Provably Secure Forward Secure Certificateless Proxy Signature Scheme

Jiguo Li, Yanqiong Li and Yichen Zhang
College of Computer and Information Engineering, Hohai University
Nanjing 210098, China
[e-mail: ljg1688@163.com,lijiguo@hhu.edu.cn]
*Corresponding author: Jiguo Li

Received May 21, 2013; revised June 20, 2013; accepted July 20, 2013; published August 30, 2013

Abstract

In order to deal with key exposure problem, we introduce forward secure technique into certificateless proxy signature scheme, and propose the formal definition and security model of the forward secure certificateless proxy signature. Our security model takes into account the super adversary in certificateless signature. Furthermore, we present a construction of forward secure certificateless proxy signature scheme with bilinear maps. Based on the difficulty of computational Diffie-Hellman problem, we prove the scheme is secure against chosen message attack in the random oracle model. Finally, we analyze efficiency of the proposed scheme.

Keywords: forward secure, certificateless proxy signature, computational Diffie-Hellman problem, random oracle model

A preliminary version of this paper appeared in NSS 2013, June 3-4, 2013, Madrid, Spain. This version includes motivations and contributions, complete security proof and efficiency analysis. This research was supported by the National Natural Science Foundation of China (60842002, 61272542, 61103183, 61103184), the Fundamental Research Funds for the Central Universities (2013B07014, 2010B07114), China Postdoctoral Science Foundation Funded Project under Grant No. 20100471373, the Six Talent Peaks Program of Jiangsu Province of China (2009182) and Program for New Century Excellent Talents in Hohai University.

http://dx.doi.org/10.3837/tiis.2013.08.013

1. Introduction

In 1996, Mambo, Usuda and Okamoto [1] firstly proposed a proxy signature scheme, in which the proxy signer can sign the message on behalf of the original signer. Proxy signature has a lot of practical applications, such as mobile communications, distributed system and electronic auction, etc. In order to satisfy different situations, researchers proposed many extensions of proxy signature, including designed verifier proxy signature [2], ID-based proxy signature [3], one-time proxy signature [4] and so on. However, we find that most of the proxy signatures are proposed in the identity based cryptography (IBC) setting or the traditional public key cryptography (PKC) setting. We know that IBC has key escrow problem while PKC has certificate management problem.

In order to solve the above problems, Al-Riyami and Paterson [5] firstly presented the certificateless public key cryptography (CL-PKC). Compared with PKC, CL-PKC does not require any certificates to ensure the authenticity of public keys. Moreover, CL-PKC overcomes the key escrow problem of IBC. Due to its advantage, many researchers have been interested in CL-PKC and some certificateless proxy signatures [6-9] have been proposed.

Nowadays, with the rapid development of computer and network, a growing number of cryptographic schemes are applied to unprotected environments such as mobile devices, thus the key exposure problem becomes more and more serious. Once the secret key is exposed, the security of the system will be lost. In order to minimize the damage caused by key exposure, the concept of forward security was put forth by Anderson in [10].

2. Related Work

In 1997, Anderson [10] firstly introduced the concept of forward security in digital signature. In the forward secure signature scheme, the public key keeps unchanged in all time periods, while the private key updates in every time period. By this means, the past signature is still valid even if the private key in a time period is exposed. In 1999, Bellare and Miner [11] proposed a concrete construction of forward secure signature scheme. In their construction, they use the numbers of a binary tree to denote the total life time periods of the scheme. In every time period, the scheme derives a new private key from the private key of the last time period and uses it to sign messages. The public key remains unchanged in all time periods. In 2001, Itkis and Reyzin [12] put forward another forward secure signature scheme. The scheme can be verified effectively, but costs much time on private key generation and update algorithm. In 2002, Malkin et al. [13] proposed a forward secure signature in which the total number of time periods does not have to be fixed in advance. Moreover, the scheme can be based on any underlying signature scheme, and does not rely on specific assumptions. In 2004, Kang et al. [14] presented two forward secure signature schemes based on Diffie-Hellman group. The scheme is superior to Bellare and Miner's scheme in the time of key generation and update, and the time is fixed within the total time periods. In 2008, Alomair et al. [15] proposed a generic method to construct forward secure signature schemes from standard signature schemes. They also defined two notions of forward secure proxy signatures. Yu et al. [16] put forward another forward secure signature scheme with bilinear maps. The scheme requires few pairing operations in verification algorithm. Based on CDH problem, they proved the scheme was secure in the random oracle model. In 2009, Nakanishi et al. [17] proposed a forward secure group signature scheme based on the group signature with pairing, in which the complexity of the key update and signature verification is $O(\log T)$. At present, many scholars

and experts have proposed effective forward secure signature schemes [18-22]. Buchmann et al. [20,21] proposed an efficient post-quantum forward secure signature scheme with minimal security assumptions. Furthermore, they presented the first implementation of a forward-secure signature scheme on a smart card, which solved the problem of on-card key generation and reduced the key generation time. Abdalla et al. [22] proposed a forward-secure signature scheme with tighter reductions. They showed that the tighter security reductions provided by their proof methodology could result in concrete efficiency gains in practice.

Motivations and Contributions. In Asiacrypt 2003, Al-Riyami and Paterson [5] proposed the concept of certificateless signature. The advantage of certificateless public key cryptography successfully eliminates the necessity of certificates in the traditional public key cryptography and simultaneously solves the inherent key escrow problem suffered in identity based cryptography. Therefore, it is a very interesting topic to construct certificateless proxy signature.

It is obviously that there also exists key exposure problem in certificateless proxy signature. If the secret key of proxy signer is exposed, a malicious user can forge proxy signature instead of proxy signer, which will damage the benefit of original signer. Furthermore, all signatures that have ever been generated by proxy signer become invalid. Forward secure signature can guarantee that even if the current time period secret key is exposed, it would not affect the validity of previous signatures, which reduces the impact of key exposure. Based on this idea, Chen et al. [23] firstly proposed a forward secure certificateless proxy signature scheme in 2009, but they didn't give the formal definition and security model of the scheme. Moreover, they didn't prove the security of the scheme. In 2010, Chen et al. [24] proposed another forward secure certificateless proxy signature scheme, but they also didn't give concrete security proof. In this paper, we firstly introduce the formal definition and security model of the scheme, and propose a new forward secure certificateless proxy signature with bilinear maps. The security of our scheme is reduced to computational Diffie-Hellman problem. The scheme not only avoids certificate management problem, but also solves key escrow problem. At the same time, the scheme has forward secure property, which can solve the key exposure problem in certificateless proxy signature. Furthermore, we analyze efficiency of our scheme.

3. Preliminaries

3.1 Bilinear Pairing

Let G_1 denote an additive group of prime order q and G_2 be a multiplicative group of the same order. Let P denote a generator in G_1. Let $e: G_1 \times G_1 \to G_2$ be a bilinear mapping with the following properties:

— Bilinear: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$, $a, b \in \mathbb{Z}_q^*$.

— Non-degenerate: $e(P, Q) \neq 1$ for $P, Q \in G_1$.

— Computable: There is an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in G_1$.

Definition 1 Computational Diffie-Hellman (CDH) Problem
Let G_1 denote an additive group of prime order q, and P denote a generator in G_1. Given (P, aP, bP), for some $a, b \in \mathbb{Z}_q^*$, compute abP.

The success probability of any probabilistic polynomial-time algorithm A solving CDH problem in G_1 is defined to be:

$$Succ_{A,G_1}^{CDH} = \Pr[A(P, aP, bP) = abP : a, b \in \square_q^*].$$

The CDH assumption states that for every probabilistic polynomial-time algorithm A, $Succ_{A,G_1}^{CDH}$ is negligible.

