1. Introduction

Public key cryptography is an important technique to realize network security. In public key cryptography, each user has a public key and a private key. The public key is published and publicly accessible while the corresponding private key is kept secret by its owner. However, the public key in traditional public key cryptography is generated with no connection to its owner’s identity. Therefore, a public key infrastructure (PKI) is employed for vouching the relationship between an identity and a public key by the public key certificates. But, the traditional PKI technology is faced with many challenges in practice, especially the complicated certificate management problem. In 1984, Shamir [17] introduced the concept of identity-based cryptography to solve the certificate management problem in traditional public key cryptography. In identity-based cryptography, a user’s public key could be an arbitrary string related to his identity and his private key is computed from his identity by a trusted authority called private key generator (PKG). The merit of identity-based cryptography lies in that it eliminates the need for public key certificates. However, identity-based cryptography inevitably suffers from the key escrow problem since all users’ private keys are known to the PKG. To solve
the key escrow problem in identity-based cryptography while eliminating the use of public key certificates in traditional public key cryptography, Al-Riyami and Paterson [1] introduced the notion of certificateless public key cryptography in 2003. In certificateless public key cryptography, a trusted third party called key generation center (KGC) is employed for generating a partial private key for every user. Each user independently generates a pair of secret key and public key, and then combines his secret key with the partial private key from the KGC to form his full private key. As KGC does not know any user’s private key, certificateless public key cryptography overcomes the key escrow problem inherent in identity-based cryptography. Furthermore, certificateless public key cryptography provides an effective implicit authentication mechanism so that each user does not need to obtain a certificate from the certificate authority for the authenticity of his public key. Since its advent, certificateless public key cryptography has attracted much attention in the research community and numerous schemes have been proposed, including many encryption schemes (e.g. [2, 5, 13, 14, 18, 27]) and signature schemes (e.g. [4, 12, 14, 19, 20, 29, 30]).

The security of public key cryptography depends on the assumption that the private keys are kept perfectly secure. However, as cryptographic operations are performed more frequently on the unprotected and insecure devices, key exposure seems to be inevitable. Actually, it is much easier for an adversary to intrude a naive user’s device to obtain his private key than to break the computational assumption(s) on which a public key cryptosystem is based. Undoubtedly, key exposure has become the most devastating attack on public key cryptography, since it means all security guarantees are lost. To fight against key exposure, Dodis et al. [6] proposed the notion of key-insulated security in 2002. Instead of trying to eliminate key exposure entirely, their approach assumes that key exposure will occur and seek to minimize the resulting damage. In the model of key-insulated public key cryptosystems proposed by Dodis et al., a user can periodically update his private key with the help of a physically-secure device called helper while maintaining a same public key. The basic idea is that the whole lifetime of the system (e.g., one year) is divided into $N$ time periods (e.g., one day). Each user begins by generating a public key, a long-time helper key and an initial temporary secret key. He publishes the public key and then respectively stores the helper key in a helper and the initial temporary secret key in a user device where the cryptographic computations are actually carried out. At the beginning of each time period, the user device interacts with the helper to derive a new temporary secret key which will be used in that time period. Meanwhile, the public key remains unchanged throughout the lifetime of the system. It is clear that the helper plays a key role in protecting the user device against key exposure. As the helper is physically secure and resistant to compromise, exposure of the temporary secret keys at some time periods will not enable an adversary to derive the temporary secret keys for the remaining time periods. Thus, the damage caused by temporary secret key exposure can be effectively mitigated. In 2003, Dodis et al. [7] further proposed the notion of strong key-insulated security. This security notion guarantees that the helper is unable to compromise the corresponding user’s temporary secret key for any time period. Obviously, it is an important property for the environments where the helper serves several different users or the helper is untrustworthy.

Following Dodis et al.’s pioneering works [6, 7], a number of key-insulated encryption schemes (e.g. [3, 9, 10, 11, 16, 26]) and key-insulated signature schemes
Certificateless key-insulated signatures have been proposed. However, most of the prior key-insulated cryptographic schemes are constructed based on either traditional public key cryptography or identity-based cryptography. So, they inevitably suffer from either the cumbersome certificate management problem or the key escrow problem.

To alleviate these problems, Wan et al. [22] introduced key-insulated security into certificateless public key cryptography and proposed a certificateless key-insulated signature scheme without random oracles. Subsequently, Wan et al. [23] proposed a certificateless strong key-insulated signature scheme without random oracles. To the best of our knowledge, there exist only two certificateless key-insulated signature schemes in the literature so far.

**Our Contributions.** In this paper, we first show that the existing two certificateless signature schemes [22, 23] do not achieve the existential unforgeability. More concretely, both of them are insecure under the attacks from either a key replacement adversary or a malicious KGC. So, it is fair to say that devising a secure certificateless (strong) key-insulated signature scheme remains an unsolved problem until now. To overcome the security flaws in the existing certificateless key-insulated signature schemes, we propose a new certificateless strong key-insulated signature scheme which is proven to be existentially unforgeable under the square computational Diffie-Hellman assumption in the standard model.

Compared with the certificateless strong key-insulated signature scheme proposed by Wan et al. in [23], our new scheme has three advantages. Firstly, it remedies the security flaws existing in Wan et al.’s scheme due to the redesigned user key generation and signature algorithms. Secondly, it is proven secure under a harder computational assumption. Finally, it enjoys shorter public system parameters and lower computational cost.

## 2. Preliminaries

### 2.1. Bilinear pairing and computational assumption. Let $G_1$ and $G_2$ denote two cyclic groups of the same prime order $p$ and $g$ be a generator of $G_1$. A map $e : G_1 \times G_1 \rightarrow G_2$ is said to be a bilinear paring if it satisfies the following properties:

- Bilinearity: $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$;
- Non-degeneracy: $e(g, g) \neq 1_{G_2}$;
- Computability: $e(u, v)$ can be efficiently computed for all $u, v \in G_1$.

The security of the scheme proposed in this paper relies on the following square computational Diffie-Hellman (Squ-CDH) assumption [29].

**Definition 2.1.** Given a group $G_1$ of prime order $p$ with generator $g$ and $g^a \in G_1$ for unknown $a \in \mathbb{Z}_p^*$, the Squ-CDH problem in $G_1$ is to compute $g^{a^2} \in G_1$. The advantage of a probabilistic polynomial time (PPT) algorithm $A$ in solving the Squ-CDH problem is defined to be $Adv^{Squ-CDH}_A(k) = Pr[A(g, g^a) = g^{a^2}]$.

We say that the Squ-CDH assumption holds in $G_1$ if for any PPT algorithm $A$, the advantage $Adv^{Squ-CDH}_A(k)$ is negligible.

In [29], the Squ-CDH assumption has been proved to be equivalent to the standard computational Diffie-Hellman (CDH) assumption.

### 2.2. Formal model of certificateless key-insulated signature. A certificateless key-insulated signature scheme can be specified by the following eight polynomial time algorithms [22, 23]:

Advances in Mathematics of Communications Volume 9, No. 3 (2015), 353–373
(1) **Setup**($k, N$): This algorithm, performed by KGC, takes a security parameter $k$ and the total number of time periods $N$ as input to generate a master secret key $m$s$k$ and a list of public parameters $params$.

(2) **PartialKeyGen**($params$, $m$s$k$, $ID$): This algorithm, performed by KGC, takes the public parameters $params$, the master key $m$s$k$ and a user’s identity $ID$ as input to generate a partial private key $psk_{ID}$.

(3) **UserKeyGen**($params$, $ID$): This algorithm, performed by a user, takes the public parameters $params$ and a user’s identity $ID$ as input to generate a public key $upk_{ID}$ and a private key $usk_{ID}$.

(4) **KeyInitialize**($params$, $psk_{ID}$, $usk_{ID}$): This algorithm, performed by a user, takes the public parameters $params$, a user’s partial private key $psk_{ID}$ and private key $usk_{ID}$ as input to generate a user’s initial temporary signing key $TSK_{ID,0}$ and a helper’s key $HK_{ID}$.

(5) **KeyUpdateH**($params$, $ID$, $HK_{ID}$, $t_{new}$, $t_{old}$): This algorithm, performed by a helper, takes the public parameters $params$, a user’s identity $ID$, a helper’s key $HK_{ID}$ and two time period indices $t_{new}$ and $t_{old}$ ($0 \leq t_{old} \leq t_{new} \leq N - 1$) as input to generate an update key $UK_{ID,T_{old},T_{new}}$. The update key $UK_{ID,T_{old},T_{new}}$ is sent to the user with identity $ID$ via a secure channel and used to update this user’s temporary signing key from the time period $t_{old}$ to the time period $t_{new}$.

(6) **KeyUpdateU**($params$, $ID$, $TSK_{ID,T_{old}}$, $UK_{ID,T_{old},T_{new}}$): This algorithm, performed by a user, takes the public parameters $params$, a user’s identity $ID$, a temporary signing key $TSK_{ID,T_{old}}$ and an update key $UK_{ID,T_{old},T_{new}}$ as input to generate a temporary signing key $TSK_{ID,T_{new}}$ for the time period $t_{new}$.

(7) **Sign**($params$, $t$, $m$, $ID$, $TSK_{ID,t}$): This algorithm, performed by a signer, takes the public parameters $params$, a time period index $t$ ($0 \leq t \leq N - 1$), a message $m$, a signer’s identity $ID$ and temporary signing key $TSK_{ID,t}$ as input and returns a pair $(t, \sigma)$ composed of the time period index $t$ and a signature $\sigma$ of the message $m$.

