcal{I}}$ is able to generate partial signatures on any message. For replaced public key P_I, the simulation is as follows.

1. An ℓ-bit commitment c_I is obtained from the adversary, which models the first message that SEM should be sent to initiate the `User-Sign` process.
2. Pick $\upsilon_i, \xi_i \in Z_q^*$ at random and set $R_U = g^{\upsilon_i} {P_I}^{(-\xi_i)}$.
3. Search for H_0^{list} to find R_S' such that $H_0(R_S') = c_I$.
4. If found, set $\xi_i = H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m)$.
5. After obtained $\langle R_S, t_i \rangle$, check if $H_0(R_S) = c_I$, stop `User-Sign`.
6. If R_S' is previously found, but $R_S' \neq R_S$, declare failure and abort.
7. If R_S' was not found, and $H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m) \neq \xi_i$, declare failure and abort since $H_0(R_S) = c_I$ but $\mathcal{B}$ cannot program H_2 accordingly.
8. If $\mathcal{B}$ is lucky enough to reach this step, send $\langle R_U, \upsilon_i + t_i \rangle$ to $\mathcal{A}$.

The above simulation fails if $H_0(R_S) = c_I$, but no R_S' can be found or $R_S' \neq R_S$. For the first case, the probability that $\mathcal{A}_{\mathcal{I}}$ can predict $H_0(R_S) = c_I$ without asking the random oracle is at most $1/2^\ell$. For the second case, collision must have occurred and the probability for this is at most $((q_H + q_S)(q_H + q_S + 1)/2)/2^\ell \leq (q_H + q_S + 1)^2/2^\ell$. We just assume $\mathcal{A}$ asked for $H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m)$ if R_S' was not found since $\mathcal{A}$ knew the value of R_S before $\mathcal{B}$.

`Complete-Sign`: 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, but much easier since we no longer need the technicality to solve the problem that either one of the R_S and R_U is unknown.

`Forgery`: The next step of the simulation is to apply the general forking lemma in [7]: Let $\langle R^*, V^* \rangle$ be a forgery of a signature on message m^* with respect to $\langle ID^*, P_{ID^*}, W^* \rangle$ that is output by $\mathcal{A}_{\mathcal{I}}$ at the end of the attack. If $\mathcal{A}_{\mathcal{I}}$ does not output $ID^* = ID_I$ as a part of the forgery then $\mathcal{B}$ aborts (the probability that $\mathcal{B}$ does not abort the simulation is $O(1/q_{H_1})$).

Consider the case that W^* has not been replaced, i.e. $W^* = W_I = g^a Y^{-e_I}$ where $e_I = H_1(ID_I||W_I)$. $\mathcal{B}$ then replays $\mathcal{A}_{\mathcal{I}}$ with the same random tape but different H_2 after the point $(ID^*||W^*||0||P_{ID^*}||R^*||m^*)$. Suppose H_2 outputs h_S and h'_S in the first round and the second round respectively, where $h_S \neq h'_S$. Note the special step in the simulation of H_2 ensures $H_2(ID^*||P_{ID^*}||1||W^*||R^*||m^*)$ remains the same after forking. Moreover, since $e_I = H_1(ID^*||W^*)$ is defined at the very beginning of the game, it remains the same as well.

So we get another valid forgery $\langle R^*, V' \rangle$, i.e.

$$R^* = g^{V^*} P_{ID}{}^{-H_2(ID^*||P_{ID^*}||1||W^*||R^*||m^*)} (W^* \cdot Y^{H_1(ID^*||W^*)})^{-h_S}$$
$$R^* = g^{V'} P_{ID}{}^{-H_2(ID^*||P_{ID^*}||1||W^*||R^*||m^*)} (W^* \cdot Y^{H_1(ID^*||W^*)})^{-h'_S}$$

$\mathcal{B}$ thus gets $V^* - ah_S = V' - ah'_S$. DLP's solution is $a = (V^* - V')/(h_S - h'_S)$.