4. The Concept and Security of Forward Secure Certificateless Proxy Signature

4.1 The Concept of Forward Secure Certificateless Proxy Signature

A forward secure certificateless signature scheme is defined by nine probabilistic polynomial-time algorithms: Setup, User-Key-Generate, Partial-Private-Key-Extract, Partial-Proxy-Key-Generate, Partial-Proxy-Key-Verify, Initial-Proxy-Key-Generate, Proxy-Key-Update, Proxy-Sign and Proxy-Verify.

— **Setup:** Takes a security parameter k and a total numbers of time periods N as input, it returns the master key s and system public parameters $params$. This algorithm is run by key generation center (KGC).

— **User-Key-Generate:** Takes system public parameters $params$, a user's identifier ID as input, it outputs a secret value x_{ID} and a public key P_{ID}. This algorithm is run by the original signer *A* and the proxy signer *B*.

— **Partial-Private-Key-Extract:** Takes the master key s and a user's identifier ID and public key P_{ID} as input, it outputs a partial private key D_{ID}. This algorithm is run by KGC.

— **Partial-Proxy-Key-Generate:** Takes system public parameters $params$, a warrant m_w, an original signer's identity ID_A, partial private key D_A, secret value x_A and public key P_A as input, it outputs a partial proxy key σ_A. This algorithm is run by the original signer *A*.

— **Partial-Proxy-Key-Verify:** Takes system public parameters $params$, a warrant m_w, an original signer's identity ID_A and public key P_A, and a partial proxy key σ_A as input, it returns true if the partial proxy key σ_A is correct or false otherwise. This algorithm is run by a proxy signer *B*.

— **Initial-Proxy-Key-Generate:** Takes system public parameters $params$, a warrant m_w, a partial proxy key σ_A, an original signer's identity ID_A and public key P_A, a proxy signer's identity ID_B, partial private key D_B, secret value x_B and public key P_B as input, it outputs a initial proxy key σ_B^0. This algorithm is run by a proxy signer *B*.

— **Proxy-Key-Update:** Takes system public parameters $params$, the current time period $t(t > 0)$, a warrant m_w, a partial proxy key σ_A, an original signer's identity ID_A and public key P_A, a proxy signer's identity ID_B, partial private key D_B, secret value x_B and public key P_B, a proxy key σ_B^{t-1} of last time period as input, it outputs the proxy key σ_B^t of current time period, and deletes the proxy key σ_B^{t-1} of last time period completely. This algorithm is run by a proxy signer *B*.

— **Proxy-Sign:** Takes system public parameters $params$, the current time period t, a message m, a warrant m_w, a proxy signer's identity ID_B and public key P_B, a proxy key

σ_B^t as input, it outputs a proxy signature ψ. This algorithm is run by a proxy signer *B*.

— **Proxy-Verify:** Takes system public parameters $params$, a message m, a warrant m_w, an identity ID_A and public key P_A of the original signer *A*, an identity ID_B and public key P_B of the proxy signer *B*, the current time period t and a proxy signature ψ as input, it returns true if the signature ψ is correct or false otherwise. This algorithm is run by a verifier.

Correctness. We show that any forward secure certificateless signature generated by algorithm **Proxy-Sign** will pass through the verification in algorithm **Proxy-Verify**.

In our construction, proxy signer uses an updated proxy key σ_B^t to generate forward secure certificateless signature which correctness is ensured by that of the any secure forward secure signature scheme. Proxy key σ_B^t can be produced by $\sigma_B^t \leftarrow$ **Proxy-Key-Update**$(Params, t, m_w, \sigma_A, ID_A, P_A, ID_B, x_B, D_B, P_B, \sigma_B^{t-1})$ and $\sigma_B^0 \leftarrow$ **Initial-Proxy-Key-Generate**$(Params, m_w, \sigma_A, ID_A, P_A, ID_B, x_B, D_B, P_B)$, which correctness is essentially deduced to that of a partial proxy key σ_A and partial private key D_B. The correctness of σ_A and D_B can be guaranteed by that of the any secure certificateless signature scheme. If only every entity in our scheme correctly performs signature step. Then, for any signature output by algorithm Proxy-Sign defined in our construction, the algorithm Proxy-Verify will always output true. That is, **Proxy-Verify**$(params, m, m_w, \psi, t, ID_A, P_A, ID_B, P_B) = true$ whenever for all $k \in N$, $m \in \{0,1\}^*$, $ID \in \{0,1\}^*$, where $\psi \leftarrow$ **Proxy-Sign**$(Params, t, m, m_w, ID_B, P_B, \sigma_B^t)$, $(params, s) \leftarrow$ **Setup**(k), $D_{ID} \leftarrow$ **Partial-Private-Key-Extract**$(Params, s, ID)$, $(x_{ID}, P_{ID}) \leftarrow$ **User-Key-Generate**$(Params, ID)$.

4.2 The Security Model of Forward Secure Cerificateless Proxy Signature

According to the security model of forward secure signature scheme proposed in [11] and the super adversary security model of cerificateless signature scheme defined in [25], we put forward the security model of forward secure certificateless proxy signature.

There are two types of adversary in forward secure certificateless proxy signature: A_I and A_{II}. A_I simulates malicious users (anyone except the KGC) who have the following capabilities. (1) A_I does not know the master key, but A_I can replace any user's public key. (2) A_I can get any proxy signer's message/proxy signature in any time period t ($0 \le t \le T$, T is the key exposure time period). A_{II} simulates a dishonest KGC who has the following capabilities. (1) A_{II} knows the master key, but A_{II} cannot replace the target user's public key. (2) A_{II} can get any proxy signer's message/proxy signature in any time period t ($0 \le t \le T$, T is the key exposure time period).

We define the security model of forward secure certificateless proxy signature scheme by using the game between a challenger *C* and an adversary $\Gamma \in \{A_{\mathrm{I}}, A_{\mathrm{II}}\}$ as follows.

Game 1:

Setup: *C* runs the algorithm to generate master key s and system parameters $params$, *C* then sends $params$ to A_I while keeping master key secret.

Chosen Message Attack Phase: In this phase, the challenger *C* keeps a list $L_1 = \{ID_i, x_i, P_i, D_i, \lambda_i\}$ to store the user's secret value, public key and partial private key. The list is initially empty, in which $\lambda_i = 0$ expresses the user's public key has not been replaced,

$\lambda_i = 1$ expresses the user's public key has been replaced. A_I can perform the following queries:

— **Public-Key-Query:** On receiving a user's identity ID_i, the challenger C firstly checks whether the ID_i has been created in L_1 list. If not, C runs User-Key-Generate and Partial-Private-Key-Extract algorithms to generate the secret value x_i, public key P_i and partial private key D_i. Then it adds $\{ID_i, x_i, P_i, D_i, 0\}$ into the L_1 list and returns P_i to A_I. Otherwise, it returns P_i to A_I directly.

— **Partial-Private-Key-Query:** On receiving an identity ID_i, C checks whether the ID_i has been created in L_1 list. If not, C runs User-Key-Generate and Partial-Private-Key-Extract algorithms to generate the secret value x_i, public key P_i and partial private key D_i. Then it adds $\{ID_i, x_i, P_i, D_i, 0\}$ into the L_1 list and returns D_i to A_I. Otherwise, it returns D_i to A_I directly.

