# A Forward-Secure Certificate-Based Signature Scheme with Enhanced Security in the Standard Model

**Yang Lu[1] and Jiguo Li[2]**
[1] School of Computer Science and Technology, Nanjing Normal University
Nanjing 210023, China
[e-mail: luyangnsd@163.com]
[2] College of Mathematics and Informatics, Fujian Normal University
Fuzhou 350117, China
[e-mail: ljg1688@163.com]
*Corresponding author: Yang Lu

*Received December 25, 2017; revised August 10, 2018; accepted October 1, 2018; published March 31 2019*

## Abstract

Leakage of secret keys may be the most devastating problem in public key cryptosystems because it means that all security guarantees are missing. The forward security mechanism allows users to update secret keys frequently without updating public keys. Meanwhile, it ensures that an attacker is unable to derive a user's secret keys for any past time, even if it compromises the user's current secret key. Therefore, it offers an effective cryptographic approach to address the private key leakage problem. As an extension of the forward security mechanism in certificate-based public key cryptography, forward-secure certificate-based signature (FS-CBS) has many appealing merits, such as no key escrow, no secure channel and implicit authentication. Until now, there is only one FS-CBS scheme that does not employ the random oracles. Unfortunately, our cryptanalysis indicates that the scheme is subject to the security vulnerability due to the existential forgery attack from the malicious CA. Our attack demonstrates that a CA can destroy its existential unforgeability by implanting trapdoors in system parameters without knowing the target user's secret key. Therefore, it is fair to say that to design a FS-CBS scheme secure against malicious CAs without lying random oracles is still an unsolved issue. To address this problem, we put forward an enhanced FS-CBS scheme without random oracles. Our FS-CBS scheme not only fixes the security weakness in the original scheme, but also significantly optimizes the scheme efficiency. In the standard model, we formally prove its security under the complexity assumption of the square computational Diffie-Hellman problem. In addition, the comparison with the original FS-CBS scheme shows that our scheme offers stronger security guarantee and enjoys better performance.

***Keywords:*** Forward-secure certificate-based signature, existential forgery attack, malicious CA, standard model, existential unforgeability

This work is supported by the National Natural Science Foundation of China (grant Nos. 61772009 and U1736112), and the Natural Science Foundation of Jiangsu Province (grant Nos. BK20181304 and BK20161511).

**http://doi.org/10.3837/tiis.2019.03.022** **ISSN : 1976-7277**

## 1. Introduction

Public key cryptography (PKC) is a crucial cryptographic primitive to accomplish information security. In a public key cryptosystem, each user possesses a public/secret key pair. The public key is usually made public. Anyone can use a user's public key to encrypt some messages for the user or verify the validness of the digital signatures issued by the user. The private key is kept secret. Anyone can decrypt the received ciphertexts or produce the digital signatures on some messages by using his/her private key. The security of a public key cryptosystem is dependent on an assumption that all secret keys have gotten perfect protection. Nevertheless, the cryptographic computations are often made on unprotected or easily stolen devices in the reality. The secret key leakage seems inevitable, because the hackers have a wide variety of means to obtain a secret key from an insecure device. Actually, for a hacker, it is easier to hack into a device to grab a user's secret key than to break the computational complexity problem(s) on which a public key cryptosystem lies. Undoubtedly, secret key disclosure has become the gravest threat on the security of PKC, since it means that all security guarantees are missing.

In order to alleviate the damage caused by private key leakage, a feasible solution is to design the public key cryptosystems with forward security. Forward security provides the benefits of frequently evolving the users' private keys without incurring the costs of changing the public keys. More specifically, in a public key cryptosystem with forward security, the system lifespan (*e.g.*, one year) is split into several time slots (*e.g.*, 365 days). Every user in the system initially produces a public/secret key pair. Then, the user makes his/her public key public and stores his/her initial secret key in a digital device where the cryptographic operations are executed actually. With the update of system time, the user's secret key will evolve periodically. At the beginning of each time slot, the device executes a key-evolving algorithm to derive a new secret key for user from the previous one and then deletes the old key permanently. Meanwhile, the user's public key remains unchanged during the whole life cycle of the system. Forward security guarantees that the leakage of a user's secret key in the current time slot does not make a hacker deduce the user's secret key used in any the past time slot. In this way, the forward security mechanism offers an effective cryptographic solution to address the threat of secret key leakage in public key cryptosystems. The concept of forward-secure PKC was introduced by Anderson [1] in ACM CCS 1997. In Crypto 1999, Bellare and Miner [2] formalized the notion of forward security in the setting of digital signature. They also presented a concrete forward-secure digital signature scheme. In Eurocrypt 2003, Canetti *et al*. [3] proposed a non-interactive forward-secure public key encryption (PKE) scheme. Motivated by the works [2, 3], many forward-secure digital signature schemes [4-10] and forward-secure PKE schemes [11-16] were proposed. However, most of the previous schemes were over either conventional PKC or identity-based cryptography (IBC) [17]. Therefore, they are inevitably subject to either the complicated certificate management or the key escrow issue.

In Eurocrypt 2003, Gentry [18] presented a practical public key cryptographic primitive named certificate-based cryptography (CBC). This new primitive not only greatly simplifies the complicated management of public key certificates in conventional PKC, but also overcomes the key escrow and distribution issues in IBC. In a CBC system, a user needs to produce a pair of public and secret keys independently. Then, he/she submits his/her identity, public key and some other necessary information to a trusted certificate authority (CA) to

apply for a public key certificate. Unlike the certificates in conventional PKC systems, a certificate in CBC is merely pushed to its holder and used as a partial decryption/signing key. As discussed in [18], this interesting property of the certificate offers an implicit authentication function so that a user requires both his/her secret key and certificate to execute the decryption/signing tasks, while the others need not be concerned about his/her certificate status. In this way, CBC avoids the issue of third-party queries for the certificate status and predigests the complicated certificate management in conventional PKI-assisted PKC systems. In addition, CBC also addresses the key escrow and distribution problems. In recent years, CBC has attracted much attention in academic circle and a lot of cryptographic schemes [19-36] (includeing many certificate-based encryption (CBE) and certificate-based signature (CBS) schemes) have been proposed.

To address the key leakage issue in CBC, Lu and Li [37] first extended the forward security mechanism into CBC and proposed the notion of forward-secure CBE (FS-CBE). They also provided a generic method to construct the FS-CBE schemes. Subsequently, Li *et al*. [38] put forward the concept of forward-secure CBS (FS-CBS) along with a FS-CBS scheme without random oracles. In the standard model, Li *et al*. proved the security of their scheme under the complexity assumptions of the many Diffie-Hellman (Many-DH) and the generalized computational Diffie-Hellman (GCDH) problems. In addition, Li *et al*. [39] also presented a much more efficient FS-CBS scheme but in the random oracle model.

### 1.1 Contributions

The goal of this paper is to fix the security vulnerability in the FS-CBS scheme presented by Li *et al*. in [38]. In Li *et al*.'s scheme, the CA is supposed to be honest but curious. More specifically, it is assumed that the CA initializes the system in accordance with the system specifications completely. But, once the system has started running, it gets curious and starts eavesdropping on or impersonating users. Nevertheless, such an assumption does not mirror the reality. In practice, a CA possibly is untrusted and malicious. As shown in [27, 35], it is very likely to compromise a target's message privacy or signature unforgeability without the knowledge of the corresponding secret key by injecting some malicious trapdoors into the system. In this paper, we demonstrate that Li *et al*.'s FS-CBS scheme [38] is insecure under the existential forgery attack from the malicious CA. Our presented attack shows that a CA can easily inject trapdoors into the system and forge valid signatures in the name of any user.

To address this problem, we propose a FS-CBS scheme with enhanced security. The enhanced FS-CBS scheme repairs the security defect in the original scheme and provides immunity against the attacks by the malicious CA. Furthermore, it also significantly optimizes the scheme performance. Our proofs in the standard model demonstrate that it achieves the unforgeability against chosen-message attacks under the complexity assumption of the square computational Diffie-Hellman problem. Compared with Li *et al*.'s FS-CBS scheme [38], the enhanced scheme enjoys three advantages. First of all, it can offer stronger safety guarantee for the practical application since it is immune to the existential forgery attack by the malicious CA. Second, its security is over a complexity problem that is harder than the ones on which Li *et al*.'s scheme is based. Finally yet importantly, it has better efficiency, especially on the communication and storage costs.

### 1.2 Paper organization

In Section 2, some notations and preliminaries are briefly introduced. In Section 3, a malicious CA attack is presented on Li *et al*.'s FS-CBS scheme. In Section 4, an enhanced FS-CBS scheme is proposed. Finally, a conclusion is given in Section 5.

## 2. Background Knowledge

### 2.1 Notations

A list of notations utilized throughout the paper is summarized as below:

– $k$: A security system parameter;
– $p$: A $k$-bit prime number;
– $G_1$, $G_2$: Two order $p$ cyclic groups;
– $e(\cdot,\cdot)$: A bilinear map from $G_1 \times G_1$ to $G_2$;
– $g$: A generator of $G_1$;
– $Z_p^*$: The field of integer numbers modulo $p$;
– *params*: A set of public system parameters;
– *msk*: The CA's master secret key;
– $T$: The number of the system time slots;
– $\omega^j$: The label of the node associated with the time slot $j$;
– $NSK_{\omega^j}$: The node secret key of the node $\omega^j$;
– $d$: Depth of the node that is corresponds to a time slot;
– *ID*: An identity;
– $PK_{ID}$: The public key of a user with identity *ID*;
– $Cert_{ID}$: The certificate of a user with identity *ID*;
– $SK_{ID}^j$: The secret key of a user with identity *ID* that is used in the time slot $j$;
– $H_u, H_v, H$: Three cryptographic hash functions;
– $n_u$, $n_v$: Bit length of a hash value outputted by $H_u$ and $H_v$ respectively.

### 2.2 Bilinear map and computational assumption

Assuming that $e$ is a map from $G_1 \times G_1$ to $G_2$. The map $e$ is called a bilinear pairing if it satisfies: (1) Bilinearity: $\forall x, y \in Z_p^*$, $e(g^x, g^y) = e(g, g)^{xy}$; (2) Non-degeneracy: $e(g, g) \neq 1$; (3) Computability: $\forall x, y \in Z_p^*$, $e(g^x, g^y)$ can be calculated efficiently.