In the second case, W^* is replaced. We would like to apply forking lemma so that $H_1(ID^*||W^*)$ changes from h to h' after forking, but H_2 queries related to W^* remain the same. Since W^* never appears in `Sign` query of any kind, it is thus safe to rearrange all those H_2 queries before the forking, without affecting the adversary's view. After forking, we get another valid forgery $\langle R^*, V' \rangle$ where

$$R^* = g^{V^*} P_{ID}{}^{-H_2(ID^*||P_{ID^*}||1||W^*||R^*||m^*)} (W^* \cdot Y^{h})^{-H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)}$$
$$R^* = g^{V'} P_{ID}{}^{-H_2(ID^*||P_{ID^*}||1||W^*||R^*||m^*)} (W^* \cdot Y^{h'})^{-H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)}$$

$\mathcal{B}$ solves the DLP by $a = (V^* - V')/(H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)(h - h'))$.

Theorem 2. *Our SMC signature scheme is existential unforgeable against the Type II adversary in the random oracle model assuming the DLP in $\mathbb{G}$ is hard.*

Proof. Let $\mathcal{A}_{\mathcal{II}}$ be a forger that breaks our scheme under adaptive chosen message attack. We show how $\mathcal{B}$ can use $\mathcal{A}_{\mathcal{II}}$ to solve the DLP instance (p, q, g, g^a).

$\mathcal{B}$ first randomly picks $s \in Z_q^*$, gives the system parameters as $(p, q, g, Y = g^s)$ and the master key s to $\mathcal{A}_{\mathcal{II}}$. It also maintains a list of tuples $(ID_i, W_i, e_i, d_i, P_i, x_i)$ which is denoted as H_1^{list}. Next, $\mathcal{B}$ randomly selects an index I such that $1 \leq I \leq q_{H_1}$, where q_{H_1} denotes the maximum number of queries to the random oracle H_1. $\mathcal{B}$ sets $P_I = g^a$, picks $z_I, e_I \in Z_q^*$ at random and computes $W_I = g^{w_I}$ and $d_I = z_I + se_I$ where $e_I = H_1(ID_I||W_I)$.

Adversary $\mathcal{B}$ interacts with $\mathcal{A}_{\mathcal{II}}$ in the `Attack` phase of the game as follows:

H_0 `Queries`: When $\mathcal{A}_{\mathcal{II}}$ queries H_0, $\mathcal{B}$ checks the corresponding H_0^{list} and outputs c_i if such query has already been made. Otherwise, $\mathcal{B}$ picks $c_i \in \{0,1\}^{\ell}$ at random, updates the H_0^{list} and outputs c_i as answer.

H_1 `Queries`: When $\mathcal{A}_{\mathcal{II}}$ queries H_1 on input $(ID_i||W_i)$, $\mathcal{B}$ checks the corresponding H_1^{list} and outputs e_i if such value is defined. Otherwise, $\mathcal{B}$ picks $e_i \in Z_q^*$ at random and outputs e_i as answer. H_1^{list} is also updated, such that $(ID_i, W_i, e_i, \perp, \tilde{\Delta}, \tilde{\Delta})$ is stored, where $\tilde{\Delta}$ means the old value (if exists) is kept.

H_2 `Queries`: When $\mathcal{A}_{\mathcal{II}}$ issues a query on these hash values, $\mathcal{B}$ parses the input as $(ID||\mu||b||\nu||R_i||m_i)$. If the query cannot be parsed as this form, simply returns a random value $h_i \in Z_q^*$. If $b = 0$, $\mathcal{B}$ sets $W_i = \mu$ and $P_i = \nu$. If $b = 1$, $\mathcal{B}$ sets $W_i = \nu$ and $P_i = \mu$. After that, $\mathcal{B}$ checks whether the value $H_1(ID||W_i)$ has been defined. If no, $\mathcal{B}$ runs the simulation of H_1 as usual to define it.

$\mathcal{B}$ then checks the corresponding H_2^{list}. If an entry for the query is found, the same answer will be given to $\mathcal{A}_{\mathcal{II}}$. Otherwise, $\mathcal{B}$ randomly selects h_i and h_i' from Z_q^*, assigns $h_i = H_2(ID||\mu||0||\nu||R_i||m_i)$ and $h_i' = H_2(ID||\nu||1||\mu||R_i||m_i)$. By symmetry, $H_2(ID||\nu||1||\mu||R_i||m_i)$ must have not been defined beforehand if $H_2(ID||\mu||0||\nu||R_i||m_i)$ has not. Finally, all queries and the corresponding answer will be stored in the H_2^{list}.