— **Public-Key-Replacement-Query:** On receiving an identity ID_i and a public key $P_i^{'}$, C checks whether the ID_i has been created in L_1 list. If so, C updates $\{ID_i, x_i, P_i, D_i, 0\}$ with $\{ID_i, \perp, P_i^{'}, D_i, 1\}$. Otherwise, C adds $\{ID_i, \perp, P_i^{'}, \perp, 1\}$ into L_1 list.

— **Secret-Value-Query:** On receiving an identity ID_i, C checks whether the ID_i has been created in L_1 list. If not, C runs User-Key-Generate and Partial-Private-Key-Extract algorithms to generate the secret value x_i, public key P_i and partial private key D_i. Then it adds $\{ID_i, x_i, P_i, D_i, 0\}$ into the L_1 list and returns x_i to A_I. Otherwise, if $\lambda_i = 0$, C returns x_i to A_I. if $\lambda_i = 1$, C returns $\perp$.

— **Partial-Proxy-Key-Query:** On receiving a warrant m_w and an identity ID_A of an original signer A, C runs Partial-Proxy-Key algorithm to generate a partial proxy key σ_A, and returns it to A_I.

— **Proxy-Key-Query:** On receiving a time period t, a warrant m_w, an identity ID_A of an original signer A and an identity ID_B of a proxy signer B, if $t = 0$, C runs Initial-Proxy-Key-Generate algorithms to generate initial proxy key σ_B^0, and returns it to A_I. If $t > 0$, C runs Proxy-Key-Update algorithms to generate a proxy key σ_B^t and returns it to A_I.

— **Proxy-Signature-Query:** On receiving a time period t, an identity ID_A of an original signer A, an identity ID_B of a proxy signer B, a message m and a warrant m_w, C returns a proxy signature ψ of the current time period to A_I.

At the end of every time period, A_I can continue to make chosen message attack or enter the *breakin* phase, while A_I must query according to the sequence of the time period strictly.

Breakin Phase: When A_I puts forward a *breakin* query, C simulates the key exposure situation, and will give all proxy signer's (including the target proxy signer) proxy key of time period T (T is key exposure period) to A_I. Here, A_I cannot achieve the partial private key and secret value of the proxy signer, but can replace the public key of the proxy signer.

Forgery Phase: A_I outputs $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^*\}$ of the time period μ^* $(0 \le \mu^* < T)$ as its forgery. We say A_I wins the game, if $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^*\}$

satisfies the following conditions. The advantage of A_I winning the game is defined as $Succ_{A_I}^{cma,cida,breakin}$.

(1) Verify $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^*\} = true$.

(2) A_I has never made Partial-Private-Key-Query on ID_A^*.

(3) A_I has never made Partial-Proxy-Key-Query on $\{m_w^*, ID_A^*, P_A^*\}$.

(4) In the time period μ^*, A_I has not made Proxy-Key-Query on $\{m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*)$.

(5) In the time period μ^*, A_I has not made Proxy-Signature-Query on $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*\}$.

Definition 2 The forward secure certificateless proxy signature is existentially unforgeable against chosen message attack of adversary A_I, if the probability of A_I winning in the Game 1 is negligible in polynomial time. In other words, $Succ_{A_I}^{cma,cida,breakin} \le \varepsilon$, where ε is negligible.

Game 2:

Setup: C runs the algorithm to generate master key s and system parameters $params$, C then sends $params$ and master key to A_{II}.

Chosen Message Attack Phase: In this phase, A_{II} can ask Public-Key-Query, Public-Key-Replacement-Query, Secret-Value-Query, Partial-Proxy-Key-Query, Proxy-Key-Query and Proxy-Signature-Query, and C will respond respectively. The operation is same as Game 1.

Breakin Phase: When A_{II} puts forward a *breakin* query, C simulates the key exposure situation, and will give all proxy signer's (including the target proxy signer) proxy key of time period T (T is key exposure period) to A_{II}.

Forgery Phase: A_{II} outputs $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^*\}$ of the time period μ^* $(0 \le \mu^* < T)$ as its forgery. We say A_{II} wins the game, if $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^*\}$ satisfies the following conditions. The advantage of A_{II} winning the game is defined as $Succ_{A_{II}}^{cma,cida,breakin}$.

(1) Verify $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^*\} = true$.

(2) A_{II} has never made Secret-Value-Query on ID_B^*, and has never made Public-Key-Replacement-Query on ID_B^*.

(3) In the time period μ^*, A_{II} has not made Proxy-Key-Query on $\{m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*\}$.

(4) In the time period μ^*, A_{II} has not made Proxy-Signature-Query on $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*\}$.

Definition 3 The forward secure certificateless proxy signature is existentially unforgeable against chosen message attack of adversary A_{II}, if the probability of A_{II} winning in the Game 2 is negligible in polynomial time. In other words, $Succ_{A_{II}}^{cma,cida,breakin} \le \varepsilon$, where ε is negligible.

5. The Construction of Forward Secure Certificateless Proxy Signature

In this section, we present the construction of forward secure certificateless proxy signature scheme. We let k be a security parameter and N be system total time periods.

Setup: This algorithm runs as follows.

(1) Let G_1, G_2 be groups of the same order q where G_1 is an additive group and G_2 is a multiplicative group, and $e: G_1 \times G_1 \to G_2$ is a bilinear paring.

(2) Choose a random generator $P \in G_1$, and select $s \in \mathbb{Z}_q^*$ randomly. Let s be the master key and set $P_0 = sP$.

(3) Choose cryptographic hash functions $H_1: \{0,1\}^* \to G_1^*$, $H_2: \{0,1\}^* \times G_1^* \times \{0,1\}^* \times G_1^* \to \mathbb{Z}_q^*$, $H_3: \{0,1\}^* \times G_1^* \times \{0,1\}^* \times G_1^* \times \{0,1\}^* \times G_1^* \to \mathbb{Z}_q^*$, $H_4: \{0,1\}^* \times \{0,1\}^* \times G_1^* \times \{0,1\}^* \times G_1^* \to G_1^*$.

The system parameters are $params = (G_1, G_2, e, q, P, P_0, N, H_1, H_2, H_3, H_4)$.

User-Key-Generate: Given system parameters *params* and a user's identity ID_i, the algorithm selects $x_i \in \mathbb{Z}_q^*$ randomly as the user's secret value, and computes the user's public key $P_i = (X_i, Y_i) = (x_i P, x_i P_0)$.

Partial-Private-Key-Extract: Given a user's identity $ID_i \in \{0,1\}^*$, the algorithm computes $Q_i = H_1(ID_i, P_i)$ and outputs the user's partial private key $D_i = sQ_i$.

Partial-Proxy-Key-Generate: Given system parameters *params*, a warrant m_w, an original signer *A*'s identity ID_A, partial private key D_A, secret value x_A and public key P_A, the algorithm computes the proxy signer *B*'s partial proxy key as follows.

(1) Randomly pick $r_A \in \mathbb{Z}_q^*$ and compute $R_A = r_A P$.

(2) Compute $h_A = H_2(m_w, ID_A, P_A, R_A)$ and $I_A = h_A D_A + (x_A + r_A) Q_A$.

(3) Output (m_w, σ_A) to *B*, and take $\sigma_A = (R_A, I_A)$ as the partial proxy key.

Partial-Proxy-Key-Verify: Given (m_w, σ_A), the algorithm computes $h_A = H_2(m_w, ID_A, P_A, R_A)$. If $e(I_A, P) = e(h_A P_0 + X_A + R_A, Q_A)$, accepts σ_A. Otherwise, rejects σ_A.

Initial-Proxy-Key-Generate: If (m_w, σ_A) is accepted, the algorithm generates the initial proxy key according to the following steps:

(1) Randomly pick $r_B^0 \in \mathbb{Z}_q^*$ and compute $R_B^0 = r_B^0 P$.