(8) **Verify**($params$, $m$, $(t, \sigma)$, $ID$, $upk_{ID}$): This algorithm, performed by a verifier, takes the public parameters $params$, a message $m$, a candidate pair of time period index and signature $(t, \sigma)$ and the signer’s identity $ID$ and public key $upk_{ID}$ as input and returns 1 if $(t, \sigma)$ is a valid signature for the message $m$ under the identity $ID$ and the public key $upk_{ID}$ or 0 otherwise.

For correctness, it is required that if $(t, \sigma) = \text{Sign}(params, t, m, ID, TSK_{ID,t})$, then $\text{Verify}(params, m, (t, \sigma), ID, upk_{ID}) = 1$.

As introduced by Al-Riyami and Paterson in [1], a certificateless cryptographic scheme should be secure against two different types of adversaries, i.e. Type-I adversary (denoted by $A_1$) and Type-II adversary (denoted by $A_2$). The adversary $A_1$ simulates an outsider adversary who does not know the master key, but can replace public keys at its will. The adversary $A_2$ simulates a malicious KGC who knows the master key and controls the generation of partial private keys for users, but is not allowed to replace public keys.

To formalize the security notions of key-insulation and strong key-insulation for certificateless key-insulated signature schemes, we first describe the following seven oracles which are used to simulate potential attacking actions. A Type-I or Type-II adversary can adaptively make requests to some of these oracles. We assume that the challenger keeps a history of “query-answer” when interacting with the adversary.

**Advances in Mathematics of Communications** Volume 9, No. 3 (2015), 353–373
(1) \(O^\text{CreateUser}\): On input an identity \(ID\), the challenger responds with a public key \(upk_{ID}\). If the identity \(ID\) has not been created, then the challenger generates a public key \(upk_{ID}\) and a private key \(usk_{ID}\) respectively and returns \(upk_{ID}\). In this case, the identity \(ID\) is said to be created. Note that other oracles defined below only respond to an identity which has been created.

(2) \(O^\text{PrivateKey}\): On receiving a private key query on an identity \(ID\), the challenger responds with a private key \(usk_{ID}\).

(3) \(O^\text{PartialPrivateKey}\): On receiving a partial private key query on an identity \(ID\), the challenger responds with a partial private key \(psk_{ID}\). Note that such an oracle is only queried by Type-I adversary \(A_1\) since Type-II adversary \(A_2\) can generate any user’s partial private key by itself.

(4) \(O^\text{ReplacePublicKey}\): On input an identity \(ID\) and a pair of values \((upk'_{ID}, usk'_{ID})\), the challenger replaces the current public key and private key \((upk_{ID}, usk_{ID})\) associated with the identity \(ID\) with \((upk'_{ID}, usk'_{ID})\). Note that such an oracle is only queried by Type-I adversary \(A_1\) since Type-II adversary \(A_2\) is disallowed to replace any user’s public key.

(5) \(O^\text{HelperKey}\): On receiving a helper key query on an identity \(ID\), the challenger responds with a helper key \(HK_{ID}\).

(6) \(O^\text{TempSigningKey}\): On receiving a temporary signing key query on an identity \(ID\) and a period time index \(t\) \((0 \leq t \leq N - 1)\), the challenger responds with a temporary signing key \(TSK_{ID,t}\).

(7) \(O^\text{Sign}\): On receiving a signing query on an identity \(ID\), a period time index \(t\) \((0 \leq t \leq N - 1)\) and a message \(m\), the challenger responds with a signature \((t, \sigma)\).

The security of certificateless key-insulated signature schemes can be formally defined through two adversarial games \textbf{Game-A} and \textbf{Game-B}, which are played between a challenger and a Type-I or Type-II adversary \(A\). The former is used to define the key-insulated security while the latter is used to define the strong key-insulated security. \textbf{Game-A} models the normal key exposure attacking scenario, in which the adversary \(A\) does not know any user’s helper key, but can compromise any user’s temporary signing key at some time periods. \textbf{Game-B} simulates the scenario that the helper is untrustworthy, in which the adversary \(A\) can access any user’s helper key, but does not know the target user’s temporary signing key at any time period. Let \(ID^*\) and \(t^*\) \((0 \leq t^* \leq N - 1)\) denote the adversary’s target identity and target time period respectively. These two games are formally described as follows:

\textbf{Game-A}: In this game, the adversary \(A\) is disallowed to query the oracle \(O^\text{HelperKey}\), but can adaptively query the oracle \(O^\text{TempSigningKey}\) on any \((ID, t)\) except \((ID^*, t^*)\).

1. \textbf{Setup}. On input a security parameter \(k\) and a total number of the time periods \(N\), the challenger runs the algorithm \textbf{Setup}(\(k, N\)) to generate a master secret key \(msk\) and a list of public parameters \(params\). Then, the challenger sends \(params\) to the adversary \(A\). If \(A\) is a Type-II adversary, the challenger also sends the master secret key \(msk\) to it.

2. \textbf{Queries}. In this phase, the adversary \(A\) can adaptively make requests to the oracles \{\(O^\text{CreateUser}, O^\text{PrivateKey}, O^\text{PartialPrivateKey}, O^\text{ReplacePublicKey}, O^\text{TempSigningKey}, O^\text{Sign}\)\} if it is a Type-I adversary or the oracles \{\(O^\text{CreateUser}, O^\text{PrivateKey}, O^\text{TempSigningKey}, O^\text{Sign}\)\} if it is a Type-II adversary. The challenger responds as described above.

3. \textbf{Forgery}. Finally, the adversary \(A\) outputs a forgery \((ID^*, m^*, (t^*, \sigma^*))\) and wins the game if the following conditions are satisfied:
- Verify(params, m*, (t*, σ*), ID*, upk_{ID*}) = 1, that is, (t*, σ*) is a valid signature for the message m* under the target identity ID* and public key upk_{ID*};
- \( \mathcal{A} \) has never queried \( O_{\text{TempSigningKey}} \) on \((ID^*, t^*)\);
- \( \mathcal{A} \) has never queried \( O_{\text{Sign}} \) on \((ID^*, t^*, m^*)\);
- If \( \mathcal{A} \) is a Type-I adversary, it has never queried \( O_{\text{PartialPrivateKey}} \) on \( ID^* \);
- If \( \mathcal{A} \) is a Type-II adversary, it has never queried \( O_{\text{PrivateKey}} \) on \( ID^* \).

**Game-A:** In this game, the adversary \( \mathcal{A} \) can adaptively query \( O_{\text{HelperKey}} \) on any identity \( ID \) (including the target identity \( ID^* \)), but is disallowed to query \( O_{\text{TempSigningKey}} \) on the target identity \( ID^* \) and any time period \( t \) \((0 \leq t \leq N - 1)\).

1. **Setup.** This phase is same as **Game-A**.
2. **Queries.** This phase is same as **Game-A** except that the adversary \( \mathcal{A} \) can adaptively query \( O_{\text{HelperKey}} \) on any identity \( ID \).
3. **Forgery.** Finally, the adversary \( \mathcal{A} \) outputs a forgery \((ID^*, m^*, (t^*, \sigma^*))\) and wins the game if the following conditions are satisfied:
   - Verify(params, m*, (t*, \sigma*), ID*, upk_{ID*}) = 1, that is, (t*, \sigma*) is a valid signature for the message m* under the target identity ID* and public key upk_{ID*};
   - \( \mathcal{A} \) has never queried \( O_{\text{TempSigningKey}} \) on \( ID^* \) and any time period \( t \) \((0 \leq t \leq N - 1)\);
   - \( \mathcal{A} \) has never queried \( O_{\text{Sign}} \) on \((ID^*, t^*, m^*)\);
   - If \( \mathcal{A} \) is a Type-I adversary, it has never queried \( O_{\text{PartialPrivateKey}} \) on \( ID^* \);
   - If \( \mathcal{A} \) is a Type-II adversary, it has never queried \( O_{\text{PrivateKey}} \) on \( ID^* \).

In both above two games, the advantage of the adversary \( \mathcal{A} \) is defined to be the probability of producing a valid forgery.

**Definition 2.2.** A certificateless key-insulated signature scheme is said to be key-insulated and existentially unforgeable under chosen-message attacks (or KI&EUF-CMA secure) if the advantage of any PPT adversary \( \mathcal{A} \) to win the game **Game-A** is negligible.

**Definition 2.3.** If a certificateless key-insulated signature scheme is KI&EUF-CMA secure, then it is said to be strong key-insulated and existentially unforgeable under chosen-message attacks (or SKI&EUF-CMA secure) if the advantage of any PPT adversary \( \mathcal{A} \) to win the game **Game-B** is negligible.

As in [7], we consider attacks in which an adversary \( \mathcal{A} \) compromises the user ID’s storage while a temporary signing key is being updated from \( TSK_{ID,t_{old}} \) to \( TSK_{ID,t_{new}} \). We call this a key-update exposure at \((ID, t_{old}, t_{new})\). When this occurs, the adversary \( \mathcal{A} \) receives \( TSK_{ID,t_{old}} \) \( UK_{ID,t_{old},t_{new}} \) and \( TSK_{ID,t_{new}} \) (actually, the latter can be derived from the former). We say that a certificateless (strong) key-insulated signature scheme has secure key updates if a key-update exposure at \((ID, t_{old}, t_{new})\) is of no more help to the adversary \( \mathcal{A} \) than the temporary signing key exposures at both \((ID, t_{old})\) and \((ID, t_{new})\).