`SEM-Extract`: Note that at any time during the simulation, equipped with the master key s, $\mathcal{A}_{\mathcal{II}}$ is able to generate SEM private key for any ID.

`User-Extract`: Suppose the query is on ID_i. $\mathcal{B}$ responds as follow.

1. $i \neq I$: If the public key does not exist, $\mathcal{B}$ picks $x_i \in Z_q^*$ at random and updates the H_1^{list} (i.e. stores x_i and g^{x_i}). Otherwise, returns the x_i stored.
2. $i = I$: $\mathcal{B}$ aborts.

`SEM-Sign`: Note that at any time during the simulation, equipped with the master key s, $\mathcal{A}_{\mathcal{II}}$ is able to generate partial signatures $\langle R_S, t_i \rangle$ on any message.

`User-Sign`: Note that at any time during the simulation, equipped with those user private keys for any $ID_i \neq ID_I$, $\mathcal{A}_{\mathcal{II}}$ is able to generate partial signatures on any message. For $ID_i = ID_I$, the simulation is as follows.

1. An ℓ-bit commitment c_I is obtained from the adversary, which models the first message that SEM should be sent to initiate the `User-Sign` process.
2. Pick $\upsilon_i, \xi_i \in Z_q^*$ at random and set $R_U = g^{\upsilon_i} P_I{}^{(-\xi_i)}$.
3. Search for H_0^{list} to find R_S' such that $H_0(R_S') = c_I$.
4. If found, set $\xi_i = H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m)$.
5. After obtained $\langle R_S, t_i \rangle$, check if $H_0(R_S) = c_I$, stop `User-Sign`.
6. If R_S' is previously found, but $R_S' \neq R_S$, declare failure and abort.
7. If R_S' was not found, and $H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m) \neq \xi_i$, declare failure and abort since $H_0(R_S) = c_I$ but $\mathcal{B}$ cannot program H_2 accordingly.
8. If $\mathcal{B}$ is lucky enough to reach this step, send $\langle R_U, \upsilon_i + t_i \rangle$ to $\mathcal{A}$.

The above simulation fails if $H_0(R_S) = c_I$, but no R'_S can be found or $R'_S \neq R_S$. For the first case, the probability that $\mathcal{A}_{\mathcal{I}}$ can predict $H_0(R_S) = c_I$ without asking the random oracle is at most $1/2^\ell$. For the second case, collision must have occurred and the probability for this is at most $((q_H + q_S)(q_H + q_S + 1)/2)/2^\ell \leq (q_H + q_S + 1)^2/2^\ell$. We just assume $\mathcal{A}$ asked for $H_2(ID_I||P_I||1||W_I||R_U \cdot R_S||m)$ if R'_S was not found since $\mathcal{A}$ knew the value of R_S before $\mathcal{B}$.

`Complete-Sign`: If the user private key is available, signing is trivial. Otherwise, this request can be simulated faithfully similar to above, but much easier since we no longer need the technicality to solve the problem that R_S is unknown.

`Forgery`: The next step of the simulation is to apply the general forking lemma in [7]: Let $\langle R^*, V^* \rangle$ be a forgery of a signature on message m^* with respect to $\langle ID^*, P_{ID^*}, W^* \rangle$ that is output by $\mathcal{A}_{\mathcal{II}}$ at the end of the attack. If $\mathcal{A}_{\mathcal{II}}$ does not output $ID^* = ID_I$ as a part of the forgery then $\mathcal{B}$ aborts (the probability that $\mathcal{B}$ does not abort the simulation is $O(1/q_{H_1})$).

$\mathcal{B}$ then replays $\mathcal{A}_{\mathcal{II}}$ with the same random tape but different H_2 after the point $(ID^*||P_{ID^*}||1||W^*||R^*||m^*)$. Suppose H_2 outputs h_U and h'_U in the first round and the second round respectively, where $h_U \neq h'_U$. Since $e_I = H_1(ID^*||W^*)$ and $H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)$ are defined before $H_2(ID^*||P_{ID^*}$ $||1||W^*||R^*||m^*)$ outputs h_U and h'_U, they remain the same as well.