(2) Compute $h_B^0 = H_3(m_w, ID_A, P_A, ID_B, P_B, R_B^0)$ and $z_B^0 = h_B^0$.

(3) Compute $K_B^0 = I_A + h_B^0 x_B D_B + r_B^0 Q_B$.

(4) Output $\sigma_B^0 = (R_B^0, z_B^0, K_B^0)$ as the initial proxy key of time period 0.

Proxy-Key-Update: Given the current time period $t \in [1, N-1)$, the algorithm computes the σ_B^t of time period t from σ_B^{t-1} of time period $t-1$.

(1) Randomly pick $r_B^t \in \mathbb{Z}_q^*$ and compute $R_B^t = R_B^{t-1} + r_B^t P = \sum_{t=0}^{t} r_B^t P = u_B^t P$, while $u_B^t = \sum_{t=0}^{t} r_B^t$.

(2) Compute $h_B^t = H_3(m_w, ID_A, P_A, ID_B, P_B, R_B^t)$ and $z_B^t = \sum_{t=0}^{t} h_B^t = \sum_{t=0}^{t-1} h_B^t + h_B^t = z_B^{t-1} + h_B^t$.

(3) Compute $K_B^t = K_B^{t-1} + h_B^t x_B D_B + r_B^t Q_B = I_A + \sum_{t=0}^{t} h_B^t x_B D_B + \sum_{t=0}^{t} r_B^t Q_B = I_A + z_B^t x_B D_B + u_B^t Q_B$.

(4) Output $\sigma_B^t = (R_B^t, z_B^t, K_B^t)$ as the proxy key of the time period t, and delete r_B^t and σ_B^{t-1}.

Proxy-Sign: Given the current time period t and a message m, the algorithm performs the following steps to generate the proxy signature.

(1) Randomly pick $r_t \in \not\subset_q^*$ and compute $R_t = r_t P$.

(2) Compute $h = H_4(m, m_w, ID_B, P_B, R_t)$.

(3) Compute $V = K_B^t + hr_t$.

(4) Output $\psi = (R_A, R_B^t, z_B^t, R_t, V)$ as the proxy signature of the time period t.

Proxy-Verify: Given the message/signature (m, m_w, ψ), the algorithm performs the following steps to verify the validity of proxy signature:

(1) Check whether the message m is consistent with m_w. If not, return false and abort. Otherwise, continue (2).

(2) Compute $h_A = H_2(m_w, ID_A, P_A, R_A)$, $h = H_4(m, m_w, ID_B, P_B, R_t)$. If $e(V, P) = e(h_A P_0 + X_A + R_A, Q_A)e(z_B^t Y_B + R_B^t, Q_B)e(R_t, h)$, return true. Otherwise, return false.

6. Security Analysis

6.1 Correctness

We can easily verify the proposed scheme is correct.

$$
\begin{aligned}
e(V,P) &= e(K_B^t + hr_t, P) = e(I_A + \sum_{t=0}^{t} h_B^t x_B D_B + \sum_{t=0}^{t} r_B^t Q_B + hr_t, P) \\
&= e(h_A D_A + (x_A + r_A)Q_A, P)e(\sum_{t=0}^{t} h_B^t x_B D_B + \sum_{t=0}^{t} r_B^t Q_B, P)e(R_t, h) \\
&= e(h_A D_A, P)e((x_A + r_A)Q_A, P)e(z_B^t x_B D_B + u_B^t Q_B, P)e(R_t, h) \\
&= e(h_A P_0, Q_A)e(X_A + R_A, Q_A)e(z_B^t Y_B + R_B^t, Q_B)e(R_t, h) \\
&= e(h_A P_0 + X_A + R_A, Q_A)e(z_B^t Y_B + R_B^t, Q_B)e(R_t, h)
\end{aligned}
$$

6.2 Strong Unforgeability

According to the definition and security model of forward secure certificateless proxy signature provided in section 3, we prove our scheme is unforgeable as follows.

Theorem 1 Suppose there exists a polynomial bounded adversary A_I against our scheme with the success probability $Succ_{A_I}^{cma,cida,breakin}$ after asking at most q_{PK} Public-Key-Query, q_{PPK} Partial-Private-Key-Query, q_{ProK} Proxy-Key-Query. Then there exists an algorithm C that can solve the CDH problem with the success probability $Succ_C^{CDH} = \frac{1}{Nq_{PK}}(1 - \frac{1}{q_{PK}})^{q_{PPK}+q_{ProK}}$. $Succ_{A_I}^{cma,cida,breakin}$ in polynomial time.

Proof: We construct an algorithm C to solve the CDH problem. Let $(P, P_1 = aP, P_2 = bP)$ be a random instance of the CDH problem in G_1, and C will play as a challenger to interact with A_I. We show how C computes abP with the ability of A_I.

Setup: C sets N as the total numbers of time periods, and $T(0 \le T \le N-1)$ as the key exposure time period. In time period T, adversary will make *breakin* query. C sets $P_0 = aP = P_1$ and the system parameters $params = (G_1, G_2, e, q, P, P_0, N, H_1, H_2, H_3, H_4)$, and returns *params* to A_I.

C initializes the time period $t=0$, and adversary A_I will output a value of d after every time period query. If $d=0$, A_I will continue to make chosen message attack. If $d=breakin$, then A_I will enter *breakin* phase. At the end of chosen message attack of time period 0, A_I outputs $d=0$. If $d \neq breakin$ and $T \neq N$, then A_I will enter the next time period to continue the chosen message attack.

Chosen Message Attack Phase: In this phase, C regards hash functions as the random oracles. A_I can ask Public-Key-Query, Partial-Private-Key-Query, Secret-Value-Query, Public-Key-Replacement-Query, Partial-Proxy-Key-Query, Proxy-Key-Query and Proxy-Signature-Query. Without loss of generality, we assume that A_I doesn't repeat any two identical queries. C keeps seven lists L_1, L_2, L_3, L_4, H_2, H_3, H_4 to store the user's answers, where each list includes items of the form $(ID_i, x_i, P_i, \alpha_i, Q_i, D_i)$, $(m_{w_i}, ID_{A_i}, P_{A_i}, r_{A_i}, \beta_{A_i}, R_{A_i}, I_{A_i})$, $(m_{A_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, t, r_{B_i}^t, \gamma_{B_i}^t, R_{B_i}^t, z_{B_i}^t, K_{B_i}^t)$, $(m_i, m_{w_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, t, r_t, \kappa_t, R_{A_i}, R_{B_i}^t, z_{B_i}^t, R_t, V)$, $(m_{w_i}, ID_{A_i}, P_{A_i}, R_{A_i}, \beta_{A_i})$, $(m_{w_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, R_{B_i}^t, \gamma_{B_i}^t)$, $(m_i, m_{w_i}, ID_{B_i}, P_{B_i}, R_t, \kappa_t)$.

— **Public-Key-Query:** C randomly picks $f \in \{1,2,\cdots,q_{PK}\}$. On receiving each of public key query, C checks whether the ID_i has been created in L_1 list. If so, C returns P_i to A_I directly. If not, C will randomly select $\alpha_i, x_i \in \mathbb{Z}_q^*$, if $i \neq f$, C sets $P_i = (x_i P, x_i P_0)$, $Q_i = \alpha_i P$, $D_i = \alpha_i P_0$. Otherwise, C sets $P_f = (x_f P, x_f P_0)$, $Q_f = \alpha_f P + P_2$, $D_f = \perp$. Finally, C adds $(ID_i, x_i, P_i, \alpha_i, Q_i, D_i)$ into the L_1 list, and returns P_i to A_I.