The enhanced FS-CBS scheme is proven secure under the complexity assumption of the square computational Diffie-Hellman (Squ-CDH) problem [40].

**Definition 1.** Given the generator $g$ of the group $G_1$ and $g^x \in G_1$ for unknown value $x \in Z_p^*$, the Squ-CDH problem over $G_1$ is to calculate $g^{x^2} \in G_1$. The Squ-CDH assumption states that for any polynomial-time algorithm $\mathcal{A}$, it has negligible advantage $Adv_{\mathcal{A}_{Squ\text{-}CDH}}(k)$ in resolving the Squ-CDH problem, where $Adv_{\mathcal{A}_{Squ\text{-}CDH}}(k)$ is defined to be the probability $\Pr[\mathcal{A}(g, g^x) = g^{x^2} \mid g \in G_1 \wedge x \in Z_p^*]$.

The equivalence between the Squ-CDH problem and the standard computational Diffie-Hellman (CDH) problem has been demonstrated in [40].

### 2.3 Framework and security definitions of FS-CBS

A FS-CBS scheme comprises six polynomial-time algorithms [38]:

- Setup($k$, $T$): Inputing $k$ and $T$, this algorithm creates the CA's master secret key $msk$ and a set of public system parameters $params$.
- UserKeyGen($params$, $ID$): Inputing $params$ and an identity $ID$, this algorithm creates a public key $PK_{ID}$ and an initial secret key $SK_{ID}^0$ for the user with identity $ID$.
- CertGen($params$, $msk$, $ID$, $PK_{ID}$): Inputing $params$, $msk$, $ID$ and $PK_{ID}$, this algorithm create a certificate $Cert_{ID}$ for the user with identity $ID$.
- KeyEvolve($params$, $j$, $SK_{ID}^j$): Inputing $params$, the index of a time slot $j \in [0, T\text{-}1)$ and a secret key $SK_{ID}^j$ in the time slot $j$, this algorithm creates a secret key $SK_{ID}^{j+1}$ for the following time slot $j + 1$.
- Sign($params$, $j$, $m$, $ID$, $Cert_{ID}$, $SK_{ID}^j$): Inputing $params$, the index of a time slot $j \in [0, T\text{-}1]$, a message $m$ to be signed and a signer's identity $ID$, certificate $Cert_{ID}$ and secret key $SK_{ID}^j$, this algorithm creates a signature $<j, \sigma>$ for the message $m$.
- Verify($params$, $m$, $<j, \sigma>$, $ID$, $PK_{ID}$): Inputing $params$, $m$, $<j, \sigma>$, the signer's identity $ID$ and public key $PK_{ID}$, this algorithm returns 1 if $<j, \sigma>$ is a correct signature on the message $m$ signed by a signer with identity $ID$ and public key $PK_{ID}$ in the time slot $j$ or 0 else.

**Definition 2.** A FS-CBS scheme is correct if for any $m$ and $j \in [0, T\text{-}1]$, $<j, \sigma> \leftarrow$ Sign($params$, $j$, $m$, $ID$, $Cert_{ID}$, $SK_{ID}^j$), then $1 \leftarrow$ Verify($params$, $m$, $<j, \sigma>$, $ID$, $PK_{ID}$), where $params$, $PK_{ID}$, $Cert_{ID}$ and $SK_{ID}^j$ are respectively created in accordance with the algorithm specifications.

As introduced in [38, 39], the adversarial model for FS-CBS should distinguish two different types of adversaries. The Type-I adversary (denoted by $\mathcal{A}_I$) acts as an outside attacker who has not been certified by the CA. This adversary is able to execute public key replacement and query any secret key (including the secret key of the target user), but is prohibited from querying the certificate for the target user. The Type-II adversary (denoted by $\mathcal{A}_{II}$) acts as a malicious CA that possesses the master secret key. This adversary is able to issue a certificate for each user using the master secret key, but is prohibited from querying the secret key of the target user and replacing public keys.

To formalize the security model of FS-CBS, we introduce five oracles. These oracles are controlled by a challenger. The adversaries are allowed to adaptively ask some of these oracles.

- $O^{UserCreate}$: To create a user, the adversary submits an identity $ID$ to the oracle. Input $ID$, this oracle responds with a public key $PK_{ID}$ if $ID$ has been previously created. Otherwise, the oracle first produces a public key $PK_{ID}$ and an initial secret key $SK_{ID}^0$ and then returns $PK_{ID}$. Note that other four oracles merely respond to an identity only if it has been created.
- $O^{PrivateKeyCorrupt}$: To corrupt the secret key of a user in a time slot $j \in [0, T\text{-}1]$, the adversary submits an identity $ID$ and the index of a time slot $j$ to the oracle. Once receiving ($ID$, $j$), this oracle responds with a private key $SK_{ID}^j$ in the time period $j$.
- $O^{Certify}$: To query a user's certificate, the adversary submits an identity $ID$ to the oracle. Once receiving $ID$, this oracle returns a certificate $Cert_{ID}$.
- $O^{PublicKeyReplace}$: To replace a public key, the adversary submits an identity $ID$ and a fake public key $PK_{ID}^f$ to the oracle. Once receiving ($ID$, $PK_{ID}^f$), this oracle replaces the current

public key corresponding to *ID* with $PK_{ID}^{f}$. Note that the adversary might be required to submit the secret value used to generate $PK_{ID}^{f}$ so that the oracles $O^{PrivateKeyCorrupt}$ and $O^{Sign}$ can be correctly simulated.

– $O^{Sign}$: To query the signature on a message issued by a user, the adversary submits an identity *ID*, the index of a time slot $j \in [0, T\text{-}1]$ and a message *m* to the oracle. Once receiving (*ID*, *j*, *m*), this oracle responds with a signature $<j, \sigma>$. This oracle executes as essentially same as the strong-signing oracle $O^{CB\text{-}StrongSign}$ introduced in [32]. Concretely, if the public key corresponding to *ID* is the original one created by $O^{UserCreate}$, this oracle answers in a normal way. Otherwise, namely that the user's public key has been replaced by the adversary, it responds with a signature that is calculated by utilizing the secret key and the certificate corresponding to the false public key $PK_{ID}^{f}$.

A FS-CBS scheme should achieve forward security and existential unforgeability under chosen-message attacks (FS&EUF-CMA). This security notion is defined by a game as described below, in which a game challenger interacts with a Type-I adversary or a Type-II adversary.

– **Setup Phase**. The challenger simulates the algorithm Setup(*k*, *T*) to create (*msk*, *params*). It then supplies the adversary $\mathcal{A}$ with *params* if it is a Type-I adversary or both (*msk*, *params*) if it is is a Type-II adversary.
– **Query-Answer Phase**. The adversary $\mathcal{A}$ is allowed to send queries to the oracles $\{O^{UserCreate}, O^{PrivateKeyCorrupt}, O^{Certify}, O^{PublicKeyReplace}, O^{Sign}\}$ if it is a Type-I adversary or the oracles $\{O^{UserCreate}, O^{PrivateKeyCorrupt}, O^{Sign}\}$ if it is a Type-II adversary.
– **Forge Phase**. The adversary $\mathcal{A}$ submits a forgery $(ID^*, m^*, <j^*, \sigma^*>)$. The adversary $\mathcal{A}$ succeeds if the following conditions are met:
  - Verify(*params*, $m^*$, $<j^*, \sigma^*>$, $ID^*$, $PK_{ID^*}$) = 1;
  - The adversary $\mathcal{A}$ has never submitted $(ID^*, j^*, m^*)$ to the oracle $O^{Sign}$;
  - The adversary $\mathcal{A}$ has never submitted $ID^*$ to the oracle $O^{Certify}$ if it is a Type-I adversary;
  - The adversary $\mathcal{A}$ has never submitted $ID^*$ and any time period $j \in [0, j^*]$ to the oracle $O^{PrivateKeyCorrupt}$ if it is a Type-II adversary.

We define the advantage of the adversary $\mathcal{A}$ in winning the game to be the probability that it forges a legitimate signature.

**Definition 3.** A FS-CBS scheme satisfies the FS&EUF-CMA security if for any polynomial-time algorithm $\mathcal{A}$, it has a negligible advantage to win the above game.

### 2.4 Tree-based key-evolving mechanism

Like most of the previous forward-secure schemes, the FS-CBS scheme presented by Li *et al.* [38] exploits a binary tree structure to evolve the users' secret keys. To construct a scheme with *T* time slots, a key-evolving tree with $\log_2(T+1)-1$ levels should be exploited. In this tree, each node is labeled with a binary number. More specifically, the root node is labeled with an empty symbol $\varepsilon$ and if a node is labeled with a binary number $\omega$, then its left child node and right child node are labeled with $\omega 0$ and $\omega 1$ respectively. In addition, the nodes of the

key-evolving tree are associated with the system time slots in a pre-order style. **Fig. 1** shows a concrete example of how to associate the nodes of a key-evolving tree with 3 levels with the time slots {0,1,…,14}.