So we get another valid forgery $\langle R^*, V' \rangle$, i.e.

$$R^* = g^{V^*} {P_{ID}}^{-h_U} (W^* \cdot Y^{H_1(ID^*||W^*)})^{-H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)}$$
$$R^* = g^{V'} {P_{ID}}^{-h'_U} (W^* \cdot Y^{H_1(ID^*||W^*)})^{-H_2(ID^*||W^*||0||P_{ID^*}||R^*||m^*)}$$

Since $P_{ID} = g^a$, $\mathcal{B}$ thus gets $V^* - ah_U = V' - ah'_U$. DLP's solution is $a = (V^* - V')/(h_U - h'_U)$. It works even the discrete logarithm of W^* is unknown.

5 Conclusion

We presented a formal study of security mediated certificateless signatures (SMC signatures). We showed that Ju *et al.*'s construction [17] is insecure. We formalize the security model of SMC signature, and discussed the subtleties arose from a public component introduced by the probabilistic SEM private key generation. We then proposed the first provable secure SMC signature scheme. Our scheme is efficient in the sense that no bilinear pairing is involved. It is provable secure in the random oracle model based on the intractability of the discrete logarithm problem. We also extended our scheme to achieve trust level 3 by adopting the technique used in [1,26]. It is worth noting that our scheme can be easily extended to be an efficient CLS scheme without pairing. Distributing the power of the SEM by threshold signature technique [25] will be our future work.

Acknowledgement

Sherman Chow is grateful for the comments of Alexander W. Dent, Joonsang Baek, Guilin Wang and Guomin Yang. This research is supported by the Malaysia IRPA grant (04-99-01-00003-EAR) and e-Science fund (01-02-01-SF0032).

References

1. Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless Public Key Cryptography. In *Proceedings of Asiacrypt 2003*, LNCS 2894, pp. 452-473. Full version available at Cryptology ePrint Archive, 2003/126.
2. Joonsang Baek, Reihaneh Safavi-Naini and Willy Susilo. Certificateless Public Key Encryption Without Pairing. In *Proc. of ISC 2005*, LNCS 3650, pp. 134-148.
3. Joonsang Baek and Yuliang Zheng. Identity-based Threshold Decryption. *Proc. of PKC 2004*, LNCS 2947, pp. 262–276.
4. Dan Boneh, Xuhua Ding and Gene Tsudik. Fine-Grained Control of Security Capabilities. *ACM Transactions on Internet Technology 2004*, 4/1, pp. 60-82.
5. Dan Boneh, Xuhua Ding, Gene Tsudik and Chi-Ming Wong. A Method for Fast Revocation of Public Key Certificates and Security Capabilities. In *USENIX 2001*.
6. Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal of Computing*, 32/3, pp. 586–615, 2003.
7. Mihir Bellare and Gregory Neven. Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma In *Proceedings of ACM-CCS 2006*, pp. 390-399.
8. Xuefei Cao, Kenneth G. Paterson and Weidong Kou. An Attack on a Certificateless Signature Scheme. Cryptology ePrint Archive, Report 2006/367.
9. Sherman S.M. Chow, Colin Boyd and Juan Manuel González Nieto. Security Mediated Certificateless Cryptography. In *Proceedings of PKC 2006*, LNCS 3958, pp. 508-524.
10. Alexander W. Dent. Personal communication, April 26, 2006.
11. Xuhua Ding and Gene Tsudik. Simple Identity-Based Cryptography with Mediated RSA. In *Proceedings of CT-RSA 2003*, LNCS 2612, pp. 193-210.
12. Marc Girault. Self-Certified Public Keys. In *Proceedings of Eurocrypt 1991*, LNCS 547, pp. 490-497.
13. M. Choudary Gorantla and Ashutosh Saxena. An Efficient Certificateless Signature Scheme. In *Computational Intelligence and Security 2005*, LNAI 3802, pp. 110-116.
14. Florian Hess. Efficient Identity Based Signature Schemes based on Pairings. In *Proceedings of SAC 2002*, LNCS 2595, pp. 310-324.
15. Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang and Xiaotie Deng. Key Replacement Attack Against a Generic Construction of Certificateless Signature. In *Proceedings of ACISP 2006*, LNCS 4058, pp. 235-246.
16. Xinyi Huang, Willy Susilo, Yi Mu and Futai Zhang. On the Security of Certificateless Signature Schemes from Asiacrypt 2003. In *Proceedings of CANS 2005*, LNCS 3810, pp. 13-25.
17. Hak Soo Ju, Dae Youb Kim, Dong Hoon Lee, Jongin Lim and Kilsoo 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.
18. Joseph K. Liu, Man Ho Au and Willy Susilo. Self-Generated-Certificate Public Key Cryptography and Certificateless Signature / Encryption Scheme in the Standard Model. In *Proceedings of ASIACCS 2007*.