— **Partial-Private-Key-Query:** Suppose the query is made on ID_i, if $i = f$, C aborts. Otherwise, C checks whether the ID_i has been created in L_1 list. If not, C randomly selects $\alpha_i, x_i \in \mathbb{Z}_q^*$, and sets $P_i = (x_i P, x_i P_0)$", $Q_i = \alpha_i P$, $D_i = \alpha_i P_0$. C adds $(ID_i, x_i, P_i, \alpha_i, Q_i, D_i)$ into the L_1 list, and returns D_i to A_I. Otherwise, C returns D_i to A_I directly.

— **Public-Key-Replacement-Query:** Suppose the query is made on (ID_i, P_i'), C checks whether the ID_i has been created in L_1 list. If not, C adds $(ID_i, \perp, P_i', \perp, \perp, \perp)$ into the L_1 list. Otherwise, C updates the item of ID_i as $(ID_i, \perp, P_i', \alpha_i, Q_i, D_i)$.

— **Secret-Value-Query:** Suppose the query is made on ID_i, C checks whether the ID_i has been created in L_1 list. If not, C randomly selects $\alpha_i, x_i \in \mathbb{Z}_q^*$, and sets $P_i = (x_i P, x_i P_0)$, $Q_i = \alpha_i P$, $D_i = \alpha_i P_0$. C adds $(ID_i, x_i, P_i, \alpha_i, Q_i, D_i)$ into the L_1 list, and returns x_i to A_I. Otherwise, if $x_i = \perp$, C returns $\perp$ to A_I. Otherwise, C returns x_i to A_I directly.

— **H_2 Query:** Suppose the query is made on $(m_{w_i}, ID_{A_i}, P_{A_i}, R_{A_i})$, C randomly selects $\beta_{A_i} \in \mathbb{Z}_q^*$ that hasn't appeared in the H_2 list. Then C adds $(m_{w_i}, ID_{A_i}, P_{A_i}, R_{A_i}, \beta_{A_i})$ into the H_2 list and returns β_{A_i} to A_I.

— **H_3 Query:** Suppose the query is made on $(m_{w_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, R_{B_i}^t)$, C randomly selects $\gamma_{B_i}^t \in \mathbb{Z}_q^*$ that hasn't appeared in the H_3 list. Then C adds $(m_{w_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, R_{B_i}^t, \gamma_{B_i}^t)$ into the H_3 list and returns $\gamma_{B_i}^t$ to A_I.

— **H_4 Query:** Suppose the query is made on $(m_i, m_{w_i}, ID_{B_i}, P_{B_i}, R_t)$, C randomly selects

$\kappa_t \in \square_q^*$ that hasn't appeared in the H_4 list. Then C adds $(m_i, m_{w_i}, ID_{B_i}, P_{B_i}, R_t, \kappa_t)$ into the H_4 list and returns κ_t to A_I.

— **Partial-Proxy-Key-Query:** Suppose the query is made on (ID_{A_i}, m_{w_i}), C firstly browses the L_1 list to get the current public key of the ID_{A_i}. Then C generates the partial proxy key according to the following steps.

(1) Randomly pick $r_{A_i}, \beta_{A_i} \in \square_q^*$ that β_{A_i} hasn't appeared in the H_2 list.

(2) Set $H_2(m_{w_i}, ID_{A_i}, P_{A_i}, R_{A_i}) = \beta_{A_i}$ and compute $R_{A_i} = r_{A_i}P - (\beta_{A_i}P_0 + X_A)$.

(3) Compute $I_{A_i} = r_{A_i}Q_{A_i}$.

C adds $(m_{w_i}, ID_{A_i}, P_{A_i}, r_{A_i}, \beta_{A_i}, R_{A_i}, I_{A_i})$ into the L_2 list and returns $\sigma_{A_i} = (R_{A_i}, I_{A_i})$ to A_I.

— **Proxy-Key-Update:** This process is completed by C alone, which makes preparations for A_I's proxy key query and *breakin* query, and A_I can't make any query in the process. Given the current time period, C simulates the proxy key update process from the initial time period 0. The specific steps are as follows:

(1) Firstly check whether the item (ID_{A_i}, m_{w_i}) is in the L_2 list. If not, C performs Partial-Proxy-Key-Query to obtain the tuple $(r_{A_i}, \beta_{A_i}, R_{A_i}, I_{A_i})$, and adds them into the L_2 list. Otherwise, C returns the tuple $(r_{A_i}, \beta_{A_i}, R_{A_i}, I_{A_i})$ directly.

(2) Randomly pick $r_{B_i}^t, \gamma_{B_i}^t \in \square_q^*$ that $\gamma_{B_i}^t$ hasn't appeared in the H_3 list.

(3) Set $H_3(m_{w_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, R_{B_i}^t) = \gamma_{B_i}^t$ and compute $z_{B_i}^t = z_{B_i}^{t-1} + \gamma_{B_i}^t$.

(4) Compute $R_{B_i}^t = r_{B_i}^t P - z_{B_i}^t Y_B$ and $K_{B_i}^t = r_{A_i}Q_{A_i} + r_{B_i}^t Q_{B_i}$.

C adds $(m_{w_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, t, r_{B_i}^t, \gamma_{B_i}^t, R_{B_i}^t, z_{B_i}^t, K_{B_i}^t)$ into the L_3 list.

— **Proxy-Key-Query:** Suppose the query is made on $(m_{w_i}, ID_{A_i}, ID_{B_i})$ in the time period t $(0 \le t < N)$, C firstly checks the L_1 list. If $x_{B_i} = \perp$, C returns $\perp$ to A_I. Otherwise, C checks L_3 list according to $(ID_{A_i}, ID_{B_i}, m_{w_i}, t)$. If $ID_{B_i} = ID_f$, C aborts. Otherwise, C returns $\sigma_{B_i}^t = (R_{B_i}^t, z_{B_i}^t, K_{B_i}^t)$ to A_I.

— **Proxy-Sign-Query:** Suppose the query is made on $(m_i, m_{w_i}, ID_{A_i}, ID_{B_i})$ in the time period t $(0 \le t < N)$, C generates the proxy signature according to the following steps:

(1) Firstly check whether the item $(ID_{A_i}, ID_{B_i}, m_{w_i}, t)$ is in the L_3 list. If not, C performs Proxy-Key-Query to obtain the tuple $(r_{A_i}, R_{A_i}, \beta_{A_i}, r_{B_i}^t, \gamma_{B_i}^t, R_{B_i}^t, z_{B_i}^t, K_{B_i}^t)$, and adds them into the L_3 list. Otherwise, C returns the tuple $(r_{A_i}, R_{A_i}, \beta_{A_i}, r_{B_i}^t, \gamma_{B_i}^t, R_{B_i}^t, z_{B_i}^t, K_{B_i}^t)$ directly.

(2) Randomly pick $r_t, \kappa_t \in \square_q^*$ that κ_t hasn't appeared in the H_4 list.

(3) Set $H_4(m_i, m_{w_i}, ID_{B_i}, P_{B_i}, R_{B_i}^t) = \kappa_t$ and compute $R_t = r_t P$.

(4) Compute $V = r_{A_i}Q_{A_i} + r_{B_i}^t Q_{B_i} + r_t\kappa_t$.

C adds $(m_i, m_{w_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, t, r_t, \kappa_t, R_{A_i}, R_{B_i}^t, z_{B_i}^t, R_t, V)$ into the L_4 list and returns $\psi = (R_{A_i}, R_{B_i}^t, z_{B_i}^t, R_t, V)$ to A_I.