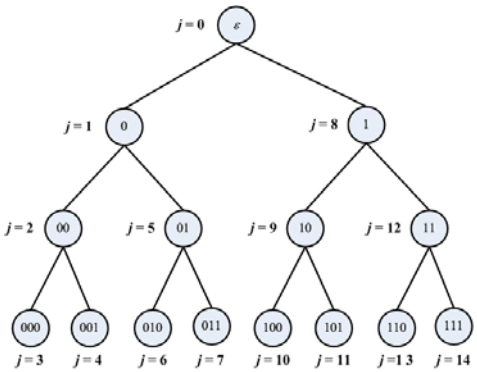


**Fig. 1.** An example of a key-evolving tree with 3 levels

| The private key of a user with identity *ID* in each time period *j* | |
|---|---|
| $SK_{ID}^{0} = \{NSK_{\varepsilon}\}$ | $SK_{ID}^{8} = \{NSK_{1}\}$ |
| $SK_{ID}^{1} = \{NSK_{0}, NSK_{1}\}$ | $SK_{ID}^{9} = \{NSK_{10}, NSK_{11}\}$ |
| $SK_{ID}^{2} = \{NSK_{00}, NSK_{01}, NSK_{1}\}$ | $SK_{ID}^{10} = \{NSK_{100}, NSK_{101}, NSK_{11}\}$ |
| $SK_{ID}^{3} = \{NSK_{000}, NSK_{001}, NSK_{01}, NSK_{1}\}$ | $SK_{ID}^{11} = \{NSK_{101}, NSK_{11}\}$ |
| $SK_{ID}^{4} = \{NSK_{001}, NSK_{01}, NSK_{1}\}$ | $SK_{ID}^{12} = \{NSK_{11}\}$ |
| $SK_{ID}^{5} = \{NSK_{01}, NSK_{1}\}$ | $SK_{ID}^{13} = \{NSK_{110}, NSK_{111}\}$ |
| $SK_{ID}^{6} = \{NSK_{010}, NSK_{011}, NSK_{1}\}$ | $SK_{ID}^{14} = \{NSK_{111}\}$ |
| $SK_{ID}^{7} = \{NSK_{011}, NSK_{1}\}$ | |

**Fig. 2.** An example of a user's secret key in each time slot

In the key-evolving tree, every node $\omega$ is endowed with a secret key $NSK_{\omega}$. Let $\omega^{j}$ be the node corresponding to the time slot $j$. A user secret key in the time slot $j$ is composed of the node secret keys of $\omega^{j}$ and all right siblings of those nodes on the path from the root node to $\omega^{j}$. **Fig. 2** gives a concrete example to show how to create the secret key of a user in each time slot by using a 3-level key-evolving tree.

For easy of description, we represent a user secret key as a stack (referred to as $\mathsf{NSK\text{-}STACK}_{\mathsf{ID}}$). The top of the stack $\mathsf{NSK\text{-}STACK}_{\mathsf{ID}}$ stores the node key of $\omega^{j}$ and the followings are the node keys of all right siblings of those nodes on the route from $\omega^{j}$ to the root.

## 3. Forgery Attack on Li et al.'s FS-CBS Scheme

In this section, we demonstrate that the FS-CBS scheme presented by Li *et al.* in [38] cannot resist the existential forgery attack by the malicious CA.

### 3.1 Description of Li *et al.*'s FS-CBS scheme

Li *et al.*'s FS-CBS scheme is described as follows:

– Setup: This algorithm chooses $\alpha \in Z_p^*$ randomly and sets $g_1 = g^{\alpha}$. It then randomly chooses $g_2 \in G_1$, $\vec{U} = (u', u_1, \ldots, u_{n_u}) \in G_1^{n_u+1}$ and $\vec{V} = (v', v_1, \ldots, v_{n_v}) \in G_1^{n_v+1}$, where $n_u, n_v \in Z^+$.

Furthermore, it chooses two cryptographic hash functions $H_u : G_1 \times G_1 \times \{0,1\}^* \rightarrow \{0,1\}^{n_u}$, $H_v : \{0,1\}^* \rightarrow \{0,1\}^{n_v}$. Finally, it sets *params* = {*l*, $G_1$, $G_2$, *p*, *e*, *g*, $g_1$, $g_2$, $\vec{U}$, $\vec{V}$, $H_u$, $H_v$} and $msk = g_2^{\alpha}$, where $l = \log_2(T + 1) - 1$.

– UserKeyGen: A user randomly selects $x \in Z_p^*$ as his/her initial secret key $SK_{ID}^0$ and calculates his/her public key $PK_{ID} = (PK_{ID,1}, PK_{ID,2}) = (g^x, g_1^x)$. The node key $NSK_{\varepsilon}$ of the root in the key-evolving tree is set to be the user's initial secret key $SK_{ID}^0$.

– CertGen: Let $\tilde{h} = H_u(PK_{ID}, ID)$ and $\tilde{h}[i]$ be the *i*th bit of the hash value $\tilde{h}$. Assuming that $\mathcal{I} \subseteq \{1, 2, ..., n_u\}$ is the set of indices *i* such that $\tilde{h}[i] = 1$. The CA randomly chooses $r_u \in Z_p^*$ and calculates $Cert_{ID} = (Cert_{ID,1}, Cert_{ID,2}) = (g_2^{\alpha} \cdot (u' \prod_{i \in \mathcal{I}} u_i)^{r_u}, g^{r_u})$.

– KeyUpdate: Let $\omega^j = \omega_1 \omega_2 \ldots \omega_d \in \{0,1\}^d$ $(1 \le d \le l)$ be the node corresponding to the time slot *j*. The node key of $\omega^j$ that comprises *d* + 1 elements (including *d* elements in $G_1$ and one element in $Z_p^*$) has the form $NSK_{\omega^j} = (R_{\omega^j|1}, R_{\omega^j|2}, \ldots, R_{\omega^j|d-1}, R_{\omega^j}, SN_{\omega^j})$, where $\omega^j|i$ $(1 \le i \le d\text{-}1)$ is the *i*-length prefix of $\omega^j$. Specially, the secret key of the root node is $NSK_{\varepsilon} = SN_{\varepsilon}$. Inputing the index of a time slot $j \in [0, T\text{-}1)$ and a secret key $SK_{ID}^j$, this algorithm creates a secret key $SK_{ID}^{j+1}$ for the time slot *j* + 1 as follows:

- If $\omega^j$ is an internal node, it pops $NSK_{\omega^{j+1}}$ off the stack NSK-STACK$_{ID}$. Then, it randomly chooses $\rho_0, \rho_1 \in Z_p^*$, computes $R_{\omega^j 0} = g_1^{\rho_0}$, $R_{\omega^j 1} = g_1^{\rho_1}$, $SN_{\omega^j 0} = SN_{\omega^j} + \rho_0$ and $SN_{\omega^j 1} = SN_{\omega^j} + \rho_1$, and sets the secret keys $NSK_{\omega^j 0}$ and $NSK_{\omega^j 1}$ as

$$NSK_{\omega^j 0} = (R_{\omega^j|1}, R_{\omega^j|2}, \ldots, R_{\omega^j|d-1}, R_{\omega^j}, R_{\omega^j 0}, SN_{\omega^j 0}), \quad (1)$$

$$NSK_{\omega^j 1} = (R_{\omega^j|1}, R_{\omega^j|2}, \ldots, R_{\omega^j|d-1}, R_{\omega^j}, R_{\omega^j 1}, SN_{\omega^j 1}). \quad (2)$$

  Finally, it pushes $NSK_{\omega^j 1}$ and $NSK_{\omega^j 0}$ onto the stack NSK-STACK$_{ID}$ respectively. Now, the node keys in the stack compose the secret key $SK_{ID}^{j+1}$.

- Else if $\omega^j$ is a leaf node, it pops $NSK_{\omega^j}$ off the stack NSK-STACK$_{ID}$. Now, the node key on the top of NSK-STACK$_{ID}$ is $NSK_{\omega^{j+1}}$. Then, the remaining node keys in the stack compose the secret key $SK_{ID}^{j+1}$.

– Sign: Let $\omega^j \in \{0,1\}^d$ $(1 \le d \le l)$ be the node corresponding to the time slot *j*. Let $\tilde{m} = H_v(m)$ and $\tilde{m}[i]$ be the *i*-th bit of the hash value $\tilde{m}$. Define $\mathcal{M} \subseteq \{1, 2, ..., n_v\}$ to be the set of indices *i* such that $\tilde{m}[i] = 1$. To sign the message *m* in the time period $j \in [0, T\text{-}1]$, the signer *ID* randomly chooses $r_{\pi}, r_m \in Z_p^*$, pops the node secret key $NSK_{\omega^j} = (R_{\omega^j|1}, R_{\omega^j|2}, \ldots, R_{\omega^j|d-1}, R_{\omega^j}, SN_{\omega^j})$ off the stack NSK-STACK$_{ID}$ and computes

$$(C, R_{\pi}, R_m) = \left( Cert_{ID,1}^{SN_{\omega^j}} \cdot \left( u' \prod_{i \in \mathcal{U}} u_i \right)^{r_{\pi}} \cdot \left( v' \prod_{i \in \mathcal{M}} v_i \right)^{r_m}, Cert_{ID,2}^{SN_{\omega^j}} \cdot g^{r_{\pi}}, g^{r_m} \right). \quad (3)$$

Then, it sets the signature to be $<j, (C, R_\pi, R_m), (R_{\omega^j|1}, R_{\omega^j|2}, \ldots, R_{\omega^j|d-1}, R_{\omega^j})>$.

– Verify: To verify a signature $<j, (C, R_\pi, R_m), (R_{\omega^j|1}, R_{\omega^j|2}, \ldots, R_{\omega^j|d-1}, R_{\omega^j})>$ by using the signer *ID*'s public key $PK_{ID} = (PK_{ID,1}, PK_{ID,2})$, a user tests whether the following equality holds:

$$e(C, g) = e(g_2, PK_{ID,2}) \cdot e\left(g_2, \prod_{\theta=1}^{d} R_{\omega^j|\theta}\right) \cdot e\left(u' \prod_{i \in \mathcal{I}} u_i, R_\pi\right) \cdot e\left(v' \prod_{i \in \mathcal{M}} v_i, R_m\right). \quad (4)$$

The algorithm returns 1 if the equality holds or 0 else.

### 3.2 The presented attack

Li *et al.*'s FS-CBS scheme suffers from the security weankness that a malicious CA is able to fake the signature on any message in the name of any user without knowing the target user's secret key. The malicious CA can do this only by implanting some trapdoors into the system parameters. More specifically, a CA executes such attack in the following way.

– **Implanting trapdoors:** During initializing the system, the malicious CA randomly picks $n_v + 1$ integers $\beta', \beta_1, \ldots, \beta_{n_v} \in Z_p^*$ and calculates $V = (v', v_1, \ldots, v_{n_v}) \in G_1^{n_v+1}$ as follows:

$$v' = g^{\beta'}, v_1 = g^{\beta_1}, \ldots, v_{n_v} = g^{\beta_{n_v}}. \quad (5)$$

Other public system parameters are created according to the specification of the algorithm Setup in Li *et al.*'s FS-CBS scheme.