**Definition 2.4.** A certificateless key-insulated signature scheme has secure key updates if the view of any adversary \( \mathcal{A} \) making a key-update exposure at \((ID, t_{old}, t_{new})\) can be perfectly simulated by the adversary \( \mathcal{A} \)’s queries to the oracle \( O_{\text{TempSigningKey}} \) on both \((ID, t_{old})\) and \((ID, t_{new})\).
3. CRYPTOANALYSIS OF THE PREVIOUS CERTIFICATELESS KEY-INSULATED SIGNATURE SCHEMES

In this section, we show that the previous certificateless key-insulated signature schemes [22, 23] do not achieve the existential unforgeability. Considering that both of them were essentially constructed from Waters’ identity-based signature scheme [24] and share similarities in the signature algorithm, we present our attacks using the certificateless strong key-insulated signature scheme proposed by Wan et al. in [23] as an example. We show that it is not existentially unforgeable under either key replacement attack or malicious KGC attack. In addition, we briefly illustrate that the certificateless key-insulated signature scheme proposed by Wan et al. in [22] is insecure under our presented malicious KGC attack too.

3.1. REVIEW OF WAN ET AL.’S CERTIFICATELESS STRONG KEY-INSULATED SIGNATURE SCHEME. Wan et al.’s certificateless strong key-insulated signature scheme [23] is described as follows:

(1) Setup($k, N$): Let $G_1$ and $G_2$ be two cyclic groups of $k$-bit prime order $p$, $g$ be a generator of $G_1$ and $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map. Also let $H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^{n_1}$, $H_2 : \{0, 1\}^* \rightarrow \{0, 1\}^{n_2}$ and $F : \{0, 1\}^* \rightarrow Z_p^*$ be three hash functions, where $n_1, n_2 \in Z^+$. The KGC randomly chooses an integer $\alpha \in Z_p$ and computes $g_1 = g^\alpha$. Moreover, it randomly chooses an element $g_0 \in G_1$, two vectors $U = (u_0, u_1, \cdots, u_{n_1}) \in G_1^{n_1+1}$ and $M = (m_0, m_1, \cdots, m_{n_2}) \in G_1^{n_2+1}$. Define two functions $f_1(W) = u_0 \prod_{i \in W} u_i$ and $f_2(T) = m_0 \prod_{i \in T} m_i$, where $W \subseteq \{1, 2, \cdots, n_1\}$ and $T \subseteq \{1, 2, \cdots, n_2\}$. The public parameters $\text{params}$ are $\{N, G_1, G_2, e, p, g, g_1, g_0, U, M, f_1, f_2, H_1, H_2, F\}$ and the master key $\text{msk}$ is $g_0^2$.

(2) PartialKeyGen($\text{params}, \text{msk}, ID$): The KGC randomly picks $r_u \in Z_p^*$ and computes a partial private key for the user with identity $ID$ as

$$psk_{ID} = (psk_{ID,1}, psk_{ID,2}) = (g_0^{\alpha} \cdot f_1^*(P_{ID}), g^r),$$

where $P_{ID} = \{i|w_2[i] = 1, w_2 = H_1(ID)\}$.

(3) UserKeyGen($\text{params}, ID$): A user with identity $ID$ randomly chooses $x \in Z_p^*$ as his private key $usk_{ID}$ and computes his public key $upk_{ID} = e(g_1, g_2)^x$.

(4) KeyInitialize($\text{params}, psk_{ID}, usk_{ID}$): A user with identity $ID$ randomly chooses his helper’s key $HK_{ID} \in \{0, 1\}^k$ and computes $r_0 = F(ID, HK_{ID}, 0)$. He then randomly chooses $r \in Z_p^*$ and computes his initial temporary signing key as

$$T_{SK_{ID,0}} = (T_{ID,0}, R_{ID,0}, R) = ((psk_{ID,1})^x \cdot f_1^*(P_{ID}) \cdot f_1^{r_0}(P_{ID,0}), g^{r_0}, (psk_{ID,2})^x \cdot g^r),$$

where $P_{ID} = \{i|w_2[i] = 1, w_2 = H_1(ID)\}, P_{ID,0} = \{i|w_1[i] = 1, w_1 = H_1(ID, 0)\}$.

(5) KeyUpdateH($\text{params}, ID, HK_{ID}, t_{old}, t_{new}$): The helper first computes $r_{old} = F(ID, HK_{ID}, t_{old})$ and $r_{new} = F(ID, HK_{ID}, t_{new})$. It then computes an update key as

$$UK_{ID, t_{old}, t_{new}} = (\tilde{T}_{ID, t_{old}, t_{new}}, \tilde{R}_{ID, t_{old}, t_{new}}) = (f_1^{r_{new}}(P_{ID, t_{new}}), g^{r_{new} - r_{old}}),$$

where $P_{ID, t_{old}} = \{i|w_1[i] = 1, w_1 = H_1(ID, t_{old})\}, P_{ID, t_{new}} = \{i|w_1[i] = 1, w_1 = H_1(ID, t_{new})\}$.

(6) KeyUpdateU($\text{params}, ID, T_{SK_{ID, t_{old}}}, UK_{ID, t_{old}, t_{new}}$): The user first respectively parses the temporary signing key $T_{SK_{ID, t_{old}}}$ as $\{T_{ID, t_{old}}, R_{ID, t_{old}}, R\}$
and the update key $UK_{ID,t_{old},t_{new}}$ as $(\tilde{T}_{ID,t_{old},t_{new}}, \tilde{R}_{ID,t_{old},t_{new}})$. It then computes a temporary signing key for the time period $t_{new}$ as

$$TSK_{ID,t_{new}} = (T_{ID,t_{new}}, R_{ID,t_{new}}, \tilde{R})$$

3.2. PUBLIC KEY REPLACEMENT ATTACK. In this subsection, we show that Yang et al.'s certificateless strong key-insulated signature scheme does not achieve existential unforgeability under key replacement attack from a Type-I adversary $A_1$.

A Type-I adversary $A_1$ can forge a signature for any message $m^*$ and any time period $t^*$ on behalf of a user with identity $ID$ through the following steps:

1. The adversary $A_1$ randomly chooses $x' \in \mathbb{Z}_p^*$ and replaces the user’s public key with $upk_{ID} = e(g, g)^{x'}$.

2. For any message $m^*$ and time period $t^*$, the adversary $A_1$ randomly chooses $r_1, r_2, r_3 \in \mathbb{Z}_p^*$ and forges a signature as

$$\left(t^*, (V_{m^*}, W_{m^*}, R^*, R_{m^*})\right)$$

$$= \left(t^*, (g_2^{r_2} \cdot f_1^{r_1} (P_{ID}) \cdot f_2^{r_2} (P_{ID,t^*}) \cdot f_2^{r_2} (M_{m^*}), g^{r_2}, g^{r_2}, g^{r_3})\right),$$

where $P_{ID} = \{i | w_2[i] = 1, w_2 = H_1(ID)\}$, $P_{ID,t^*} = \{i | w_1[i] = 1, w_1 = H_1(ID, t^*)\}$ and $M_{m^*} = \{i | w_3[i] = 1, w_3 = H_2(m^*)\}$.

Obviously, the forged signature $(t^*, (V_{m^*}, W_{m^*}, R^*, R_{m^*}))$ is valid under the identity ID and the false public key $upk_{ID}^t$ since it satisfies the verification equation

$$e(V_{m^*}, g) = upk_{ID} \cdot e(f_1(P_{ID}), R^* \cdot e(f_1(P_{ID,t^*}), W_{m^*}) \cdot e(f_2(M_{m^*}), R_{m^*}).$$

3.3. MALICIOUS KGC ATTACK. In this subsection, we show that Yang et al.’s certificateless strong key-insulated signature scheme is not existentially unforgeable against a malicious KGC.

A malicious KGC can forge a signature on any message on behalf of a user with identity ID in the following way.

1. In the system setup phase, the KGC randomly selects $\gamma_0, \gamma_1, \cdots, \gamma_m \in \mathbb{Z}_p^*$ and sets the vector $M = (g^{\gamma_0}, g^{\gamma_1}, \cdots, g^{\gamma_m})$. The master key and other public parameters are generated normally.

2. After seeing a signature $(t, (V_m, W_m, R, R_m))$ on a message $m$ issued by the user with identity $ID$, the KGC chooses a random value $r_m \in \mathbb{Z}_p^*$ and defines a new signature on any message $m^*$ as

$$\left(t, (V_m \cdot (R_m)^{-1}, \sum_{i=1}^m (\gamma_i \cdot w_i)), f_2^{r_m} (M_{m^*}), W_m, R, g^{r_m}\right).$$
where $M_{m^*} = \{ i | w_3[i] = 1, w_3 = H_2(m^*) \}$ and $w_i$ denotes the $i$-th bit of $H_2(m)$.