Breakin Phase: A_I outputs a decision value d, and C decides whether to enter the *breakin* phase. If $0 \le t < T$ and $d = 0$, A_I enters the next time period to continue the chosen message attack. If $t = T$ and $d = breakin$, C returns all proxy signer's (including the target proxy signer) proxy key of the current time period to A_I. Otherwise, C aborts.

Once A_I enters into the *breakin* phase, he can't continue the chosen message attack. After the *breakin* query, A_I enters into the forgery phase.

Forgery Phase: If C doesn't abort in the simulation, then A_I outputs a validly forged proxy signature $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^*\}$ of the time period $\mu^* (0 \le \mu^* < T)$.

Analysis: If $ID_A^* \ne ID_f$, aborts. If $ID_A^* = ID_f$ and $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^* = (R_A^*, R_B^{\mu^*}, z_B^{\mu^*}, R_\mu^*, V^*)\}$ satisfies the requirements as defined in game 1, according to forking lemma[26], C selects different hash function $H_2^{'}$ and uses A_I's capability to get another valid tuple $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^{*'} = (R_A^*, R_B^{\mu^*}, z_B^{\mu^*}, R_\mu^*, V^{*'})\}$, in which $H_2(m_w^*, ID_A^*, P_A^*, R_A^*) = \beta_A^* \ne \beta_A^{*'} = H_2^{'}(m_w^*, ID_A^*, P_A^*, R_A^*)$. Then C gets two tuples $e(V^*, P) = e(\beta_A^* P_0 + X_A^* + R_A^*, Q_A^*)\, e(z_B^{\mu^*} Y_B^* + R_B^{\mu^*}, Q_B^*) e(R_\mu^*, h)$ and $e(V^{*'}, P) = e(\beta_A^{*'} P_0 + X_A^* + R_A^*, Q_A^*) e(z_B^{\mu^*} Y_B^* + R_B^{\mu^*}, Q_B^*) e(R_\mu^*, h)$, thus $e(V^* - V^{*'}, P) = e((\beta_A^* - \beta_A^{*'}) P_0, Q_A^*)$, in which $Q_A^* = \alpha_f P + P_2 = \alpha_f P + bP$. Then $e(V^* - V^{*'}, P) = e((\beta_A^* - \beta_A^{*'}) P_0, \alpha_f P + P_2)$, and $V^* - V^{*'} = (\beta_A^* - \beta_A^{*'})(a_f + b) P_0$, in which $P_0 = P_1 = ap$. C checks L_1 list get α_f, and computes $abP = (V^* - V^{*'})(\beta_A^* - \beta_A^{*'})^{-1} - \alpha_f P_1$.

Probability of Success: Event E_1 denotes that the algorithm C does not exit throughout the simulation. Event E_2 denotes that A_I outputs $t = T$ and $d = breakin$. Event E_3 denotes that A_I forges a valid proxy signature in the time period $\mu^* (0 \le \mu^* < T)$. Event E_4 denotes that A_I outputs a valid tuple $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^*\}$, and $ID_A^* = ID_f$ when event E_3 occurs. Then $Succ_C^{CDH} = \Pr[E_1 \wedge E_2 \wedge E_3 \wedge E_4] = \Pr[E_1] \Pr[E_2 \mid E_1] \Pr[E_3 \mid E_1 \wedge E_2] \Pr[E_4 \mid E_1 \wedge E_2 \wedge E_3]$. Among that $\Pr[E_1] = (1 - \frac{1}{q_{PK}})^{q_{PPK} + q_{ProK}}$, $\Pr[E_2 \mid E_1] = \frac{1}{N}$, $\Pr[E_3 \mid E_1 \wedge E_2] = Succ_{A_I}^{cma,cida,breakin}$, $\Pr[E_4 \mid E_1 \wedge E_2 \wedge E_3] = \frac{1}{q_{PK}}$. Hence $Succ_C^{CDH} = \frac{1}{N q_{PK}} \cdot (1 - \frac{1}{q_{PK}})^{q_{PPK} + q_{ProK}} Succ_{A_I}^{cma,cida,breakin}$.

Theorem 2 Suppose there exists a polynomial bounded adversary A_{II} against our scheme with the success probability $Succ_{A_{II}}^{cma,cida,breakin}$ after asking at most q_{PK} Public-Key-Query, q_{PKR} Public-Key-Replacement-Query, q_{SV} Secret-Value-Query, q_{ProK} Proxy-Key-Query. Then there exists an algorithm C can solve the CDH problem with the success probability $Succ_C^{CDH} = \frac{1}{N q_{PK}} (1 - \frac{1}{q_{PK}})^{q_{PKR} + q_{SV} + q_{ProK}} Succ_{A_{II}}^{cma,cida,breakin}$ in polynomial time.

Proof: We construct an algorithm C to solve the CDH problem. Let $(P, P_1 = aP, P_2 = bP)$ be a random instance of the CDH problem in G_1, and C will play as a challenger to interact with A_{II}. We show how C computes abP with the ability of A_{II}.

Setup: C sets N as the total numbers of time periods, and $T (0 \le T \le N - 1)$ as the key exposure time period. In time period T, adversary will make *breakin* query. C randomly

selects $s \in \square_q^*$, sets $P_0 = sP$ and the system parameters $params = (G_1, G_2, e, q, P, P_0, N, H_1, H_2, H_3, H_4)$. Then C returns master key s and $params$ to A_{II}.

C initializes the time periods $t = 0$, and adversary A_{II} will output a value of d after every time period query. If $d = 0$, A_{II} will continue to make chosen message attack. If $d = breakin$, then A_{II} will enter *breakin* phase. At the end of chosen message attack of time period 0, A_{II} outputs $d = 0$. If $d \neq breakin$ and $T \neq N$, then A_{II} will enter the next time period to continue the chosen message attack.

Chosen Message Attack Phase: In this phase, C regards hash functions as the random oracles. A_{II} can ask Public-Key-Query, Secret-Value-Query, Public-Key-Replacement-Query, Partial-Proxy-Key-Query, Proxy-Key-Query and Proxy-Signature-Query. Without loss of generality, we assume that A_{II} doesn't repeat any two identical queries. C keeps seven lists L_1, L_2, L_3, L_4, H_2, H_3, H_4 to store the user's answers, where each list includes items of the form $(ID_i, x_i, P_i, \alpha_i, Q_i, D_i)$, $(m_{w_i}, ID_{A_i}, P_{A_i}, r_{A_i}, \beta_{A_i}, R_{A_i}, I_{A_i})$, $(m_{A_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, t, r_{B_i}^t, \gamma_{B_i}^t, R_{B_i}^t, z_{B_i}^t, K_{B_i}^t)$, $(m_i, m_{w_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, t, r_t, \kappa_t, R_{A_i}, R_{B_i}^t, z_{B_i}^t, R_t, V)$, $(m_{w_i}, ID_{A_i}, P_{A_i}, R_{A_i}, \beta_{A_i})$, $(m_{w_i}, ID_{A_i}, P_{A_i}, ID_{B_i}, P_{B_i}, R_{B_i}^t, \gamma_{B_i}^t)$, $(m_i, m_{w_i}, ID_{B_i}, P_{B_i}, R_t, \kappa_t)$.