– **Faking signatures:** To simulate a user with identity *ID*, the malicious CA first gets a message/signature pair $(m, <j, (C, R_\pi, R_m), (R_{\omega^j|1}, R_{\omega^j|2}, \ldots, R_{\omega^j|d-1}, R_{\omega^j})>)$ issued by the user. After that, it selects $r'_m \in Z_p^*$ randomly and creates a new signature $<j, (C', R'_\pi, R'_m), (R_{\omega^j|1}, R_{\omega^j|2}, \ldots, R_{\omega^j|d-1}, R_{\omega^j})>$ on a different message $m' (\neq m)$ in the following way:

$$C' = C \cdot (R_m)^{-(\beta' + \sum_{i \in \mathcal{M}} \beta_i)} \cdot \left(v' \prod_{i \in \mathcal{M}'} v_i\right)^{r'_m}, R'_\pi = R_\pi, R'_m = g^{r'_m}, \quad (6)$$

where $\mathcal{M} = \{i \mid \tilde{m}[i] = 1, \tilde{m} = H_v(m)\}$, $\mathcal{M}' = \{i \mid \tilde{m}'[i] = 1, \tilde{m}' = H_v(m')\}$ and $\tilde{m}[i]$ is the *i*th bit of $\tilde{m}$ and $\tilde{m}'[i]$ is the *i*th bit of $\tilde{m}'$.

According to the above scheme description, if $<j, (C, R_\pi, R_m), (R_{\omega^j|1}, R_{\omega^j|2}, \ldots, R_{\omega^j|d-1}, R_{\omega^j})>$ is a legitimate signature signed on the message *m* by the user *ID*, then

$$(C, R_\pi, R_m) = \left(Cert_{ID,1}^{SN_{\omega^j}} \cdot \left(u' \prod_{i \in \mathcal{U}} u_i\right)^{r_\pi} \cdot \left(v' \prod_{i \in \mathcal{M}} v_i\right)^{r_m}, Cert_{ID,2}^{SN_{\omega^j}} \cdot g^{r_\pi}, g^{r_m}\right). \quad (7)$$

Thus, we have

$$
\begin{aligned}
C' &= C \cdot (R_m)^{-(\beta' + \sum_{i \in \mathcal{M}} \beta_i)} \cdot \left( v' \prod_{i \in \mathcal{M}'} v_i \right)^{r'_m} \\
&= Cert_{ID,1}^{SN_{\omega^j}} \cdot \left( u' \prod_{i \in \mathcal{I}} u_i \right)^{r_\pi} \cdot \left( v' \prod_{i \in \mathcal{M}} v_i \right)^{r_m} \cdot (R_m)^{-(\beta' + \sum_{i \in \mathcal{M}} \beta_i)} \cdot \left( v' \prod_{i \in \mathcal{M}'} v_i \right)^{r'_m} \\
&= Cert_{ID,1}^{SN_{\omega^j}} \cdot \left( u' \prod_{i \in \mathcal{I}} u_i \right)^{r_\pi} \cdot \left( v' \prod_{i \in \mathcal{M}} v_i \right)^{r_m} \cdot \left( v' \prod_{i \in \mathcal{M}} v_i \right)^{-r_m} \cdot \left( v' \prod_{i \in \mathcal{M}'} v_i \right)^{r'_m} \\
&= Cert_{ID,1}^{SN_{\omega^j}} \cdot \left( u' \prod_{i \in \mathcal{I}} u_i \right)^{r_\pi} \cdot \left( v' \prod_{i \in \mathcal{M}'} v_i \right)^{r'_m}.
\end{aligned}
$$

Clearly, the CA's forgery ($m'$, $<j, (C', R'_\pi, R'_m)$, ($R_{\omega^j|1}, R_{\omega^j|2}, \ldots, R_{\omega^j|d-1}, R_{\omega^j}$)>) passes the following signature verification equation

$$
e(C', g) = e(g_2, PK_{ID,2}) e\left( g_2, \prod_{\theta=1}^{d} R_{\omega^j|\theta} \right) e\left( u' \prod_{i \in \mathcal{U}} u_i, R'_\pi \right) e\left( v' \prod_{i \in \mathcal{M}'} v_i, R'_m \right). \tag{8}
$$

Therefore, the CA succeeds in faking a legitimate pair of message and signature in the name of the user *ID*. This implies that the existential unforgeability of the scheme is broken by the CA completely.

## 4. An Enhanced FS-CBS Scheme

Next, we propose an enhanced FS-CBS scheme to fix the security defect in Li *et al.*'s FS-CBS scheme [38]. Our proofs in the standard model demonstate that the enhanced scheme meets the FS&EUF-CMA security under the compexity assumtion of the Squ-CDH problem.

### 4.1 Basic idea

The security weakness in Li *et al.*'s FS-CBS scheme [38] is mainly due to the reason that the component pertaining to the message (*i.e.*, $\left( v' \prod_{i \in \mathcal{M}} v_i \right)^{r_m}$) in a signature is independent on the signer's secret key. Thus, by injecting trapdoors into public system parameters properly, a CA can tamper a signature by replacing $\left( v' \prod_{i \in \mathcal{M}} v_i \right)^{r_m}$ with $\left( v' \prod_{i \in \mathcal{M}'} v_i \right)^{r'_m}$ without affecting the correctness of the signature. To overcome this security weakness, we implant part of the signer's public key (*i.e.*, $PK_{ID,2}$) in the signatures. Specifically, in the enhanced scheme, the component of a signature $<j, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)>$ pertaining to the message (namely $\sigma_3$) is defined to the form $\left( Cert_{ID,2} \right)^{Y_{\omega^j}} \cdot F(\lambda_{ID})^{r'} \cdot \left( PK_{ID,2}^{\phi} \cdot v_\beta \right)^{r}$. If wanting to fake a new signature from a signature $<j, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)>$, the CA needs to remove $\left( PK_{ID,2}^{\phi} \cdot v_\beta \right)^{r}$ from $\sigma_3$. Clearly, it can do this unless it knows either the secret value to generate the signer's public key or the random $r$ chosen by the singer to create the signature$<j, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)>$. Obviously, both of these two values are unkonw to the CA. In this way, the enhanced scheme obtains the

security against the malicious CA attack presented in Subsection 3.2.

Besides the secuirty enhancement, the enhanced scheme also optimizes the system performance. Specifically, it refines the key update algorithm so as to greatly shorten the length of the node keys. In the original scheme, the secret key of a node at depth $d$ consists of $d + 1$ elements (including $d$ elements in $G_1$ and an element in $Z_p^*$), while that in the enhanced scheme only consists of two elements (including an element in $G_1$ and an element in $Z_p^*$). Therefore, the enhanced scheme enjoys the shorter private key/signature length, which leads to the lower storage and communication costs.

### 4.2 Description of the enhanced FS-CBS scheme

The enhanced FS-CBS scheme is composed of the following algorithms:

– Setup($k$, $T$): This algorithm selects an integer $\alpha \in Z_p^*$ randomly and sets $g_1 = g^\alpha$. It also randomly chooses $g_2, v_0, v_1 \in G_1$, a vector $\vec{U} = (u', u_1, \ldots, u_{n_u}) \in G_1^{n_u+1}$ and two hash functions $H$: $\{0,1\}^* \to Z_p^*$ and $H_u$: $\{0,1\}^* \times G_1 \times G_1 \to \{0,1\}^{n_u}$, where $n_u \in Z^+$. Additionally, it defines a two-value function $f$: $G_1 \to \{0,1\}$: for any point $\pi \in G_1$, $f(\pi) = 1$ if the $x$-coordinate of the point $\pi$ is odd or $f(\pi) = 0$ else. Finally, it outputs $params = \{T, G_1, G_2, p, e, g, g_1, g_2, v_0, v_1, \vec{U}, H_u, H, f\}$ and $msk = g_1^\alpha$. For simplicity, for any $n_u$-bit binary string $\lambda = \lambda_1\lambda_2\ldots\lambda_{n_u} \in \{0,1\}^{n_u}$, we define a function $F_u : \{0,1\}^{n_u} \to G_1$ as $F_u(\lambda) = u' \prod_{i=1}^{n_u} u_i^{\lambda_i}$.

– UserKeyGen($params$, $ID$): This algorithm chooses an integer $x \in Z_p^*$ randomly, calculates an initial secret key $SK_{ID}^0 = x^2$ and a public key $PK_{ID}$ = ($PK_{ID,1}$, $PK_{ID,2}$, $PK_{ID,3}$) = $\left(g_1^x, g_2^{1/x}, e(g_1, g_1)^{x^2}\right)$. The node key of the root of the key-evolving tree emplyed by the user $ID$ is set to be $NSK_\varepsilon = SK_{ID}^0$ and pushed into the stack NSK-STACK$_{ID}$. Note that the validity of a public key $PK_{ID}$ = ($PK_{ID,1}$, $PK_{ID,2}$, $PK_{ID,3}$) can be checked by verifying whether the equations $e(PK_{ID,1}, PK_{ID,2}) = e(g_1, g_2)$ and $e(PK_{ID,1}, PK_{ID,1}) = PK_{ID,3}$ hold.

– CertGen($params$, $msk$, $ID$, $PK_{ID}$): This algorithm chooses a random integer $s \in Z_p^*$ and creates a certificate $Cert_{ID}$ = ($Cert_{ID,1}$, $Cert_{ID,2}$) = $(g^s, g_1^\alpha \cdot F(\lambda_{ID})^s)$, where $\lambda_{ID} = H_1(ID, PK_{ID})$.