19. X. Li, K. Chen and L. Sun. Certificateless Signature and Proxy Signature Schemes from Bilinear Pairings. *Lithuanian Mathematical Journal*, 45/1, pp. 76–83, 2005.
20. Benoît Libert and Jean-Jacques Quisquater. Efficient Revocation and Threshold Pairing Based Cryptosystems. In *Proceedings of PODC 2003*, pp. 163-171.
21. Silvio Micali. Novomodo: Scalable Certificate Validation and Simplified PKI Management. In *Proceedings of 1st Annual PKI Research Workshop 2002*, pp. 15-25.
22. M. Myers, R. Ankney, A. alpani, S. Galperin and C. Adams. X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol (OCSP), *RFC 2560*.
23. Antonio Nicolosi, Maxwell Krohn, Yevgeniy Dodis, David Mazières. Proactive Two-Party Signatures for User Authentication. In *Proceedings of 10th NDSI 2003*.
24. Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In *Proceedings of CRYPTO 1989*, LNCS 435, pp. 239-252.
25. Jun Shao, Zhenfu Cao, and Licheng Wang. Efficient ID-based Threshold Signature Schemes without Pairings. Cryptology ePrint Archive, Report 2006/308.
26. Dae Hyun Yum and Pil Joong Lee. Generic Construction of Certificateless Signature. In *Proceedings of ACISP 2004*, LNCS 3108, pp. 200-211.
27. Zhenfeng Zhang, Duncan S. Wong, Jing Xu and Dengguo 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.

A Chow *et al.* vs. Ju *et al.*

Chow *et al.* [9] provided two constructions of encryption schemes: a generic construction in the standard model (which is earlier than [18]), with the restriction that decryption may not work after public key replacement, and a concrete scheme assuming random oracles, with the aforementioned restriction removed.

Ju *et al.*'s [17] security model for the encryption scheme does not allow the adversary to ask for the private key of the target user (*weak semantic security against insider attacks* in the term of [20]). Obviously, such assumption is too strong. They suggested to strengthen their scheme by using the technique of [3], but Chow *et al.*'s scheme [9] took a step further with a solution more elegant and efficient than such modification. Moreover, it is easy to show the hierarchical variant in [17] is insecure against chosen ciphertext attack.

B Errata of Chow *et al.*'s Generic SMC Encryption

Below give two errata of the generic SMC encryption in [9]. Both are about the encryption algorithm. Decryption algorithms should be revised accordingly.

1. Instead of $\alpha = H(C_1, C_2, \ell)$, α should be $H(C_1, C_2, \ell, s_1)$; otherwise, a chosen ciphertext attack is possible [10].
2. Instead of $(C_1, C_2) = (\mathsf{IBE.Enc}_{\mathsf{params}}(\mathsf{ID}_A, s_1), C_2 = \mathsf{PKE.Enc}^{\ell}_{\mathsf{EK}}(s_2))$, it should be $(C_1, C_2) = (\mathsf{IBE.Enc}_{\mathsf{params}}(\mathsf{ID}_A, s_1 || H(\mathsf{VK})), C_2 = \mathsf{PKE.Enc}^{\mathsf{VK}}_{\mathsf{EK}}(s_2))$, i.e. the IBE encryption also encrypts the hash value of VK, and the label as the input of the public key encryption should be VK instead of the label ℓ of the SMC encryption. This amendment has been made in the PKC '06 presentation. Without such change, a chosen ciphertext attack is possible.