— **Public-Key-Query:** C randomly picks $f \in \{1, 2, \cdots, q_{PK}\}$. On receiving each of public key query, C checks whether the ID_i has been created in L_1 list. If so, C will returns P_i to A_I directly. If not and $i \neq f$, C will randomly select $\alpha_i, x_i \in \square_q^*$, and set $P_i = (x_i P, x_i P_0)$, $Q_i = \alpha_i P$, $D_i = \alpha_i P_0$. If $i = f$, C randomly selects $\alpha_f \in \square_q^*$, and sets $P_f = (P_1, sP_1)$, $Q_f = \alpha_f P + P_2$, $x_f = \perp$, $D_f = \perp$. Finally, C adds $(ID_i, x_i, P_i, \alpha_i, Q_i, D_i)$ into the L_1 list, and returns P_i to A_{II}.

— **Public-Key-Replacement-Query:** Suppose the query is made on (ID_i, P_i'), If $i = f$, C aborts. Otherwise, C checks whether the ID_i has been created in L_1 list. If not, C adds $(ID_i, \perp, P_i', \perp, \perp, \perp)$ into the L_1 list. Otherwise, C updates the item of ID_i as $(ID_i, \perp, P_i', \alpha_i, Q_i, D_i)$.

— **Secret-Value-Query:** On receiving such a query ID_i, If $i = f$, C aborts. If $i \neq f$, C checks whether the ID_i has been created in L_1 list. If not, C will randomly select $\alpha_i, x_i \in \square_q^*$, and set $P_i = (x_i P, x_i P_0)$, $Q_i = \alpha_i P$, $D_i = \alpha_i P_0$. C adds $(ID_i, x_i, P_i, \alpha_i, Q_i, D_i)$ into the L_1 list, and returns x_i to A_I. Otherwise, if $x_i \neq \perp$, C returns x_i to A_{II}. Otherwise, C outputs $\perp$.

H_2 Query, H_3 Query, H_4 Query, Partial-Proxy-Key-Query, Proxy-Key-Query and Proxy-Sign-Query are the same as theorem 1.

Breakin Phase: A_{II} outputs a decision value d, and C decides whether to enter the *breakin* phase. If $0 \leq t < T$ and $d = 0$, A_{II} enters the next time period to continue the chosen message attack. If $t = T$ and $d = breakin$, C returns all proxy signer's (including the target proxy signer) proxy key of the current time period to A_{II}. Otherwise, C aborts.

Once A_{II} enters into the *breakin* phase, he can't continue the chosen message attack. After the *breakin* query, A_{II} enters into the forgery phase.

Forgery Phase: If C doesn't abort in the simulation, then A_{II} outputs a validly forged proxy signature $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^*\}$ of the time period $\mu^*(0 \le \mu^* < T)$.

Analysis: If $ID_B^* \neq ID_f$, aborts. If $ID_B^* = ID_f$ and $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^* = (R_A^*, R_B^{\mu^*}, z_B^{\mu^*}, R_\mu^*, V^*)\}$ satisfies the requirements as defined in game 2, according to forking lemma[26], C selects different hash function $H_3^{'}$ and uses A_{II}'s capability to get another valid tuple $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^{*'} = (R_A^*, R_B^{\mu^*}, z_B^{\mu^{*'}}, R_\mu^*, V^{*'})\}$, in which $H_3(m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, R_B^{\mu^*}) = \gamma_B^{\mu^*} \neq \gamma_B^{\mu^{*'}} = H_3^{'}(m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, R_B^{\mu^*})$ and $z_B^{\mu^*} \neq z_B^{\mu^{*'}}$. Then C gets two valid tuples $e(V^*, P) = e(h_A^* P_0 + X_A^* + R_A^*, Q_A^*) e(z_B^{\mu^*} Y_B^* + R_B^{\mu^*}, Q_B^*) e(R_\mu^*, h)$ and $e(V^{*'}, P) = e(h_A^* P_0 + X_A^* + R_A^*, Q_A^*) e(z_B^{\mu^{*'}} Y_B^* + R_B^{\mu^*}, Q_B^*) e(R_\mu^*, h)$, thus $e(V^* - V^{*'}, P) = e((z_B^{\mu^*} - z_B^{\mu^{*'}}) Y_B^*, Q_B^*)$, in which $Q_B^* = \alpha_f P + P_2 = \alpha_f P + bP$ and $Y_B^* = sP_1$. Then $e(V^* - V^{*'}, P) = e((z_B^{\mu^*} - z_B^{\mu^{*'}}) sP_1, \alpha_f P + bP)$, and $V^* - V^{*'} = (z_B^{\mu^*} - z_B^{\mu^{*'}})(a_f + b) sP_1$, in which $P_1 = aP$. C checks L_1 list to get α_f, and computes $abP = s^{-1}(V^* - V^{*'})(z_B^{\mu^*} - z_B^{\mu^{*'}})^{-1} - \alpha_f P_1$.

Probability of Success: Event E_1 denotes that the algorithm C does not exit throughout the simulation. Event E_2 denotes that A_{II} outputs $t = T$ and $d = breakin$. Event E_3 denotes that A_{II} forges a valid proxy signature in the time period $\mu^*(0 \le \mu^* < T)$. Event E_4 denotes that A_{II} outputs a valid tuple $\{m^*, m_w^*, ID_A^*, P_A^*, ID_B^*, P_B^*, \mu^*, \psi^*\}$, and $ID_B^* = ID_f$ when event E_3 occurs. Then $Succ_C^{CDH} = \Pr[E_1 \Lambda E_2 \Lambda E_3 \Lambda E_4] = \Pr[E_1] \Pr[E_2 \mid E_1] \Pr[E_3 \mid E_1 \Lambda E_2] \Pr[E_4 \mid E_1 \Lambda E_2 \Lambda E_3]$. Among that $\Pr[E_1] = (1 - \frac{1}{q_{PK}})^{q_{PKR} + q_{SV} + q_{ProK}}$, $\Pr[E_2 \mid E_1] = \frac{1}{N}$, $\Pr[E_3 \mid E_1 \Lambda E_2] = Succ_{A_{II}}^{cma, cida, breakin}$, $\Pr[E_4 \mid E_1 \Lambda E_2 \Lambda E_3] = \frac{1}{q_{PK}}$. Hence $Succ_C^{CDH} = \frac{1}{N q_{PK}} \cdot (1 - \frac{1}{q_{PK}})^{q_{PKR} + q_{SV} + q_{ProK}} Succ_{A_{II}}^{cma, cida, breakin}$.

6.3 Efficiency Analysis

Table 1 gives the comparison of computational efforts and security proof of our scheme with those of the schemes in [23] and [24]. We denote by BP the bilinear pairing operation, H the hash operation, M the scalar multiplication in G_1, E the exponentiation in the group G_2.

Compared to scheme in [23], our **Proxy-Verify** algorithm requires one more multiplication operations, but our scheme gives the concrete security proof, while the scheme [23] is insecure under the public key replacement attack. Compare to scheme in [24], although our **Proxy-Verify** algorithm requires more multiplication operations, our scheme is still more efficient, because exponentiation operations requires much more computational efforts. The most important is the scheme in [23] and [24] don't give concrete security proof, while our scheme is proved to be secure in the random oracle model.

Table 1. Efficiency Analysis

Scheme	Proxy-Sign	Proxy-Verify	Security Proof
Scheme in [23]	$1H + 1M$	$3H + 1M + 4BP$	No
Scheme in [24]	$1H + 1M$	$2H + 1E + 4BP$	No
Our Scheme	$1H + 1M$	$2H + 2M + 4BP$	Yes

7. Conclusion

In this paper, we propose a forward secure certificateless proxy signature in the random oracle model. Our security model takes into account the super adversary in certificateless signature. We prove our scheme is existentially unforgeable against chosen message attack under CDH assumption. What is worth mentioning is that we firstly give the formal definition and security model of forward secure certificateless proxy signature. Moreover, the scheme has effectively dealt with the key exposure problem and has no certificate management problem.