Obviously, the forged signature $(t, (V_{m^*}, W_{m^*}, R^*, R_{m^*}))$ is valid under the identity $ID$ and the public key $upk_{ID}$ since it satisfies the verification equation

$$e(V_{m^*}, g) = e(f_1(P_{ID}), R^*) \cdot e(f_1(P_{ID}, t^*), W_{m^*}) \cdot e(f_2(M_{m^*}), R_{m^*}).$$

**Remark 1.** The above malicious KGC attack also can act on the certificateless key-insulated signature scheme proposed by Wan et al. in [22]. We skip the description of the scheme and briefly illustrate the concrete attack. Assume that a malicious KGC randomly selects $\gamma_0, \gamma_1, \cdots, \gamma_n \in \mathbb{Z}_p^*$ and defines the vector in the public parameters $V = (v_0, v_1, \cdots, v_n) = (g^{\gamma_0}, g^{\gamma_1}, \cdots, g^{\gamma_n})$. After obtaining a signature $(t, (D, B, a_{ID}, c_{ID, i}))$ issued by a user $ID$, the KGC chooses a random value $a_{m^*} \in \mathbb{Z}_p^*$ and defines a new signature on any message $m^*$ as $(t, (D^*, B^*, a_{ID}', c_{ID, i}')) = (g^{a_{m^*}}, B \cdot (D)^{-\gamma_0+\sum_{i=1}^n(\gamma_i, u_i)} \cdot f_{a_{m^*}}(M^*), a_{ID}, c_{ID, i})$, where $M^* = \{ i | M^*[i] = 1, M^* = H(m^*) \}$ and $w_i$ denote the $i$-th bit of $H(m)$. It is easy to verify that the above forgery is valid.

4. **The proposed scheme**

To overcome the security weaknesses in the previous certificateless key-insulated signature schemes [22, 23], we propose a new certificateless strong key-insulated signature scheme and prove it to be SKI\&EUF-CMA secure under the Squ-CDH assumption in the standard model. We also make a comparison between our new scheme and Wan et al.’s certificateless strong key-insulated signature scheme [23].

4.1. **DESCRIPTION OF THE PROPOSED SCHEME.** A formal description of the proposed scheme is as follows:

1. **Setup**($k, N$): Let $G_1$ and $G_2$ be two cyclic groups of same prime order $p$, $g$ be a generator of $G_1$ and $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map. The KGC randomly chooses $\alpha \in \mathbb{Z}_p^*$ and computes $g_1 = g^\alpha$. It also randomly chooses $g_2, v_0, u_1, \cdots, u_n, v_0, v_1 \in G_1$. Furthermore, it chooses three collision-resistant hash functions $H_1 : \{ 0, 1 \}^* \rightarrow \{ 0, 1 \}^{n_0}, H_2 : \{ 0, 1 \}^* \rightarrow \mathbb{Z}_p^*$ and $H_3 : \{ 0, 1 \}^* \rightarrow \mathbb{Z}_p^*$. Let $Q$ be a point in $G_1$. Define a function $f(Q)$ as follows. If the $x$-coordinate of $Q$ is odd, then $f(Q) = 1$; else, $f(Q) = 0$. The public parameters are $\text{params} = \{ N, G_1, G_2, e, p, g, g_1, g_2, u_0, u_1, \cdots, u_n, v_0, v_1, H_1, H_2, H_3, f \}$ and the master key is $\text{msk} = g_1^\alpha$.

2. **PartialKeyGen**($\text{params}$, $\text{msk}$, $ID$): The KGC randomly picks $r \in \mathbb{Z}_p^*$ and computes a partial private key for the user with identity $ID$ as

$$psk_{ID} = (psk_{ID}^{(1)}, psk_{ID}^{(2)}) = (g_1^r \cdot (u_0 \prod_{i=1}^n \lambda_{ID}^{(i)}))', g^r),$$

where $\lambda_{ID}^{(i)}$ denotes the $i$-th bit of $\lambda_{ID} = H_1(ID)$.

3. **UserKeyGen**($\text{params}$, $ID$): A user with identity $ID$ randomly chooses $x_{ID} \in \mathbb{Z}_p^*$ as his private key $usk_{ID}$ and computes his public key as

$$upk_{ID} = (upk_{ID}^{(1)}, upk_{ID}^{(2)}, upk_{ID}^{(3)}) = (g^x_{ID}, g_2^\frac{1}{x_{ID}}, e(g_1, g_2)x_{ID}).$$

The public key $upk_{ID} = (upk_{ID}^{(1)}, upk_{ID}^{(2)}, upk_{ID}^{(3)})$ can be validated by checking whether the equations $e(upk_{ID}^{(1)}, upk_{ID}^{(2)}) = e(g_1, g_2)$ and $e(upk_{ID}^{(1)}, upk_{ID}^{(1)}) = upk_{ID}^{(3)}$ hold.
(4) **KeyInitialize** *(params, pskID, uskID):* A user with identity ID randomly chooses his helper’s key $HK_ID \in \{0,1\}^k$. He then randomly chooses $r' \in \mathbb{Z}_p$ and computes an initial temporary signing key as

$$TSK_{ID,0} = (TSK^{(1)}_{ID,0}, TSK^{(2)}_{ID,0}, TSK^{(3)}_{ID,0})$$

$$= ((psk_{ID})^{uskID^2} \cdot (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID}^{(i)}})^{r'} \cdot (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,0}^{(i)}})^{r_0}, g^{r_0},$$

$$((psk_{ID})^{uskID^2} \cdot g^{r'}).$$

where $r_0 = H_2(ID, HK_{ID,0})$, $\lambda_{ID}^{(i)}$ and $\lambda_{ID,0}^{(i)}$ denote the i-th bit of of $\lambda_{ID} = H_1(ID)$ and $\lambda_{ID,0} = H_1(ID,0)$ respectively.

Obviously, if let $\tilde{r} = r \cdot x_{ID}^2 + r'$, then $TSK_{ID,0} = (g^{\sigma_{x_{ID}^2} \cdot (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID}^{(i)}})^{\tilde{r}} \cdot (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,0}^{(i)}})^{r_0}}, g^{r_0}, g^{\tilde{r}})$. 

(5) **KeyUpdateH** *(params, ID, HKID, told, tnew):* The helper first computes $r_{old} = H_2(ID, HK_{ID}, t_{old})$ and $r_{new} = H_2(ID, HK_{ID}, t_{new})$. It then computes an update key as

$$UK_{ID,t_{old},t_{new}} = (UK^{(1)}_{ID,t_{old},t_{new}}, UK^{(2)}_{ID,t_{old},t_{new}})$$

$$= ((u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,t_{old}}^{(i)}})^{r_{new}} / (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,t_{old}}^{(i)}})^{r_{old}}), g^{r_{new} - r_{old}}),$$

where $\lambda_{ID,t_{old}}^{(i)}$ and $\lambda_{ID,t_{new}}^{(i)}$ denote the i-th bit of of $\lambda_{ID,0} = H_1(ID, t_{old})$ and $\lambda_{ID,0} = H_1(ID, t_{new})$ respectively.

(6) **KeyUpdateU** *(params, ID, TSKID,told, UKID,told,tnew):* On receiving an update key $UK_{ID,t_{old},t_{new}} = (UK^{(1)}_{ID,t_{old},t_{new}}, UK^{(2)}_{ID,t_{old},t_{new}})$, the user uses his old temporary signing key $TSK_{ID,t_{old}} = (TSK^{(1)}_{ID,t_{old}}, TSK^{(2)}_{ID,t_{old}}, TSK^{(3)}_{ID,t_{old}})$ to compute a new temporary signing key for the time period $t_{new}$ as

$$TSK_{ID,t_{new}} = (TSK^{(1)}_{ID,t_{new}}, TSK^{(2)}_{ID,t_{new}}, TSK^{(3)}_{ID,t_{new}})$$

$$= (UK^{(1)}_{ID,t_{old},t_{new}} \cdot TSK^{(1)}_{ID,t_{old}}, UK^{(2)}_{ID,t_{old},t_{new}} \cdot TSK^{(2)}_{ID,t_{old}},$$

$$TSK^{(3)}_{ID,t_{old}}).$$

It is not difficult to deduce that

$$TSK_{ID,t_{new}} = (g^{\sigma_{x_{ID}^2}} \cdot (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,t_{new}}^{(i)}})^{t_{new}} / (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,t_{new}}^{(i)}})^{r_{new}}), g^{r_{new}}, g^{t_{new}}).$$

(7) **Sign** *(params, t, m, ID, TSKID,t):* To sign a message $m$ in the time period $t$, a signer with identity ID first randomly chooses $r'_t, s \in \mathbb{Z}_p$ and uses his temporary signing key $TK_ID,t = (TSK^{(1)}_{ID,t}, TSK^{(2)}_{ID,t}, TSK^{(3)}_{ID,t})$ to compute

$$\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4) = (g^s \cdot TSK^{(2)}_{ID,t} \cdot g^{r'_t}, TSK^{(3)}_{ID,t},$$

$$TSK^{(1)}_{ID,t} \cdot (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,t}^{(i)}})^{t'}, (((upk_{ID})^{\gamma} \cdot v_3)^s).$$

where $\beta = f(s_2), \gamma = H_3(\sigma_1, \sigma_2, \sigma_3, ID, upk_{ID}, m, v_3)$ and $\lambda_{ID,t}^{(i)}$ denotes the i-th bit of $\lambda_{ID,t} = H_1(ID, t)$. It then sets the signature for the message $m$ as $(t, \sigma)$.  