– KeyUpdate($params$, $j$, $SK_{ID}^j$): Let $\omega^j = \omega_1\omega_2\ldots\omega_h \in \{0,1\}^d$ be the node in the key-evolving tree corresponding to the time slot $j$. The node key of $\omega^j$ has the form $NSK_{\omega^j} = \left(X_{\omega^j}, Y_{\omega^j}\right)$. Specially, the node key corresponding to the root node is $NSK_\varepsilon = Y_\varepsilon$. Given the index of a time slot $j \in [0, T\text{-}1)$ and a secret key $SK_{ID}^j$, this algorithm pops the node key $NSK_{\omega^j}$ off the stack NSK-STACK$_{ID}$ and creates a secret key $SK_{ID}^{j+1}$ for the time slot $j + 1$ as follows:

- If $\omega^j$ is an internal node, then it selects two random integers $t_{\omega^j 0}, t_{\omega^j 1} \in Z_p^*$ and computes the node secret keys $NSK_{\omega^j 0}$ and $NSK_{\omega^j 1}$ for the nodes $\omega^j 0$ and $\omega^j 1$ as follows:

$$NSK_{\omega^j 0} = \left(X_{\omega^j 0}, Y_{\omega^j 0}\right) = \left(X_{\omega^j} \cdot (g_1)^{t_{\omega^j 0}}, Y_{\omega^j} + t_{\omega^j 0}\right), \tag{9}$$

$$NSK_{\omega^j 1} = \left(X_{\omega^j 1}, Y_{\omega^j 1}\right) = \left(X_{\omega^j} \cdot (g_1)^{t_{\omega^j 1}}, Y_{\omega^j} + t_{\omega^j 1}\right). \quad (10)$$

It pushes $NSK_{\omega^j 1}$ and then $NSK_{\omega^j 0}$ onto the stack NSK-STACK$_{ID}$ and sets the node keys in the stack as the secret key $SK_{ID}^{j+1}$. Obviously, the node key of any $\omega^j$ ($j > 0$) has the form $NSK_{\omega^j} = \left(g_1^{t'}, x^2 + t'\right)$ for some $t' \in Z_p^*$.

- Else if $\omega^j$ is a leaf node, it sets the remaining node keys in the stack NSK-STACK$_{ID}$ as the secret key $SK_{ID}^{j+1}$. Now, the node key on the top of NSK-STACK$_{ID}$ is $NSK_{\omega^{j+1}}$.

– Sign(*params, j, m, ID*, $SK_{ID}^j$, $Cert_{ID}$): Assuming that $\omega^j \in \{0,1\}^n$ is the node in the key-evolving tree corresponding to the time slot $j \in [0, T\text{-}1]$. This algorithm selects two random integers $r, r' \in Z_p^*$, retrieves the node key $NSK_{\omega^j} = \left(X_{\omega^j}, Y_{\omega^j}\right)$ from NSK-STACK$_{ID}$ and sets

$$\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4) = \left(g^r, \left(Cert_{ID,1}\right)^{Y_{\omega^j}} \cdot g^{r'}, \left(Cert_{ID,2}\right)^{Y_{\omega^j}} \cdot F(\lambda_{ID})^{r'} \cdot \left(PK_{ID,2}^{\phi} \cdot v_\beta\right)^r, X_{\omega^j}\right), \quad (11)$$

where $\lambda_{ID} = H_1(ID, PK_{ID})$, $\beta = f(\sigma_2)$ and $\phi = H_2(ID, PK_{ID}, \sigma_1, \sigma_2, m, v_\beta)$. Finally, it outputs $<j, \sigma>$ as the signature for the message $m$. Because $Cert_{ID} = (Cert_{ID,1}, Cert_{ID,2}) = (g^s, g_1^\alpha \cdot F(\lambda_{ID})^s)$, we have that

$$\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4) = \left(g^r, g^{\tilde{r}}, (g_1)^{\alpha \cdot Y_{\omega^j}} \cdot F(\lambda_{ID})^{\tilde{r}} \cdot \left(PK_{ID,2}^{\phi} \cdot v_\beta\right)^r, X_{\omega^j}\right), \quad (12)$$

where $\tilde{r} = sY_{\omega^j} + r'$.

– Verify(*params, m*, <*j*, $\sigma$>, *ID*, $PK_{ID}$): To verify a signature $<j, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)>$ by using the signer's public key $PK_{ID} = (PK_{ID,1}, PK_{ID,2}, PK_{ID,3})$, this algorithm verifies if the following equality is satisfied:

$$e(\sigma_3, g) = PK_{ID,3} \cdot e(g_1, \sigma_4) \cdot e\left(F(\lambda_{ID}), \sigma_2\right) \cdot e\left(PK_{ID,2}^{\phi} \cdot v_\beta, \sigma_1\right), \quad (13)$$

where $\lambda_{ID} = H_1(ID, PK_{ID})$, $\beta = f(\sigma_2)$ and $\phi = H_2(ID, PK_{ID}, \sigma_1, \sigma_2, m, v_\beta)$. The algorithm outputs 1 if it does or 0 else.

### 4.3 Correctness and security proofs

First of all, we demonstrate the correctness of the enhanced FS-CBS scheme.

**Theorem 1.** The enhanced FS-CBS scheme is correct.

Proof. Let $<j, \sigma>$ be the signature on a message $m$ signed by a user with identity $ID$. Then, $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4) = \left(g^r, g^{\tilde{r}}, (g_1)^{\alpha \cdot Y_{\omega^j}} \cdot F(\lambda_{ID})^{\tilde{r}} \cdot \left(PK_{ID,2}^{\phi} \cdot v_\beta\right)^r, X_{\omega^j}\right)$, where $r, \tilde{r} \in Z_p^*$, $\lambda_{ID} = H_1(ID, PK_{ID})$, $\beta = f(\sigma_2)$ and $\phi = H_2(ID, PK_{ID}, \sigma_1, \sigma_2, m, v_\beta)$. Because the node key associated with the node $\omega^j$ in the key-evolving tree corresponding to the time slot $j$ has the form $NSK_{\omega^j} =$

$(X_{\omega^j}, Y_{\omega^j}) = \left(g_1^{t'}, x^2 + t'\right)$ for some $t' \in Z_p^*$, we can deduce that

$$\begin{aligned} e(\sigma_3, g) &= e\left((g_1)^{\alpha \cdot Y_{\omega^j}} \cdot F(\lambda_{ID})^{\tilde{r}} \cdot \left(PK_{ID,2}^{\phi} \cdot v_\beta\right)^r, g\right) \\ &= e\left((g_1)^{\alpha \cdot (x^2 + t')}, g\right) \cdot e\left(F(\lambda_{ID})^{\tilde{r}}, g\right) \cdot e\left(\left(PK_{ID,2}^{\phi} \cdot v_\beta\right)^r, g\right) \\ &= e\left(g_1, g_1\right)^{x^2} \cdot e\left(g_1, g_1^{t'}\right) \cdot e\left(F(\lambda_{ID}), g^{\tilde{r}}\right) \cdot e\left(PK_{ID,2}^{\phi} \cdot v_\beta, g^r\right) \\ &= PK_{ID,3} \cdot e\left(g_1, \sigma_4\right) \cdot e\left(F(\lambda_{ID}), \sigma_2\right) \cdot e\left(PK_{ID,2}^{\phi} \cdot v_\beta, \sigma_1\right). \end{aligned}$$

Therefore, the correctness of the scheme is proved. #

The following are the security statement and proof.

**Theorem 2**. The enhanced FS-CBS scheme achieves the FS&EUF-CMA security in the standard model if the Squ-CDH assumption holds in the group $G_1$.

Proof. Assuming that $\mathcal{A}_I$ is a Type-I adversary who has a non-negligible advantage $\varepsilon$ to crack the FS&EUF-CMA security of the enhanced FS-CBS scheme. We show how to construct an algorithm $\mathcal{B}$ who has a non-negligible advantage $\varepsilon' \geq \dfrac{\varepsilon}{8(n_u + 1)(q_C + q_S)}$ to resolve the Squ-CDH problem in $G_1$ by employing the adversary $\mathcal{A}_I$ as a subprocedure, where $q_C$ and $q_S$ are the maximum number of the adversary $\mathcal{A}_I$'s queries submitted to $O^{Certify}$ and $O^{Sign}$ respectively.

Given a random Squ-CDH problem instance ($g$, $A = g^a$) in the group $G_1$, the algorithm $\mathcal{B}$ simulates as a game challenger and plays with the adversary $\mathcal{A}_I$ to compute $g^{a^2}$ in the following way:

– **Setup Phase**. We set $l = 2(q_C + q_S)$. Suppose that $p$ is a large prime number that is greater than $l(n_u + 1)$. The algorithm $\mathcal{B}$ selects integers $\beta', \beta_1, \ldots, \beta_{n_u} \in Z_l$, $\gamma', \gamma_1, \ldots, \gamma_{n_u} \in Z_p$ and $b, c, d_0, d_1 \in Z_p^*$ randomly. Furthermore, it selects a random number $k \in \{0,1,\ldots,n_u\}$. Then, it sets $g_1 = A$, $g_2 = g^b$, $u' = A^{p-lk+\beta'} g^{\gamma'}$, $u_i = A^{\beta_i} g^{\gamma_i}$ ($i = 1,2,\ldots,n_u$), $v_0 = g^{d_0}$ and $v_1 = A^c g^{d_1}$. Other public system parameters are created according to the specification of the algorithm Setup. The above simulation implies that the CA's master secret key is implicitly defined to be $g_1^a = g^{a^2}$ which is unknown to the algorithm $\mathcal{B}$. As a matter of convenience, for any $n_u$-bit binary string $\lambda = \lambda_1 \lambda_2 \ldots \lambda_{n_u} \in \{0,1\}^{n_u}$, we introduce the following two functions:

$$J(\lambda) = (p - lk) + \beta' + \sum_{i=1}^{n_u} \lambda_i \beta_i, \tag{14}$$

$$K(\lambda) = \gamma' + \sum_{i=1}^{n_u} \lambda_i \gamma_i. \tag{15}$$

We can easily derive that $F(\lambda) = u' \prod_{i=1}^{n_u} u_i^{\lambda_i} = A^{J(\lambda)} g^{K(\lambda)}$.