Acknowledgments.

We would like to thank anonymous referees for their helpful comments and suggestions.

References

[1] M. Mambo, K. Usuda and E. Okamoto, "Proxy signature: delegation of the power to sign messages," *IEICE Transactions on Fundamentals*, vol. E79-A, no. 9, pp. 1338-1353, 1996. Article (CrossRef Link)

[2] X. Y. Huang, Y. Mu, W. Sulilo and F. T. Zhang, "Short designed verifier proxy signature from pairings," in *Proc. of EUC Workshops 2005*, LNCS 3823, pp. 835-844, 2005. Article (CrossRef Link)

[3] F. G. Zhang and K. Kim, "Efficient ID-based blind signature and proxy signature from bilinear pairings," in *Proc. of ACISP 2003*, LNCS 2727, pp. 312-323, 2003. Article (CrossRef Link)

[4] H. X. Wang and J. Pieprzyk, "Efficient one-time proxy signature," in *Proc. of ASIACRYPT 2003*, LNCS 2894, pp. 507-522, 2003. Article (CrossRef Link)

[5] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. of ASIACRYPT 2003*, LNCS 2894, pp. 452-473, 2003. Article (CrossRef Link)

[6] W. Yap, S. Heng and B. Goi, "Cryptanalysis of some proxy signature schemes without certificates," in *Proc. of WISTP 2007*, LNCS 4462, pp. 115-126, 2007. Article (CrossRef Link)

[7] H. Chen, F. T. Zhang and R. S. Song, "Certificateless proxy signature with provable security," *Journal of Software*, vol. 20, no. 3, pp. 692-701, 2009. Article (CrossRef Link)

[8] H. Xiong, F. G. Li and Z. G. Qin, "A provably secure proxy signature scheme in certificateless cryptography," *International Journal of Informatica*, vol. 21, no. 2, pp. 277-294, 2010. Article (CrossRef Link)

[9] J. G. Li, X. Y. Huang, Y. Mu and W. Wu, "Cryptanalysis and improvement of an efficient certificateless signature scheme," *Journal of Communications and Networks*, vol. 10, no. 1, pp. 10-17, 2008. Article (CrossRef Link)

[10] R. Anderson, "Two remarks on public key cryptology," Invited lecture, in *Proc. of the 4th ACM Conf. on Computer and Communications Security*, 1997. Article (CrossRef Link)

[11] M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in *Proc. of CRYPTO'99*, LNCS 1666, pp. 431-448, 1999. Article (CrossRef Link)

[12] G. Itkis and L. Reyzin, "Forward-secure signature with optical signing and verifying," in *Proc. of CRYPTO 2001*, LNCS 2139, pp. 332-354, 2001. Article (CrossRef Link)

[13] T. Malkin, D. Micciancio and S. Miner, "Efficient generic forward-secure signature with an unbounded number of time periods," in *Proc. of EUROCRYPT 2002*, LNCS 2332, pp. 400-417, 2002. Article (CrossRef Link)

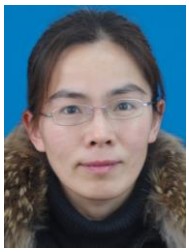
[14] B. G. Kang, J. H. Park and S. G. Hahn, "A new forward secure signature scheme," *Cryptology ePrint Archive*, Report 2004/183. Article (CrossRef Link)

[15] B. Alomair, K. Sampigethaya and R. Poovendran, "Efficient generic forward-secure signatures and proxy signatures," in *Proc. of EuroPKI 2008*, LNCS 5057, pp. 166-181, 2008. Article

(CrossRef Link)
[16] J. Yu, F. Y. Kong, X. G. Cheng, R. Hao and G. W. Li, "Construction of yet another forward secure signature scheme using bilinear maps," in *Proc. of ProvSec 2008*, LNCS 5324, pp. 83-97, 2008. Article (CrossRef Link)
[17] T. Nakanishi, Y. Hira and N. Funabiki, "Forward-secure group signatures from pairings," in *Proc. of Pairing 2009*, LNCS 5671, pp. 171-186, 2009. Article (CrossRef Link)
[18] J. Yu, R. Hao, F. Y. Kong, X. G. Cheng and X. F. Guo, "Forward-secure multi-signature in the standard model: security model and construction," *Journal of Software*, vol. 21, no. 11, pp. 2920-2932, 2010. Article (CrossRef Link)
[19] B. Libert and M. Yung, "Fully forward-secure group signature," in *Proc. of Cryptography and Security: From Theory to Applications*, LNCS 6805, pp. 156-184, 2012. Article (CrossRef Link)
[20] J. Buchmann, E. Dahmen and A. Hülsing, "XMSS-a practical forward secure signature scheme based on minimal security assumptions," in *Proc. of PQCrypto'2011*, LNCS 7071, pp. 117-129, 2011. Article (CrossRef Link)
[21] A. Hülsing, C. Busold and J. Buchmann, "Forward secure signatures on smart cards," in *Proc. of SAC 2012*, LNCS 7707, pp. 66-80, 2013. Article (CrossRef Link)
[22] M. Abdalla, F. B. Hamouda and D. Pointcheval, "Tighter reductions for forward-secure signature schemes," in *Proc. of PKC 2013*, LNCS 7778, pp. 292-311, 2013. Article (CrossRef Link)
[23] H. B. Chen, X. Y. Yang and Z. Y. Liang, "Forward secure certificateless proxy signature scheme," *Application Research of Computers*, vol. 26, no. 8, pp. 3019-3021, 2009. Article (CrossRef Link)
[24] H. B. Chen, X. Y. Yang, Z. Y. Liang and X. G. Wu, "Forward secure certificateless proxy signature scheme," *Computer Engineering*, vol. 36, no. 2, pp. 156-157, 2010. Article (CrossRef Link)
[25] X. Y. Huang, Y. Mu, W. Sulilo, D. S. Wong and W. Wu, "Certificateless signature revisited," in *Proc. of ACISP 2007*, LNCS 4586, pp. 308-322, 2007. Article (CrossRef Link)
[26] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signature," *Journal of Cryptology*, vol. 13, no. 3, pp. 361-396, 2000. Article (CrossRef Link)

Jiguo Li received the B.S. degree in mathematics from Heilongjiang University, Harbin, China in 1996, M.S. degree in mathematics and PhD degree in computer science from Harbin Institute of Technology, Harbin, China in 2000 and 2003, respectively. He is a visiting scholar in the School of Computer Science and Software Engineering and the Centre for Computer and Information Security Research, University of Wollongong, Wollongong, Australia between 2006 and 2007. He is currently professor in the College of Computer and Information Engineering, Hohai University, Nanjing, China. He is currently holding the National Natural Science Foundation of China, the Fundamental Research Funds for the Central Universities, and the ”Six Talent Peaks Program” of Jiangsu Province of China. His research interests include cryptography, network security, wireless security and trusted computing etc. He has published more than 80 referred research papers and two books. He has served on PC member of several international conferences and served as the reviewers of some international journal and conference.

Yanqiong Li received B.S. degree in computer science and technology from Henan Polytechnic University, Jiaozuo, China in 2010. She received M.S. degree in computer science and technology from Hohai University, Nanjing, China in 2013. Her research interests include cryptography and information security, network security.



Yichen Zhang received the B.S. degree in computer science from the Qiqihar University, Qiqihar, China in 1995. She is currently lecturer and PhD student in the College of Computer and Information Engineering, Hohai University, Nanjing, China. Her research interests include cryptography, network security. She has published more than 20 referred research papers at international conferences and journals.