**Advances in Mathematics of Communications** Volume 9, No. 3 (2015), 353–373
If let \( \tilde{r}_i = r_t + r'_t \), then we have that
\[
\sigma = (g^s, g^{\tilde{r}_i}, g^{\tilde{r}}, g_1^{\tilde{r}_1} \cdot (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,i}^{(1)}})^{\tilde{r}_i} \cdot (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,i}^{(1)}})^{\tilde{r}_i} \cdot ((upk_{ID})^\gamma \cdot v_\beta)^{\tilde{r}_i}).
\]

(8) **Verify** \((\text{params}, m, (t, \sigma), ID, upk_{ID})\): To verify a signature \((t, (\sigma_1, \sigma_2, \sigma_3, \sigma_4))\), a verifier checks whether the following equality holds:
\[
e(\sigma_4, g) = upk_{ID}^{(3)} \cdot e((u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,i}^{(1)}})^{\tilde{r}_i} \cdot (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,i}^{(1)}})^{\tilde{r}_i} \cdot (u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,i}^{(1)}})^{\tilde{r}_i} \cdot ((upk_{ID})^\gamma \cdot v_\beta)^{\tilde{r}_i}).
\]

where \( \lambda_{ID}^{(i)} \) and \( \lambda_{ID,t}^{(i)} \) denote the \( i \)-th bit of \( \lambda_{ID} = H_1(ID) \) and \( \lambda_{ID,t} = H_1(ID, t) \) respectively, \( \beta = f(\sigma_2) \) and \( \gamma = H_3(\sigma_1, \sigma_2, \sigma_3, ID, upk_{ID}, m, v_\beta). \) If it holds, output 1. Otherwise, output 0.

The correctness of the proposed scheme can be verified by the following equation:
\[
e(\sigma_4, g) = upk_{ID}^{(3)} \cdot e((u_0 \prod_{i=1}^{n_u} u_i^{\lambda_{ID,i}^{(1)}})^{\tilde{r}_i} \cdot ((upk_{ID})^\gamma \cdot v_\beta)^{\tilde{r}_i}).
\]

**Remark 2.** The public key replacement attack presented in Section 3.2 is based on the fact that there is no public key validity checking mechanism in Wan et al.’s certificateless strong key-insulated signature scheme [23]. To avoid this attack, we redesign the public key generation algorithm. In our new scheme, a user’s public key \( upk_{ID} = (upk_{ID}^{(1)}, upk_{ID}^{(2)}, upk_{ID}^{(3)}) \) is composed of three components and can be validated by checking the equations \( e(upk_{ID}^{(1)}, upk_{ID}^{(2)}) = e(g_1, g_2) \) and \( e(upk_{ID}^{(1)}, upk_{ID}^{(3)}) = upk_{ID}^{(3)}. \) These two equations guarantee that a public key has a correct relation and its third component is always under the form of \( e(g_1, g_1)^x \) or \( e(g_2, g_2)^x \) for some random value \( x \in \mathbb{Z}_p^* \) even if it has been replaced. Thus, a Type-I adversary cannot forge a signature as in Section 3.2 unless it knows the master key or the corresponding partial private key.

**Remark 3.** The basic reason of the malicious KGC attack presented in Section 3.3 is that in both the previous two certificateless key-insulated signature schemes [22, 23], the part of signatures including the messages is irrelevant to the signer’s private key. More specifically, a malicious KGC can add or remove \( f_{ID}^m(M_m) \) or \( f_{ID}^m(M) \) in a signature without affecting the validity of the signature. To fight against this attack, our scheme refines the signature algorithm by embedding the second component of the signer’s public key \( upk_{ID}^{(2)} \) in the signatures. Thus, a
malicious KGC can not forge a new signature \((t,\sigma^*)\) from an existing one \((t,\sigma)\) as in Section 3.3 unless it can remove \((upk^{(2)}_{kID})^{\gamma,s}\) from \(\sigma\).

4.2. Security analysis.

**Theorem 4.1.** If the Squ-CDH assumption holds in \(G_1\), then the proposed scheme is KI & EUF-CMA secure in the standard model.

This theorem can be proved by combining the following two lemmas.

**Lemma 4.2.** If \(A_1\) is a Type-I adversary against the proposed scheme that has advantage at least \(\epsilon\) to win the game \(\text{Game-A}\), then there exists an algorithm \(B\) to solve the Squ-CDH problem with advantage \(\epsilon' \geq 16\epsilon(n_u+1)q_{psk}^q q_{usk}^q\), where \(q_{psk}\) and \(q_{usk}\) respectively denote the number of \(A_1\)'s queries to \(\text{OPartialPrivateKey}\) and \(\text{OTempSigningKey}\).

Proof. We show how to construct an algorithm \(B\) to solve the Squ-CDH problem from the adversary \(A_1\). Assume that the algorithm \(B\) is given a random instance \((g,A = g^\alpha)\) of the Squ-CDH problem in \(G_1\) and asked to compute \(g^{\alpha^2}\). In order to use the adversary \(A_1\) to solve the given Squ-CDH problem, the algorithm \(B\) needs to simulate a challenger and all oracles for the adversary \(A_1\).

**Setup.**
Let \(l = 2(q_{psk} + q_{usk})\). We assume that \(p\) is sufficiently bigger than \(l(n_u + 1)\). The algorithm \(B\) randomly chooses \(x_0, x_1, \cdots, x_{n_u} \in \mathbb{Z}_p\) and \(x, z, e_0, e_1 \in \mathbb{Z}_p^*\). It also chooses a random integer \(k \in \{0, n_u\}\). Then it assigns the public parameters as \(g_1 = A, g_2 = g^\gamma, u_0 = A^{-lk+x_0} \cdot g^{y_0}, u_i = A^{x_i} \cdot g^{y_i} (1 \leq i \leq n_u), v_0 = g^{\alpha_0}\) and \(v_1 = A^{z} \cdot g^{\alpha_1}\).

Note that the above assignment implies that the master key will be \(g_1^\alpha = g^{\alpha^2}\) which is unknown to the algorithm \(B\). The algorithm \(B\) gives the public system parameters \(\text{params}\) to the adversary \(A_1\).

For the convenience of analysis, we respectively define two functions \(J(\zeta) = \gamma - lk + x_0 + \sum_{i=1}^{n_u} \zeta_i \cdot x_i\) and \(K(\zeta) = y_0 + \sum_{i=1}^{n_u} \zeta_i \cdot y_i\) for an \(n_u\)-length binary string \(\zeta = \zeta_1 \cdots \zeta_{n_u}\). It is not difficult to deduce that \(F(\zeta) = u_0 \prod_{i=1}^{n_u} u_i^{\zeta_i} = A^{J(\zeta)} \cdot g^{K(\zeta)}\).

**Queries.** Given the public parameters, the adversary \(A_1\) asks a series of polynomially bounded number of queries in an adaptive fashion. The algorithm \(B\) maintains a list \(\text{KeyList}\) of tuples \((ID, upk_{kID}, usk_{kID}, HK_{kID})\) which is initially empty and simulates all oracles as follows:

1. \(\text{OCreateUser}\): On receiving such a query on an identity \(ID\), the algorithm \(B\) looks up the list \(\text{KeyList}\) to find a tuple \((ID, upk_{kID}, usk_{kID}, HK_{kID})\) associated with the identity \(ID\). If such a tuple exists, the algorithm \(B\) returns \(upk_{kID}\) to the adversary \(A_1\). Otherwise, it performs the algorithm \(\text{UserKeyGen}\) to generate a public key and private key pair \((upk_{kID}, usk_{kID})\), randomly chooses a helper key \(HK_{kID} \in \{0,1\}^k\), stores a new tuple \((ID, upk_{kID}, usk_{kID}, HK_{kID})\) in the list \(\text{KeyList}\) and returns \(upk_{kID}\) to the adversary \(A_1\).

2. \(\text{OPrivateKey}\): On receiving a private key query on an identity \(ID\), the algorithm \(B\) looks up the list \(\text{KeyList}\) to find a tuple \((ID, upk_{kID}, usk_{kID}, HK_{kID})\) associated with the identity \(ID\) and returns \(usk_{kID}\) to the adversary \(A_1\).

3. \(\text{ReplacePublicKey}\): On input an identity \(ID\) and a pair of values \((upk'_{kID}, usk'_{kID})\), the algorithm \(B\) looks up the list \(\text{KeyList}\) to find a tuple \((ID, upk_{kID}, usk_{kID}, HK_{kID})\) associated with the identity \(ID\) and replaces the tuple with \((ID, upk'_{kID}, usk'_{kID}, HK_{kID})\).

4. \(\text{OPartialPrivateKey}\): On receiving a partial private key query on an identity \(ID\), the algorithm \(B\) does the following:
- If $J(\lambda_{ID}) = 0 \mod p$ where $\lambda_{ID} = H_1(ID)$, then it aborts the simulation.
- Otherwise, it randomly chooses $r \in \mathbb{Z}_p^*$ and computes a partial private key for the identity $ID$ as

$$psk_{ID} = (A^{-K(\lambda_{ID})/J(\lambda_{ID})} \cdot F(\lambda_{ID})^r, A^{-1/J(\lambda_{ID})} \cdot g^r).$$

Note that the algorithm $B$ can generate such a partial private key if and only if $J(\lambda_{ID}) \not= 0 \mod l$, which suffices to have $J(\lambda_{ID}) \not= 0 \mod p$. If let $r_{ID} = r - a/J(\lambda_{ID})$, we get

$$psk_{ID} = (A^a \cdot (1^{-J(\lambda_{ID})} \cdot g^{K(\lambda_{ID})})^{-a/J(\lambda_{ID})} \cdot F(\lambda_{ID})^r, g^{r_{ID}}) = (g^{a^2} \cdot F(\lambda_{ID})^{r_{ID}}, g^{r_{ID}}).$$

Then, the algorithm $B$ returns the key $psk_{ID}$ to the adversary $A_1$.