– **Query-Answer Phase.** After receiving the public system parameters *params*, the adversary $\mathcal{A}_I$ sends queries to the oracles {$O^{UserCreate}$, $O^{PrivateKeyCorrupt}$, $O^{Certify}$, $O^{PublicKeyReplace}$, $O^{Sign}$} adaptively. The algorithm $\mathcal{B}$ keeps a list *UserList* containing tuples

($ID_i$, $PK_{ID_i}$ , $x_i$) and a list *PrivateKeyList* containing tuples ($ID_i$, $j$, $SK_{ID_i}^j$ ), and answers the adversary's various oracle queries as below:

- $O^{UserCreate}$: Inputing an identity $ID_i$, the algorithm $\mathcal{B}$ retrieves *UserList* to seek out a tuple ($ID_i$, $PK_{ID_i}$ , $x_i$).
  (a) If such a tuple exists, it returns $PK_{ID_i}$ ;
  (b) Otherwise, it selects a random integer $x_i \in Z_p^*$ and calculates $PK_{ID_i} = \left(g_1^{x_i}, g_2^{1/x_i}, e(g_1, g_1)^{x_i^2}\right)$. Then, it adds ($ID_i$, $PK_{ID_i}$ , $x_i$) to *UserList* and outputs $PK_{ID_i}$ .
- $O^{Certify}$: Inputing an identity $ID_i$, the algorithm $\mathcal{B}$ retrieves *UserList* to seek out a tuple ($ID_i$, $PK_{ID_i}$ , $x_i$) and performs as below:
  (a) If $J(\lambda_{ID_i}) = 0 \pmod p$ where $\lambda_{ID_i} = H_1(ID_i, PK_{ID_i})$ , it terminates the game;
  (b) Else, it selects a random integer $s \in Z_p^*$ and creates a certificate

$$Cert_{ID_i} = \left(Cert_{ID_i,1}, Cert_{ID_i,2}\right) = \left(A^{-1/J(\lambda_{ID_i})} \cdot g^s, A^{-K(\lambda_{ID_i})/J(\lambda_{ID_i})} \cdot F(\lambda_{ID_i})^s\right). \tag{16}$$

  Then, it returns $Cert_{ID_i}$ to the adversary $\mathcal{A}_I$. It is clear that the algorithm $\mathcal{B}$ is able to produce such certificate iff $J(\lambda_{ID_i}) \neq 0 \pmod l$, which also implies $J(\lambda_{ID_i}) \neq 0 \pmod p$. If let $s' = s - a / J(\lambda_{ID_i})$ , then we get

$$\begin{aligned} Cert_{ID_i,1} &= A^{-1/J(\lambda_{ID_i})} \cdot g^s = g^{s'}, \\ Cert_{ID_i,2} &= A^{-K(\lambda_{ID_i})/J(\lambda_{ID_i})} \cdot F(\lambda_{ID_i})^s \\ &= A^a \cdot (A^{J(\lambda_{ID_i})} \cdot g^{K(\lambda_{ID_i})})^{-a/J(\lambda_{ID_i})} \cdot F(\lambda_{ID_i})^s \\ &= g^{a^2} \cdot F(\lambda_{ID_i})^{s'}. \end{aligned}$$

- $O^{PrivateKeyCorrupt}$: Inputing an identity $ID_i$ and the index of a time slot $j$, the algorithm $\mathcal{B}$ retrieves *PrivateKeyList* to seek out a tuple ($ID_i$, $j$, $SK_{ID_i}^j$ ).
  (a) If *PrivateKeyList* has contained such a tuple, it returns $SK_{ID_i}^j$ directly;
  (b) Else if $j$ = 0, it retrieves *UserList* to seek out a tuple ($ID_i$, $PK_{ID_i}$ , $x_i$), computes $SK_{ID_i}^0 = x_i^2$ and then returns $SK_{ID_i}^0$ ;
  (c) Otherwise, it retrieves the list *UserList* to seek out a tuple ($ID_i$, $PK_{ID_i}$ , $x_i$) and creates a secret key $SK_{ID_i,j}$ for the time slot $j$ by performing KeyEvolve(…KeyEvolve(*params*, $x_i^2$ , 0), …, $j$ - 1). Then, it adds ($ID_i$, $j$, $SK_{ID_i}^j$ ) to *PrivateKeyList* and returns $SK_{ID_i}^j$ .
- $O^{PublicKeyReplace}$: Inputing an identity $ID_i$, a false public key $PK_{ID_i}^f$ and a secret value $x_i'$ , the algorithm $\mathcal{B}$ retrieves *UserList* to seek out a tuple ($ID_i$, $PK_{ID_i}$ , $x_i$) and replaces it with ($ID_i$, $PK_{ID_i}^f$ , $x_i'$ ).

- $O^{Sign}$: Inputing an identity $ID_i$, the index of a time slot $j$ and a message $m$, the algorithm $\mathcal{B}$ retrieves *UserList* to seek out a tuple ($ID_i$, $PK_{ID_i}$ , $x_i$) and does the following:
  (a) If $J(\lambda_{ID_i}) = 0 \pmod p$ where $\lambda_{ID_i} = H_1(ID_i, PK_{ID_i})$, then it terminates the game;
  (b) Else, it first derives a certificate $Cert_{ID_i}$ by calling the oracle $O^{Certify}$ and a secret key $SK_{ID_i}^j$ for the time slot $j$ by calling the oracle $O^{PrivateKeyCorrupt}$ respectively. Then, it runs the algorithm Sign(*params*, *j*, *m*, $ID_i$, $SK_{ID_i}^j$, $Cert_{ID_i}$) to generate a signature <*j*, $\sigma$> and outputs the result to the adverary.

– **Forge Phase.** In the end, the adversary $\mathcal{A}_I$ outputs a forgery ($ID^*$, $m^*$, <$j^*$, $\sigma^*$>), where $\sigma^* = \left(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*\right)$. If $J(\lambda_{ID^*}) \neq 0 \pmod p$ or $f(\sigma_2^*) = 1$ where $\lambda_{ID^*} = H_1(ID^*, PK_{ID^*})$, the algorithm $\mathcal{B}$ terminates the game. Else, it first fetches back the secret value $x^*$ from the tuple ($ID^*$, $PK_{ID^*}$ , $x^*$ ) and the secret key $NSK_{\omega^{j^*}}$ = ( $X_{\omega^{j^*}}$ , $Y_{\omega^{j^*}}$ ) from the private key $SK_{ID^*}^{j^*}$ after querying the oracle $O^{PrivateKeyCorrupt}$ on <$ID^*$, $j^*$>. Then, it sets $\phi^*$ = $H_2(ID^*, PK_{ID^*}, \sigma_1^*, \sigma_2^*, m^*, v_0)$, computes

$$T = \left( \frac{\sigma_3^*}{(\sigma_1^*)^{b\cdot\phi^*/x^*} \cdot (\sigma_1^*)^{d_0} \cdot (\sigma_2^*)^{K(\lambda_{ID^*})}} \right)^{\frac{1}{Y_{\omega^{j^*}}}} \tag{17}$$

and then outputs it as the answer to the given Squ-CDH problem.

If <$j^*$, $\sigma^*$> is a legitimate signature under $ID^*$ and $PK_{ID^*} = \left(PK_{ID^*,1}, PK_{ID^*,2}, PK_{ID^*,3}\right)$, then we have that

$$\sigma^* = \left(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*\right) = \left( g^r, g^{\tilde{r}}, (g_1)^{a\cdot Y_{\omega^{j^*}}} \cdot F(\lambda_{ID^*})^{\tilde{r}} \cdot \left(PK_{ID^*,2}^{\phi^*} \cdot v_0\right)^r, X_{\omega^{j^*}} \right), \tag{18}$$

where $r, \tilde{r} \in Z_p^*$ and $\lambda_{ID^*} = H_1(ID^*, PK_{ID^*})$ . In addition, $F(\lambda_{ID^*}) = A^{J(\lambda_{ID^*})} g^{K(\lambda_{ID^*})} = g^{K(\lambda_{ID^*})}$ because $J(\lambda_{ID^*}) = 0 \pmod p$. Thus, we can deduce that

$$T = \left( \frac{(g_1)^{a\cdot Y_{\omega^{j^*}}} \cdot F(\lambda_{ID^*})^{\tilde{r}} \cdot \left(PK_{ID^*,2}^{\phi^*} v_0\right)^r}{\left(g^r\right)^{b\cdot\phi^*/x^*} \cdot \left(g^r\right)^{d_0} \cdot \left(g^{\tilde{r}}\right)^{K(\lambda_{ID^*})}} \right)^{\frac{1}{Y_{\omega^{j^*}}}} = \left( \frac{g^{a^2\cdot Y_{\omega^{j^*}}} \cdot \left(g^{K(\lambda_{ID^*})}\right)^{\tilde{r}} \cdot \left(\left(g^{b/x^*}\right)^{\phi^*} g^{d_0}\right)^r}{\left(g^r\right)^{b\cdot\phi^*/x^*} \cdot \left(g^r\right)^{d_0} \cdot \left(g^{\tilde{r}}\right)^{K(\lambda_{ID^*})}} \right)^{\frac{1}{Y_{\omega^{j^*}}}} = g^{a^2}.$$

Therefore, $T$ is the correct solution to the Squ-CDH problem.

*Analysis*. Next, we derive the bound on the algorithm $\mathcal{B}$'s advantage. Accodrding to the above interaction, the algorithm $\mathcal{B}$ does not terminate the game if the following constraints are satisfied:

– All the adversary $\mathcal{A}_I$'s requests to $O^{Certify}$ satisfy $J(\lambda_{ID_i}) \neq 0 \pmod{p}$;
– All the adversary $\mathcal{A}_I$'s requests to $O^{Sign}$ satisfy $J(\lambda_{ID_i}) \neq 0 \pmod{p}$;
– The adversary $\mathcal{A}_I$'s forgery ($ID^*$, $m^*$, $<j^*, \sigma^*>$) satisfies $J(\lambda_{ID^*}) = 0 \pmod{p}$ and $f(\sigma_2^*) = 0$.

In the adversary $\mathcal{A}_I$'s queries, we assume $\lambda_1, \lambda_2, \ldots, \lambda_{q_h}$ to be the hash values that are not equal to $\lambda_{ID^*}$. Clearly, we have $q_h \leq q_C + q_S$. We define the following events:

– $A^*$: $J(\lambda_{ID^*}) = 0 \pmod{p}$,
– $A_i$: $J(\lambda_{ID_i}) \neq 0 \pmod{p}$, where $i = 1, 2, \ldots, q_h$,
– $B^*$: $f(\sigma_2^*) = 0$.

Clearly, the probability of the event that the algorithm $\mathcal{B}$ does not terminate the game is $\Pr[\neg terminate] = \Pr\left[\bigwedge_{i=1}^{q_h} A_i \wedge A^* \wedge B^*\right]$. The premise $l(n_u + 1) << p$ means that if $J(\lambda_{ID^*}) = 0 \pmod{p}$ then $J(\lambda_{ID^*}) = 0 \pmod{l}$. In addition, this premise also means that if $J(\lambda_{ID^*}) = 0 \pmod{l}$. As a result, there uniquely exists an integer $k$ ($0 \leq k \leq n_u$) such that $J(\lambda_{ID^*}) = 0 \pmod{p}$. Hence, we have

$$\begin{aligned}\Pr[A^*] &= \Pr[J(\lambda_{ID^*}) = 0 \pmod{p} \wedge J(\lambda_{ID^*}) = 0 \pmod{l}] \\ &= \Pr[J(\lambda_{ID^*}) = 0 \pmod{p} \mid J(\lambda_{ID^*}) = 0 \pmod{l}] \cdot \Pr[J(\lambda_{ID^*}) = 0 \pmod{l}] \\ &= \frac{1}{(n_u+1)l}.\end{aligned}$$

Because the events $A^*$ and $A_i$ ($i = 1, 2, \ldots, q_h$) are independent with each other, we get

$$\Pr\left[\bigwedge_{i=1}^{q_h} A_i \wedge A^*\right] = \left(1 - \Pr\left[\bigvee_{i=1}^{q_h} \neg A_i \mid A^*\right]\right) \cdot \Pr[A^*] = \left(1 - \frac{q_h}{l}\right)\frac{1}{(n_u+1)l} \geq \left(1 - \frac{q_C + q_S}{l}\right) \cdot \frac{1}{(n_u+1)l}.$$

Furthermore, $B^*$ and $\bigwedge_{i=1}^{q_h} A_i \wedge A^*$ are also independent with each other and $\Pr[B^*] = 1/2$. Therefore, we get

$$\Pr[\neg terminate] \geq \Pr\left[\bigwedge_{i=1}^{q_h} A_i \wedge A^*\right] \cdot \Pr[B^*] \geq \left(1 - \frac{q_C + q_S}{l}\right) \cdot \frac{1}{(n_u+1)l} \cdot \frac{1}{2} = \frac{1}{8(n_u+1)(q_C+q_S)}.$$

Clearly, if the algorithm $\mathcal{B}$ does not terminate the game, then the adversary $\mathcal{A}_I$ succeeds in forging a valid signature with advantage $\varepsilon$. Thus, the algorithm $\mathcal{B}$'s advantage in successfully resolving the Squ-CDH problem is $\varepsilon' \geq \frac{\varepsilon}{8(n_u+1)(q_C+q_S)}$. This contradicts the Squ-CDH assumption.

In a similar way, we can demonstrate that if there exists a Type-II adversary who has a non-negligible advantage to crack the FS&EUF-CMA security of the enhanced FS-CBS scheme, then a polynomial-time algorithm can be constructed to successfully resolve the Squ-CDH problem with a non-negligible advantage. #

### 4.4 Comparison

Below, we compare the enhanced FS-CBS scheme with Li *et al.*'s scheme [38]. Considering that the FS-CBS scheme in [39] is designed in the random oracle model, we do not include it in the comparison.

**Table 1.** Notations used in the comparison

| Notations | Descriptions |
|---|---|
| $T_B$ | Running time of a pairing operation |
| $T_E$ | Running time of an exponentiation operation in $G_1$ |
| $T_{M1}$ | Running time of a multiplication in $G_1$ |
| $T_{M2}$ | Running time of a multiplication in $G_2$ |
| $\|G_1\|$ | Length of an element in $G_1$ |
| $\|Z_p^*\|$ | Length of an integer in $Z_p^*$ |

**Table 2.** Comparison of two FS-CBS schemes without random oracles

| Compared items | Original scheme [38] | Enhanced scheme |
|---|---|---|
| Complexity problem(s) | GCDH + Many-DH | Squ-CDH |
| Secure against the malicious CA attack? | no | yes |
| Secret key evolving cost | $2T_E$ | $2T_E+2T_{M1}$ |
| Message signing cost | $6T_E+(n_u/2+n_v/2+3)T_{M1}$ | $7T_E+(n_u/2+4)T_{M1}$ |
| Signature verifying cost | $5T_B+(n_u/2+n_v/2+d-1)T_{M1}+3T_{M2}$ | $4T_B+T_E+(n_u/2+1)T_{M1}+3T_{M2}$ |
| Public system parameters size | $(n_u+n_v+5)\|G_1\|$ | $(n_u+6)\|G_1\|$ |
| Signature size | $(d+3)\|G_1\|$ | $4\|G_1\|$ |
| Private key size | $d\|Z_p^*\|+\sum_{i=1}^{d} i\|G_1\|$ | $d\|Z_p^*\|+d\|G_1\|$ |

As listed in **Table 1**, we mainly consider four distinct cryptographic operations in the computation cost comparison, including the bilinear pairing, the multiplication in $G_2$, the exponentiation in $G_1$ and the multiplication in $G_1$, respectively. The running time of hash and integer modular addition are ignored as usual. On average, computing $u'\prod_{i\in\mathcal{U}} u_i$ and $v'\prod_{i\in\mathcal{M}} v_i$ in Li *et al.*'s scheme respectively requires performing $n_u/2$ and $n_v/2$ multiplication operations in $G_1$, while computing $F_u(\lambda)$ in our scheme requires performing $n_u/2$ multiplication operations in $G_1$. We evaluate the computational efficiency of an algorithm by adding the time of the basic operations. As an example, the enhanced FS-CBS scheme needs to calculate 7 exponentiations and $(n_u/2 + 4)$ multiplications in $G_1$ to sign a message. Therefore, the time cost of the algorithm Sign is $7T_E + (n_u/2 + 4)T_{M1}$. In the comparison of communication/storage cost, the size of the public system parameters/a user secret key/a signature is measured by the sizes of involved elements and integers. For instance, a signature in the enhanced FS-CBS scheme comprises 4 elements in $G_1$. Thus, the signature size is $4|G_1|$ bits. In addition, the secret key of root node in Li *et al.*'s scheme consists of one integer in $Z_p^*$,

while the secret key of any other node at depth $d$ comprises one integer in $Z_p^*$ and $d$ elements in $G_1$. Recall that a user secret key used in the time slot $j$ is composed of the node key of $\omega^j$ and all node keys of the right siblings of the nodes on the routine from the root node to $\omega^j$. Therefore, the size of a private key is at most $d \mid Z_p^* \mid + \sum_{i=1}^{d} i \mid G_1 \mid$ bits. **Table 2** shows the details of Li *et al.*'s scheme and the enhanced scheme.

We implement two FS-CBS schemes on a PC that runs Windows 7 (64bit) with Intel(R) Core i7 CPU@2.3GHz and 8GB RAM memory by employing the PBC library [41]. **Table 3** provides the time cost of distinct cryptographic operations and the size of distinct elements. We instantiate the bilinear map using the Type 1 pairing over the elliptic curve $E(F_q)$: $y^2 = x^3 + x$ with embedding degree 2. The group size $q$ is a 512-bit prime satisfying $q + 1 = pr$ and $p$ is a 160-bit Solinas prime number. For ease of comparison, the number of the system time slots $T$ is set to be 127. Therefore, the depth $d$ of the node in the evolving tree associated with the time slot in which the message is signed changes from 1 to 6 (= $\log_2(127 + 1) - 1$). In addition, all cryptographic hash functions in two schemes are simulated by SHA-512. Therefore, the size of a hash value is 512 bits (namely that $n_u$ = 512 and $n_v$ = 512). **Table 4** and **Table 5** respectively show the concrete computation and commnication/storage costs of two schemes corresponding to the different values of the depth $d$.

**Table 3.** Benchmark time of various cryptographic operations and size of various elements

| Operations/Elements | Running Time (ms)/Bit-Length (bit) |
|---|---|
| $T_B$ | 3.296 |
| $T_E$ | 2.757 |
| $T_{M1}$ | 0.013 |
| $T_{M2}$ | 0.003 |
| $\|G_1\|$ | 512 |
| $\| Z_p^* \|$ | 160 |

**Table 4.** Experimental results of the computation costs

| Schemes | Compared items | Computation costs (ms) | | | | | |
|---|---|---|---|---|---|---|---|
| | | $d$ = 1 | $d$ = 2 | $d$ = 3 | $d$ = 4 | $d$ = 5 | $d$ = 6 |
| Li *et al.*'s [38] | Key evolving | 5.514 | | | | | |
| | Signing | 23.237 | | | | | |
| | Verifying | 23.145 | 23.158 | 23.171 | 23.184 | 23.197 | 23.210 |
| Ours | Key evolving | 5.54 | | | | | |
| | Signing | 22.679 | | | | | |
| | Verifying | 19.291 | | | | | |

To shorten the length of node keys, the enhanced FS-CBS scheme has to compute two additional multiplications in $G_1$. Therefore, as shown in **Table 4**, the key evolving operation is less efficient than that of the original scheme. But, the short node keys in the enhanced scheme lead to the better computation efficiency in both the message signing and signature verifying algorithms. **Table 5** indicates that it enjoys better performance in the communication and

storage efficiency than the original scheme. Moreover, the security of the enhanced scheme is over the complexity assumption of the Squ-CDH problem that is harder than the GCDH and Many-DH problems on which Li *et al.*'s scheme is based. At last and most importantly, the enhanced scheme provides much stronger security guarantee, because it resists the malicious CA attack while Li *et al.*'s scheme does not.

**Table 5.** Experimental results of the communication and storage costs

| Schemes | Compared items | Communication and storage costs (bit) | | | | | |
|---|---|---|---|---|---|---|---|
| | | $d = 1$ | $d = 2$ | $d = 3$ | $d = 4$ | $d = 5$ | $d = 6$ |
| Li *et al.*'s [38] | Public parameters | 526848 | | | | | |
| | Signature | 2048 | 2560 | 3072 | 3584 | 4096 | 4608 |
| | Private key | 672 | 1856 | 3552 | 5760 | 8480 | 11712 |
| Ours | Public parameters | 265216 | | | | | |
| | Signature | 2408 | | | | | |
| | Private key | 672 | 1344 | 2016 | 2688 | 3360 | 4032 |

## 5. Conclusions

In this paper, we demonstrate that the FS-CBS scheme presented by Li *et al.* [38] can not resist the existential forgery attack from the malicious CA. To address this issue, we put forward an enhanced FS-CBS scheme and formally prove it to satisfy the existential unforgeability under the complexity assumption of the Squ-CDH problem in the standard model. The enhanced scheme repaires the defect in Li *et al.*'s FS-CBS scheme and provides resistance against the attacks by the malicious CA. Comparisons show that it offers stronger security guarantee and enjoys better performance than the original scheme.