(5) $O^{TempSigningKey}$. On receiving a temporary signing key query on an identity $ID$ and a period time index $t(0 \leq t \leq N - 1)$, the algorithm $B$ does the following:
- If $J(\lambda_{ID}) = 0 \mod p$ where $\lambda_{ID} = H_1(ID)$, then it aborts the simulation.
- Otherwise, it randomly chooses $r, r' \in \mathbb{Z}_p^*$, looks up the list KeyList to find a tuple $(ID, upk_{ID}, usk_{ID}, HK_{ID})$ associated with the identity $ID$ to retrieve $usk_{ID}$ and $HK_{ID}$, and then computes a temporary signing key $TSK_{ID, t}$ for the identity $ID$ in the time period $t$ as

$$TSK_{ID, t} = ((A^{-K(\lambda_{ID})/J(\lambda_{ID})} \cdot F(\lambda_{ID})^r) \cdot usk_{ID}^2 \cdot F(\lambda_{ID})^{r'}, (g^{r_1} \cdot (A^{-1/J(\lambda_{ID})} \cdot g^r) \cdot usk_{ID}^2 \cdot g^{r'}),$$

where $r_1 = H_2(ID, HK_{ID}, t)$, $\lambda_{ID} = H_3(ID)$ and $\lambda_{ID, t} = H_3(ID, t)$ respectively. Note that the algorithm $B$ can generate such a temporary signing key if and only if $J(\lambda_{ID}) \not= 0 \mod l$, which suffices to have $J(\lambda_{ID}) \not= 0 \mod p$. If let $r = (r - a/J(\lambda_{ID})) \cdot usk_{ID}^2 + r'$, we get

$$TSK_{ID, t} = (g^{a^2} \cdot F(\lambda_{ID})^r, F(\lambda_{ID, t})^{r'}, g^{r'}, g^{r'}).$$

Then, the algorithm $B$ returns the key $TSK_{ID, t}$ to the adversary $A_1$.

(6) $O^{Sign}$. On receiving a signing query on an identity $ID$, a period time index $t(0 \leq t \leq N - 1)$ and a message $m$, the algorithm $B$ does the following:
- If $J(\lambda_{ID}) \not= 0 \mod p$ where $\lambda_{ID} = H_1(ID)$, it first makes a query to the oracle $O^{TempSigningKey}$ on $(ID, t)$ to obtain a temporary signing key $TSK_{ID, t}$. Then it runs the algorithm $Sign(params, t, m, ID, TSK_{ID, t})$ to generate a signature $(t, \sigma) = (t, (\sigma_1, \sigma_2, \sigma_3, \sigma_4))$ for the message $m$ as follows:

$$\sigma_1 = A^{-usk_{ID}^2/z} \cdot g^a = g^{a^2 - a \cdot usk_{ID}^2/z}, \sigma_2 = g^{r_1}, \sigma_3 = g^r,$$

$$\sigma_4 = (upk_{ID}^{(1)})^{-((x \gamma + e \cdot usk_{ID}^2)/z)} \cdot F(\lambda_{ID})^r \cdot F(\lambda_{ID, t})^{r_1} \cdot ((upk_{ID}^{(2) \gamma} \cdot v_1)^s)$$

$$= (g^{usk_{ID}^2})^{-((x \gamma + e \cdot usk_{ID}^2)/z)} \cdot F(\lambda_{ID})^r \cdot F(\lambda_{ID, t})^{r_1} \cdot ((upk_{ID}^{(2) \gamma} \cdot v_1)^s)$$

$$= g^{-a \cdot x \cdot usk_{ID}^2 + a \cdot x \cdot usk_{ID}^2/z} \cdot F(\lambda_{ID})^r \cdot F(\lambda_{ID, t})^{r_1} \cdot ((upk_{ID}^{(2) \gamma} \cdot v_1)^s)$$

$$= (g^{a^2 \cdot usk_{ID}^2} \cdot g^{a^2 \cdot usk_{ID}^2} \cdot F(\lambda_{ID})^r \cdot F(\lambda_{ID, t})^{r_1})$$
Thus, the algorithm $B$ will not abort during the simulation if the following holds simultaneously:

1. All $A_1$'s queries to the oracle $O_{\text{PartialPrivateKey}}$ on $ID$ have $J(\lambda_{ID}) \neq 0 \mod p$;
2. All $A_1$'s queries to the oracle $O_{\text{TempSigningKey}}$ on $(ID, t)$ have $J(\lambda_{ID}) \neq 0 \mod p$;
3. $A_1$'s forgery $(ID^*, m^*, (t^*, \sigma^*))$ satisfies $J(\lambda_{ID^*}) = 0 \mod p$, $J(\lambda_{ID^*, t^*}) = 0 \mod p$ and $f(\sigma_2^*) = 0$.

Let $\lambda_1, \lambda_2, \cdots, \lambda_h$ be the hash values appearing in these queries not involving $\lambda_{ID^*}$ and $\lambda_{ID^*, t^*}$. Clearly, we have $h \leq d_{psk} + q_{tsk}$. Define the events $A_i$, $A^*$, $B^*$ and $C^*$ as follows:

\begin{align*}
\text{Advantages in Mathematics of Communications Volume 9, No. 3 (2015), 353–373}
\end{align*}
- $A_i : J(\lambda_i) \neq 0 \mod p$, where $i = 1, 2, \cdots, h$;
- $A^* : J(\lambda_{ID^*}) = 0 \mod p$;
- $B^* : J(\lambda_{ID^*,t^*}) = 0 \mod p$;
- $C^* : f(\sigma_2^*) = 0$.

Thus, the probability of the algorithm $B$ not aborting is

$$Pr[\neg \text{abort}] \geq Pr \left[ \bigwedge_{i=1}^{h} A_i \land A^* \land B^* \land C^* \right].$$

The assumption $l(n_u + 1) < p$ implies that if $J(\lambda_{ID^*}) = 0 \mod p$ then $J(\lambda_{ID^*}) = 0 \mod l$. Furthermore, this assumption also implies that if $J(\lambda_{ID^*}) = 0 \mod l$, then there is a unique integer $\theta(0 \leq \theta \leq n_u)$ such that $J(\lambda_{ID^*}) = 0 \mod p$. Therefore, we have

$$Pr[A^*] = Pr[J(\lambda_{ID^*}) = 0 \mod p] = Pr[J(\lambda_{ID^*}) = 0 \mod p \land J(\lambda_{ID^*}) = 0 \mod l] = Pr[J(\lambda_{ID^*}) = 0 \mod p \mid J(\lambda_{ID^*}) = 0 \mod l] \cdot Pr[J(\lambda_{ID^*}) = 0 \mod l] = \frac{1}{(n_u + 1)l}.$$

Similarly, we have $Pr[B^*] = \frac{1}{(n_u + 1)l}$. Because the events $A_i$, $A^*$ and $B^*$ are independent for any $i$, we get

$$Pr \left[ \bigwedge_{i=1}^{h} A_i \land A^* \land B^* \right] = Pr \left[ \bigwedge_{i=1}^{h} A_i \mid A^* \land B^* \right] \cdot Pr[A^* \land B^*] \geq \left(1 - Pr \left[ \bigvee_{i=1}^{h} \neg A_i \mid A^* \land B^* \right] \right) \cdot Pr[A^*] \cdot Pr[B^*] = \left(1 - \frac{h}{l}\right) \cdot \frac{1}{(n_u + 1)^2} \cdot l^2 \geq \frac{8(n_u + 1)^2(q_{psk} + q_{tsk})^2}{l^2}.$$

Furthermore, the events $\bigwedge_{i=1}^{h} A_i \land A^* \land B^*$ and $C^*$ are independent and $Pr[C^*] = 1/2$ since with probability the $x$-coordinate of $g^*$ is odd for a random value $r \in \mathbb{Z}_p^*$. Therefore, we get

$$Pr[\neg \text{abort}] \geq Pr \left[ \bigwedge_{i=1}^{h} A_i \land A^* \land B^* \right] \cdot Pr[C^*] = \frac{1}{16(n_u + 1)^2(q_{psk} + q_{tsk})^2}.$$

If the simulation does not abort, the adversary $A_1$ will win the game with probability at least $\epsilon$. Thus, the algorithm $B$ can solve the given Squ-CDH problem with probability

$$\epsilon' \geq \frac{\epsilon}{16(n_u + 1)^2(q_{psk} + q_{tsk})^2}. \quad \square$$

**Lemma 4.3.** If $A_2$ is a Type-II adversary against the proposed scheme that has advantage at least $\epsilon$ to win the game $Game-A$, then there exists an algorithm $B$ to solve the Squ-CDH problem with advantage $\epsilon' \geq \frac{\epsilon}{16(n_u + 1)^2q_{tsk}}$, where $q_{tsk}$ denotes the number of $A_2$’s queries to $OTempSigningKey$. 

ADVANCES IN MATHEMATICS OF COMMUNICATIONS Volume 9, No. 3 (2015), 353–373
Proof. We show how to construct an algorithm $B$ to solve the Squ-CDH problem from the adversary $A_2$. Assume that the algorithm $B$ is given a random instance $(g, A = g^2)$ of the Squ-CDH problem in $G_1$ and asked to compute $g^{a_2}$. In order to use the adversary $A_2$ to solve the given Squ-CDH problem, the algorithm $B$ needs to simulate a challenger and all oracles for the adversary $A_2$.