## References

[1] R. Anderson, "Two Remarks on public key cryptology," in *Proc. of ACM CCS 1997*, invited lecture, April 1-4, 1997. Article (CrossRef Link).

[2] M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in *Proc. of Crypto 1999*, pp. 431-448, August 15-19, 1999. Article (CrossRef Link).

[3] R. Canetti, S. Halevi and J. Katz, "A forward-secure public-key encryption scheme," in *Proc. of Eurocrypt 2003*, pp. 255-271, May 4-8, 2003. Article (CrossRef Link).

[4] M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," in *Proc. of Asiacrypt 2000*, pp. 116-129, December 3-7, 2000. Article (CrossRef Link).

[5] G. Itkis and L. Reyzin, "Forward-secure signatures with optimal signing and verifying," in *Proc. of Crypto 2001*, pp. 499-514, August 19-23, 2001. Article (CrossRef Link).

[6] T. Malkin, D. Micciancio, S. K. Miner, "Efficient generic forward-secure signatures with an unbounded number of time periods," in *Proc. of Eurocrypt 2002*, pp. 400-417, April 28 - May 2, 2002. Article (CrossRef Link).

[7] B. Libert, J. Quisquater, M. Yung, "Forward-secure signatures in untrusted update environments," in *Proc. of ACM CCS 2007*, pp. 266-275, Oct 29 - Nov 2, 2007. Article (CrossRef Link).

[8] J. Yu, R. Hao, F. Kong, X. Cheng, J. Fan and Y. Chen, "Forward-secure identity-based signature: security notions and construction," *Information Sciences*, vol. 181, no. 3, pp. 648-660, February, 2011. Article (CrossRef Link).

[9] J. Wei, W. Liu and X. Hu, "Forward-secure identity-based signature with efficient revocation," *International Journal of Computer Mathematics*, vol. 94, no. 7, July, 2016. Article (CrossRef Link).
[10] J. Yu, H. Xia, H. Zhao, R. Hao, Z. Fu and X. Cheng, "Forward-secure identity-based signature scheme in untrusted update environments," *Wireless Personal Communications*, vol. 86, no. 3, pp. 1467-1491, February, 2016. Article (CrossRef Link).
[11] D. Yao, N. Fazio, Y. Dodis, A. Lysyanskaya, "ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption," in *Proc. of ACM CCS 2004*, pp. 354-363, October 25 - 29, 2004. Article (CrossRef Link).
[12] Y. Lu and J. Li, "A practical forward-secure public-key encryption scheme," *Journal of Networks*, vol. 6, no. 9, pp. 1254-1261, June, 2011. Article (CrossRef Link).
[13] J. Yu, F. Kong, X. Cheng, R. Hao and J. Fan, "Forward-secure identity-based public-key encryption without random oracles," *Fundamenta Informaticae*, vol. 111, no. 2, pp. 241-256, February, 2011. Article (CrossRef Link).
[14] K. Singh and N. Trichy, "Lattice forward-secure identity based encryption scheme," *Journal of Internet Services and Information Security*, vol. 2, no. 3/4, pp. 118-128, April, 2012. Article (CrossRef Link).
[15] Y. Lu and J. Li, "New forward-secure public-key encryption without random oracles," *International Journal of Computer Mathematics*, vol. 90, no. 12, pp. 2603-2613, December, 2013. Article (CrossRef Link).
[16] Y. Lu and J. Li, "Forward-secure identity-based encryption with direct chosen-ciphertext security in the standard model," *Advances in Mathematics of Communications*, 2017, vol. 11, vol. 1, pp. 161-177, March, 2017. Article (CrossRef Link).
[17] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. of Crypto 1984*, pp. 47-53, August 19-22, 1984. Article (CrossRef Link).
[18] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in *Proc. of Eurocrypt 2003*, pp. 272-293, May 4-8, 2003. Article (CrossRef Link).
[19] W. Wu, Y. Mu, W. Susilo, X. Huang and L. Xu, "A provably secure construction of certificate-based encryption from certificateless encryption," *The Computer Journal*, vol. 55, no. 10, pp. 1157-1168, January, 2012. Article (CrossRef Link).
[20] D. Galindo, P. Morillo and C. Ràfols, "Improved certificate-based encryption in the standard model," *Journal of Systems and Software*, vol. 81, no. 7, pp. 1218-1226, July, 2008. Article (CrossRef Link).
[21] J. K. Liu and J. Zhou, "Efficient certificate-based encryption in the standard model," in *Proc. of SCN 2008*, pp. 144-155, September 10-12, 2008. Article (CrossRef Link).
[22] Y. Lu and J. Li, "Efficient construction of certificate-based encryption secure against public key replacement attacks in the standard model," *Journal of Information Science and Engineering*, vol. 30, no. 5, pp. 1553-1568, September, 2014. Article (CrossRef Link).
[23] Q. Yu, J. Li and Y. Zhang, "Leakage-resilient certificate-based encryption," *Security and Communication Networks*, vol. 8, no, 18, pp. 3346-3355, May, 2015. Article (CrossRef Link).
[24] Y. Lu and Q. Zhang, "Enhanced certificate-based encryption scheme without bilinear pairings," *KSII Transactions on Internet and Information Systems*, vol. 10, no. 2, pp. 881-896, February, 2016. Article (CrossRef Link).
[25] Q. Yu, J. Li, Y. Zhang, W. Wu, X. Huang and Y. Xiang, "Certificate-based encryption resilient to key leakage," *Journal of Systems and Software*, vol. 116, pp. 101-112, June, 2016. Article (CrossRef Link).
[26] J. Li, Y. Guo, Q. Yu, Y. Lu, Y. Zhang, F. Zhang, "Continuous leakage-resilient certificate-based encryption," *Information Sciences*, vol. 355-356, pp. 1-14, August, 2016. Article (CrossRef Link).
[27] Y. Lu and J. Li, "A provably secure certificate-based encryption scheme secure against malicious CA attacks in the standard model," *Information Sciences*, vol. 372, pp. 745-757, December, 2016. Article (CrossRef Link).

[28] Y. Lu and J. Li, "A pairing-free certificate-based proxy re-encryption scheme for secure data sharing in public clouds," *Future Generation Computer Systems*, vol. 62, pp. 140-147, September, 2016. Article (CrossRef Link).
[29] B. G. Kang, J. H. Park and S. G. Hahn, "A certificate-based signature scheme," in *Proc. of Topics in Cryptology - CT-RSA 2004*, pp. 99-111, February 23-27, 2004. Article (CrossRef Link).
[30] M.H. Au, J.K. Liu, W. Susilo and T.H. Yuen, "Certificate based (linkable) ring signature," in *Proc. of ISPEC 2007*, pp. 79-92, May 7 - 10, 2007. Article (CrossRef Link).
[31] W. Wu, Y. Mu, W. Susilo, X. Huang, "Certificate-based signatures, revisited," *Journal of Universal Computer Science*, vol. 15, no. 8, pp. 1659-1684, April, 2009. Article (CrossRef Link).
[32] J.K. Liu, F. Bao and J. Zhou, "Short and efficient certificate-based signature," in *Proc. of Networking 2011 Workshops*, pp. 167-178, May 13, 2011. Article (CrossRef Link).
[33] J. Li, X. Huang, Y. Zhang, L. Xu, "An Efficient short certificate-based signature scheme," *Journal of Systems and Software*, vol. 85, no. 2, pp. 314-322, February, 2012. Article (CrossRef Link).
[34] J. Li, Z. Wang and Y. Zhang, "Provably secure certificate-based signature scheme without pairings," *Information Science*, vol. 233, pp. 313-320, June, 2013. Article (CrossRef Link).
[35] Y. Lu and J. Li, "An improved certificate-based signature scheme without random oracles," *IET Information Security*, vol. 10, no. 2, pp. 80-86, February, 2016. Article (CrossRef Link).
[36] Y. Lu, Jiguo Li and Jian Shen, "Weakness and improvement of a certificate-based key-insulated signature in the standard model," *The Computer Journal*, vol. 60, no. 12, pp. 1729-1744, December, 2017. Article (CrossRef Link).
[37] Y. Lu and J. Li, "Forward-secure certificate-based encryption and its generic construction," *Journal of Networks*, vol. 5, no. 5, pp. 527-534, May, 2010. Article (CrossRef Link).
[38] J. Li, Y. Zhang, H. Teng, "A forward-secure certificate-based signature scheme in the standard model," in *Proc. of CSS 2012*, pp. 362-376, December 12-13, 2012. Article (CrossRef Link).
[39] J. Li, H. Teng, X. Huang, Y. Zhang and J. Zhou, "A forward-secure certificate-based signature scheme," *The Computer Journal*, vol. 58, no. 4, pp. 853-866, April, 2015. Article (CrossRef Link).
[40] F. Zhang, R. Safavi-Naini and W. Susilo, "An efficient signature scheme from bilinear parings and its applications," in *Proc. of PKC 2004*, pp. 277-290, March 1-4, 2004. Article (CrossRef Link).
[41] B. Lynn, "PBC library: The pairing-based cryptography library," http://crypto.stanford.edu/pbc/. Article (CrossRef Link).

**Yang Lu** received the Ph.D. degree in computer science from PLA University of Science and Technology, Nanjing, China, in 2009. He is currently a professor with the School of Computer Science and Technology, Nanjing Normal University, Nanjing, China. His major research interests include information security and cryptography, network security and cloud security, etc. He has published more than 60 scientific papers in international conferences and journals.

**Jiguo Li** received the Ph.D. degree from Harbin Institute of Technology in 2003. He is currently a professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China. His research interests include cryptography and information security, cloud computing, wireless security and trusted computing etc. He has published over 100 research papers in refereed international conferences and journals. His work has been cited more than 2000 times at Google Scholar.