Setup. Let $l = 2q_{sk}$. We assume that $p$ is sufficiently bigger than $l(n_u + 1)$. The algorithm $B$ randomly chooses $x_0, x_1, \ldots, x_n \in \mathbb{Z}_p, y_0, y_1, \ldots, y_n \in \mathbb{Z}_p$ and $\alpha, x, z, e_0, e_1 \in \mathbb{Z}_p^*$. It also chooses a random integer $k \in [0, n_u]$. Then it assigns the public parameters as $g_1 = g^a$, $g_2 = A^x$, $u_0 = A^{p-\lambda_1} \cdot g^{b_0}$, $u_i = A^{x} \cdot g^{b_i} (1 \leq i \leq n_u)$, $v_0 = g^{e_0}$ and $v_1 = A^z \cdot g^{e_1}$.

The algorithm $B$ gives the public system parameters $\text{params}$ and the master key $g^a$ to the adversary $A_2$. For the convenience of analysis, we respectively define two functions $J(\zeta) = (p - \lambda_1) + x_0 + \sum_{i=1}^{n_u} \zeta_i \cdot x_i$ and $K(\zeta) = y_0 + \sum_{i=1}^{n_u} \zeta_i \cdot y_i$ for an $n_u$-length binary string $\zeta = \zeta_1 \cdots \zeta_{n_u}$. It is not difficult to deduce that $F(\zeta) = u_0 \prod_{i=1}^{n_u} u_i^{\zeta_i} = A^{J(\zeta)} \cdot g^{K(\zeta)}$.

Queries. Given the public parameters and the master key, the adversary $A_2$ asks a series of polynomially bounded number of queries in an adaptive fashion. The algorithm $B$ maintains a list $\text{KeyList}$ of tuples $(ID, upk_{ID}, usk_{ID}, H_{K_{ID}})$ which is initially empty and simulates all oracles as follows:

1. $O_{\text{CreateUser}}$: On input an identity $ID$, the algorithm $B$ looks up the list $\text{KeyList}$ to find a tuple $(ID, upk_{ID}, usk_{ID}, H_{K_{ID}})$ associated with the identity $ID$. If such a tuple exists, the algorithm $B$ returns $upk_{ID}$ to the adversary $A_2$. Otherwise, it randomly chooses $usk_{ID} \in \mathbb{Z}_p^*$ and $H_{K_{ID}} \in \{0,1\}^k$, computes $upk_{ID} = (A^{\alpha_{usk_{ID}}}, g^{\alpha_{usk_{ID}}}, e(A^{\alpha_{usk_{ID}}}, A^{\alpha_{usk_{ID}}}))$, stores $(ID, upk_{ID}, usk_{ID}, H_{K_{ID}})$ in the list $\text{KeyList}$ and returns $upk_{ID}$ to the adversary $A_2$.

2. $O_{\text{PrivateKey}}$: On receiving a private key query on an identity $ID$, the algorithm $B$ looks up the list $\text{KeyList}$ to find a tuple $(ID, upk_{ID}, usk_{ID}, H_{K_{ID}})$ associated with the identity $ID$ and returns $usk_{ID}$ to the adversary $A_2$.

3. $O_{\text{TempSigningKey}}$: On receiving a temporary signing key query on an identity $ID$ and a period time index $t$ ($0 \leq t \leq N - 1$), the algorithm $B$ does the following:

- If $J(\lambda_{ID}) = 0 \mod p$ where $\lambda_{ID} = H_1(ID)$, then it aborts the simulation.
- Otherwise, it randomly chooses $r \in \mathbb{Z}_p^*$, looks up the list $\text{KeyList}$ to find a tuple $(ID, upk_{ID}, usk_{ID}, H_{K_{ID}})$ associated with the identity $ID$ to retrieve $usk_{ID}$ and $H_{K_{ID}}$, and then computes a temporary signing key $TSK_{ID,t}$ for the identity $ID$ in the time period $t$ as

$$TSK_{ID,t} = (A^{-\frac{K(\lambda_{ID})}{2J(\lambda_{ID})}} \cdot (\alpha_{usk_{ID}})^2 \cdot F(\lambda_{ID})^r \cdot F(\lambda_{ID,t})^{r_t},$$

$$g^{r_t}, A^{-\frac{2}{J(\lambda_{ID})} \cdot g^{r_t}}),$$

where $r_t = H_2(ID, H_{K_{ID},t}), \lambda_{ID} = H_1(ID)$ and $\lambda_{ID,t} = H_1(ID,t)$ respectively. Note that the algorithm $B$ can generate such a temporary signing key if and only if $J(\lambda_{ID}) \neq 0 \mod l$, which suffices to have $J(\lambda_{ID}) \neq 0 \mod p$. If let $\bar{r} = r - a \cdot (\alpha_{usk_{ID}})^2 / J(\lambda_{ID})$, we get

$$TSK_{ID,t} = (g^{a \cdot \alpha_{usk_{ID}} \cdot 2} \cdot F(\lambda_{ID})^{\bar{r}} \cdot F(\lambda_{ID,t})^{r_t}, g^{r_t}, g^{\bar{r}}).$$

Then, the algorithm $B$ returns the key $TSK_{ID,t}$ to the adversary $A_2$.

4. $O_{\text{Sign}}$: On receiving a signing query on an identity $ID$, a period time index $t$ ($0 \leq t \leq N - 1$) and a message $m$, the algorithm $B$ does the following:
- If $J(\lambda_{ID}) \neq 0 \mod p$ where $\lambda_{ID} = H_1(ID)$, it first makes a query to the oracle $OTempSimgingKey$ on $(ID, t)$ to obtain a temporary signing key $TSK_{ID,t}$.

Then it runs the algorithm $Sign(params, t, m, ID, TSK_{ID,t})$ to generate a signature $(t, \sigma)$ and returns it to the adversary $A_2$.

- Otherwise, it first randomly chooses $\tilde{r}_t \in \mathbb{Z}_p^*$ such that $f(g^{\tilde{r}_t}) = 1$. Then it randomly chooses $s, \tilde{r} \in \mathbb{Z}_p^*$ and generates a signature $(t, \sigma) = (t, (\sigma_1, \sigma_2, \sigma_3, \sigma_4))$ for the message $m$ as follows:

$$
\sigma_1 = A \cdot \alpha \cdot \frac{\alpha - (a \cdot usk_{ID})^2}{z} \cdot g^s \cdot \gamma^\ast, \quad \sigma_2 = g^{\tilde{r}_t}, \quad \sigma_3 = g^{\tilde{r}},
$$

$$
\sigma_4 = (upk_{ID}^{(2)})^{\frac{1}{\gamma}} \cdot F(\lambda_{ID})^{\tilde{r}_t} \cdot F(\lambda_{ID,t})^{\tilde{r}_t} \cdot (\left(upk_{ID}^{(2)}\right)^\gamma \cdot v_1)^s \cdot \frac{\alpha - (a \cdot usk_{ID})^2}{z} \cdot g^s \cdot \gamma^\ast
$$

$$
= g^s \cdot g^{\tilde{r}_t} \cdot g^{\tilde{r}} \cdot g^{(a-a \cdot usk_{ID})^2} \cdot F(\lambda_{ID})^{\tilde{r}_t} \cdot F(\lambda_{ID,t})^{\tilde{r}_t} \cdot (\left(upk_{ID}^{(2)}\right)^\gamma \cdot v_1)^s
$$

where $\lambda_{ID} = H_1(ID)$, $\lambda_{ID,t} = H_1(ID,t)$ and $\gamma = H_3(\sigma_1, \sigma_2, \sigma_3, ID, upk_{ID}, m, v_1)$. If $s = s - a \cdot (\alpha \cdot usk_{ID})^2 / z$, then we get

$$
\sigma = (g^s, g^{\tilde{r}_t}, g^{\tilde{r}}, g^{(a-a \cdot usk_{ID})^2} \cdot F(\lambda_{ID})^{\tilde{r}_t} \cdot F(\lambda_{ID,t})^{\tilde{r}_t} \cdot (upk_{ID}^{(2)})^\gamma \cdot v_1)^s.
$$

The algorithm $B$ returns the signature $(t, \sigma)$ to the adversary $A_2$.

**Forgery.** Finally, the adversary $A_2$ outputs a forgery $(ID^*, m^*, (t^*, \sigma^*))$, where $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*)$. If $J(\lambda_{ID^*}) \neq 0 \mod p$ or $J(\lambda_{ID^*,t^*}) \neq 0 \mod p$ or $f(\sigma_2^*) = 1$ where $\lambda_{ID^*} = H_1(ID^*)$ and $\lambda_{ID^*,t^*} = H_1(ID^*, t^*)$, then the algorithm $B$ aborts the simulation. Otherwise, it retrieves the private key $usk_{ID^*}$ associated with $ID^*$ and computes

$$
T = \left(\frac{\sigma_4^*}{(\sigma_1^*)^{\gamma^* \cdot \alpha \cdot (\alpha \cdot usk_{ID^*})^2} \cdot (\sigma_2^*)^{K(\lambda_{ID^*,t^*})} \cdot (\sigma_3^*)^{K(\lambda_{ID^*})}}{\gamma^* \cdot \alpha \cdot (\alpha \cdot usk_{ID^*})^2}\right)^{\frac{1}{\gamma^* \cdot \alpha \cdot (\alpha \cdot usk_{ID^*})^2}},
$$

where $\gamma^* = H_3(\sigma_1^*, \sigma_2^*, \sigma_3^*, ID^*, upk_{ID^*}, m^*, v_0)$.

If $(t^*, \sigma^*)$ is a valid signature for the message $m^*$ under the target signer’s identity $ID^*$ and public key $upk_{ID^*}$, then we have that

$$
\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*)
$$

$$
= (g^s, g^{\tilde{r}_t}, g^{\tilde{r}}, g^{(a \cdot usk_{ID^*})^2} \cdot F(\lambda_{ID^*})^{\tilde{r}_t} \cdot F(\lambda_{ID^*,t^*})^{\tilde{r}_t} \cdot (upk_{ID^*}^{(2)})^\gamma \cdot v_0)^s
$$

$$
= (g^s, g^{\tilde{r}_t}, g^{\tilde{r}}, g^{(a \cdot usk_{ID^*})^2} \cdot g^{K(\lambda_{ID^*,t^*})^\ast} \cdot g^{K(\lambda_{ID^*})^\ast} \cdot (upk_{ID^*}^{(2)})^\gamma \cdot v_0)^s
$$

for some random values $s^*, \tilde{r}_t^*, \tilde{r}_t^* \in \mathbb{Z}_p^*$. Hence, we get

$$
T = \left(\frac{g^{(a \cdot usk_{ID^*})^2} \cdot g^{K(\lambda_{ID^*})^\gamma} \cdot g^{K(\lambda_{ID^*,t^*})^\gamma} \cdot (upk_{ID^*}^{(2)})^\gamma \cdot v_0)^s}{g^s \cdot g^{\gamma^* \cdot \alpha \cdot (\alpha \cdot usk_{ID^*})^2}}\right)^{\frac{1}{\gamma^* \cdot \alpha \cdot (\alpha \cdot usk_{ID^*})^2}} = g^{s^2}.
$$
Thus, the algorithm $B$ can output $T$ as the solution to the given Squ-CDH problem.

Now we assess the advantage of the algorithm $B$. From the above construction, we can see that the algorithm $B$ will not abort during the simulation if the following conditions hold simultaneously:

1. All $A_2$’s queries to the oracle $O_{\text{TempSigningKey}}$ on $(ID, t)$ have $J(\lambda_{ID}) \neq 0 \mod p$;
2. $A_2$’s forgery $(ID^*, m^*, (t^*, \sigma^*))$ satisfies $J(\lambda_{ID^*}) = 0 \mod p$, $J(\lambda_{ID^*, t^*}) = 0 \mod p$ and $f(\sigma^*_2) = 0$.

Let $\lambda_1, \lambda_2, \cdots, \lambda_h$ be the hash values appearing in these queries not involving $\lambda_{ID^*}$ and $\lambda_{ID^*, t^*}$. Clearly, we have $h \leq q_{tsk}$. Define the events $A_i$, $A^*$, $B^*$ and $C^*$ as follows:

- $A_i : J(\lambda_i) \neq 0 \mod p$, where $i = 1, 2, \cdots, h$;
- $A^* : J(\lambda_{ID^*}) = 0 \mod p$;
- $B^* : J(\lambda_{ID^*, t^*}) = 0 \mod p$;
- $C^* : f(\sigma^*_2) = 0$.

Thus, the probability of the algorithm $B$ not aborting is

$$Pr[\neg \text{abort}] \geq Pr\left[\bigwedge_{i=1}^{h} A_i \land A^* \land B^* \land C^*\right].$$

Analogously to the proof of Lemma 4.2, we can deduce that

$$Pr[\neg \text{abort}] \geq \frac{1}{16(n_u + 1)^2 q_{tsk}^2}.$$

Hence, the algorithm $B$ can solve the given Squ-CDH problem with probability

$$\epsilon' \geq \frac{\epsilon}{16(n_u + 1)^2 q_{tsk}^2}.$$

**Theorem 4.4.** If the Squ-CDH assumption holds in $G_1$, then the proposed scheme is SKI & EUF-CMA secure in the standard model.

This theorem can be proved by combining Theorem 4.1 and the following two lemmas.

**Lemma 4.5.** If $A_1$ is a Type-I adversary against the proposed scheme that has advantage at least $\epsilon$ to win the game $\text{Game-B}$, then there exists an algorithm $B$ to solve the Squ-CDH problem with advantage $\epsilon' \geq \frac{\epsilon}{16(n_u + 1)^2 q_{psk}^2 q_{tsk}^2}$, where $q_{psk}$ and $q_{tsk}$ respectively denote the number of $A_1$’s queries to $O_{\text{PartialPrivateKey}}$ and $O_{\text{TempSigningKey}}$.

**Proof.** The proof of this lemma is almost as same as that of Lemma 4.2. The only difference is that the adversary $A_1$ can adaptively query the oracle $O_{\text{HelperKey}}$ on any identity $ID$. When receiving such a query on an identity $ID$, the algorithm $B$ looks up the list KeyList to find the corresponding tuple $(ID, upk_{ID}, usk_{ID}, HK_{ID})$ and returns $HK_{ID}$ to the adversary $A_1$.

**Lemma 4.6.** If $A_2$ is a Type-II adversary against the proposed scheme that has advantage at least $\epsilon$ to win the game $\text{Game-B}$, then there exists an algorithm $B$ to solve the Squ-CDH problem with advantage $\epsilon' \geq \frac{\epsilon}{16(n_u + 1)^2 q_{tsk}^2}$, where $q_{tsk}$ denotes the number of $A_2$’s queries to $O_{\text{TempSigningKey}}$.
Certificateless key-insulated signature 371

Proof. The proof of this lemma is almost as same as that of Lemma 4.3. The only difference is that the adversary $A_2$ can adaptively query the oracle $\mathcal{O}_{\text{HelperKey}}$ on any identity $ID$. When receiving such a query on an identity $ID$, the algorithm $B$ looks up the list $\text{KeyList}$ to find the corresponding tuple $(ID, upk_{ID}, usk_{ID}, HK_{ID})$ and returns $HK_{ID}$ to the adversary $A_2$.

4.3. Comparison. To evaluate the performance of our scheme, we make a comparison between our scheme and Wan et al.’s certificateless strong key-insulated signature scheme [23]. The comparison result is summarized in Table 1. In the computational cost comparison, we consider four operations: pairing, exponentiation in $G_1$, multiplication in $G_1$ and multiplication in $G_2$. For simplicity, we denote these operations by $P$, $E$, $M_1$ and $M_2$ respectively. Note that computing $f_1(\cdot)$ and $f_2(\cdot)$ in Wan et al.’s scheme requires computing $n_u/2$ and $n_m/2$ multiplications in $G_1$ on the average respectively. In the communicational cost comparison, the sizes of the public parameters, public keys and signatures are measured in terms of the number of group elements. In addition, the “Attacks” row indicates that whether the scheme can resist both the key replacement attack and malicious KGC attack presented in Section 3.

<table>
<thead>
<tr>
<th></th>
<th>Wan et al.’s scheme [23]</th>
<th>Our scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign cost</td>
<td>$4E + (n_u/2 + n_m/2 + 3)M_1$</td>
<td>$5E + (n_u/2 + 4)M_1$</td>
</tr>
<tr>
<td>Verify cost</td>
<td>$4P + (n_u + n_m/2)M_1 + 3M_2$</td>
<td>$4P + 1E + (n_u + 1)M_1 + 3M_2$</td>
</tr>
<tr>
<td>Public parameters size</td>
<td>$(n_u + n_m + 5)G_1$</td>
<td>$(n_u + 6)G_1$</td>
</tr>
<tr>
<td>Public key size</td>
<td>$1G_2$</td>
<td>$2G_1 + 1G_2$</td>
</tr>
<tr>
<td>Signature size</td>
<td>$4G_1$</td>
<td>$4G_1$</td>
</tr>
<tr>
<td>Complexity assumption</td>
<td>NGBDH+Many-DH</td>
<td>Squ-CDH $(\Leftrightarrow$CDH)</td>
</tr>
<tr>
<td>Attacks</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 1. Comparison of the certificateless strong key-insulated signature schemes

From the table, we can see that $n_u + n_m + 5$ group elements are needed as public parameters in Wan et al.’s scheme while only $n_u + 6$ group elements are required in our scheme. Therefore, our scheme enjoys a smaller public parameters size, which makes it be more suitable for the low-bandwidth or low-storage applications. Regarding the public key size, a public key in our scheme consists of two additional elements in the group $G_1$ (which are about 342 bits when instantiated in MNT curves with 80 bits security) and thus is a bit longer than the one in Wan et al.’s scheme. However, our scheme provides an effective mechanism to check the public key validity so that it can fight against the public key replacement attack. Our scheme is more efficient than Wan et al.’s scheme in the computation efficiency since $n_m/2$ multiplications in $G_1$ on the average are less needed in both the algorithms Sign and Verify. Furthermore, our scheme is proven secure under the Squ-CDH assumption which is strictly harder than both the NGBDH assumption and the Many-DH assumption on which Wan et al.’s scheme is based. Most importantly,
our scheme can resist both the attacks presented in Section 3 while Wan et al.’s scheme can not.

5. Conclusions

In this paper, we have shown that the previous two certificateless key-insulated signature schemes without random oracles are not secure under either key replacement attack or malicious KGC attack. We propose a new certificateless strong key-insulated signature scheme which can be proven to be strong key-insulated and existentially unforgeable under the Squ-CDH assumption in the standard model. The new scheme overcomes the security weaknesses in previous certificateless key-insulated signature schemes and enjoys higher computational efficiency and shorter public parameters.

Acknowledgements

We would like to thank the anonymous referees for their helpful comments. This work is supported by the National Natural Science Foundation of China grant 61272542, a Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions and the Fundamental Research Funds for the Central Universities.

References


Received August 2014; revised March 2015.

*E-mail address:* luyangnsd@163.com  
*E-mail address:* zhangquanling99@163.com  
*E-mail address:* ljg1688@163.com