Cryptanalysis and Improvement of an Efficient Certificateless Signature Scheme

Jiguo Li, Xinyi Huang, Yi Mu, and Wei Wu

Abstract: **In traditional digital signature schemes, certificates signed by a trusted party are required to ensure the authenticity of the public key. In Asiacrypt 2003, the concept of certificateless signature scheme was introduced. 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. Recently, Yap *et al.* proposed an efficient certificateless signature scheme and claimed that their scheme is existentially unforgeable in the random oracle model. In this paper, we show that the certificateless signature scheme proposed by Yap *et al.* is insecure against public key replacement attacks. Furthermore, we propose an improved certificateless signature scheme, which is existentially unforgeable against adaptive chosen message attacks under the computational Diffie-Hellman assumption in the random oracle model and provide the security proof of the proposed scheme.**

Index Terms: **Certificateless cryptography, certificateless signature, public key replacement attack, security analysis.**

I. INTRODUCTION

In traditional digital signature schemes, certificates that are signed by a trusted third party are used to 'bind' the user's identity and his public key. In [1], Shamir introduced a new notion called identity-based cryptography where the user's public key is indeed his identity (such as an email address, an IP address, etc.). This way, the need of certification can be avoided. In identity-based cryptography, the user's secret key is generated by Key Generation Center (KGC). Therefore, there is an inherent key escrow problem in such a cryptosystem.

To fill the gap between traditional cryptography and identity-based cryptography, Al-Riyami and Paterson proposed a new paradigm called *certificateless cryptography* in [2]. In a certificateless cryptosystem, KGC is involved to issue a partial key for user. The user also independently generates an additional public/secret key pair and performs some cryptographic operations in such a way that they can only be carried out when both the user's partial key and the user's secret key are known. Knowing one of them should not be able to impersonate the user and carry out any of the cryptographic operations as the user. Therefore, certificateless cryptography not only solves the key escrow problem that is inherent in identity-based cryptography, but also eliminates the use of certificates as in traditional digital signature schemes, which are generally considered to be costly to use and manage.

Due to the lack of public key authentication, it is important to assume that an adversary can replace the user's public key by a false key of its choice [2]. In order to provide a secure certificateless signature scheme, this type of attacks must not be able to produce signatures that pass verification with the replaced public key [2]. An assumption that must be made is that the KGC does not mount a public key replacement attack since he possesses the master-key and can generate all user's partial private keys. It is reasonable that we suppose KGC is trusted not to replace public keys. This way, the level of trust is similar to the trust in a CA in a traditional PKI. We will review the adversarial model defined in [2], [3] in the next section.

Many certificateless public key encryption schemes [2], [4]–[7] and certificateless signature schemes (CLS) [2], [3], [8]–[11] have been proposed since certificateless public key cryptography was introduced. The first CLS scheme was proposed by Al-Riyami and Paterson in [2] but there is no security proof provided. Besides, their CLS scheme has been proven insecure in their defined model by Huang *et al.*[11]. They showed that an adversary can successfully forge a certificateless signature by replacing the public key of the signer and proposed a new scheme, which resists against the above attack. Recently, Huang *et al.* [17] revisited the security models of certificateless signatures and proposed two new constructions which are provably secure in the random oracle model. Yum and Lee [8] proposed a generic construction of CLS based on an identity-based signature scheme and a traditional public key signature scheme, which is a different approach in constructing CLS. The merit of the above approach is that the resulting CLS scheme can achieve the same trust level as that of a traditional signature scheme. Hu *et al.* [12] showed that their generic construction was insecure against public key replacement attack. In particular, they showed that the security requirements of their generic building blocks are insufficient to support some security claims stated in [8]. Hu *et al.* also proposed a modification of their scheme and showed its security in a new and simplified security model. Wang *et al.* proposed a certificateless threshold signature scheme [10] based on bilinear pairings. Their scheme is robust and existentially unforgeable against adaptive chosen message attacks under computational Diffie-Hellman assumption in the random oracle model. Zhang *et al.* [3] presented a security model for certificateless public-key signature schemes, and proposed an efficient construction based on bilinear pairings. Recently, Yap *et al.* proposed an efficient certificateless signature

Manuscript received November 24, 2006; approved for publication by Huaxiong Wang, Division I Editor, October 12, 2007.

J. Li is with the College of Computer and Information Engineering, Hohai University, Nanjing, P.R.China, email: lijiguo@hhu.edu.cn.

X. Huang, Y. Mu, and W. Wu are with School of Computer Science & Software Engineering, University of Wollongong, Australia, email: {xh068, ymu}@uow.edu.au, weiwu81@gmail.com.

The work is partially supported by the National Natural Science Foundation of China (No.60673070), the National High-Tech Research and Development Plan of China (No. 2007AA01Z409), Natural Science Foundation of Jiangsu Province (No.BK2006217), and the project of Jiangsu Province Police Ministry (No.200503002).

scheme [9]. In Yap's scheme, by using a shorter public key, there are only two pairing computations in the verification algorithm. Besides, no pairing computation is needed in the signing algorithm. Hence, their scheme is more efficient than the existing certificateless signature scheme. However, as we will show in this paper, the certificateless signature scheme proposed by Yap *et al.* is insecure against public key replacement attack.

Our Contribution

In this paper, we show that the scheme proposed by Yap *et al.* does not resist against public key replacement attacks, and gives the main cause suffered from attack. More precisely, we show that an attacker who does *not* possess the master-key but can only do a public key replacement attack, can always successfully forge a signature. In addition, the message of the forgery can be chosen arbitrarily by the adversary. The main difference compared with a common public key replacement attack used in the other certificateless signature scheme [11], [12] is that the adversary can obtain the user's full private key defined in their scheme [9] when he/she replaces the public key of this user. Furthermore, we propose a provably secure efficient certificateless signature scheme, which is existentially unforgeable in the random oracle model under adaptive chosen message attacks and proves the security of the proposed scheme in Zhang's security model [3]. Compared with Yap's security proof, our proof can be obtained without the requirement that $\mathcal{A}_I$ should have submitted the secret information s_{ID} corresponding to the replaced public-key PK_{ID} when querying the signing oracle and making forgery.

Organization of the Paper

In the next section, we will review some preliminaries required throughout the paper. We review an efficient certificateless signature scheme [9] in Section III. In Section IV, we will show the above scheme fails to resist against the public key replacement attack. We propose an efficient certificateless signature scheme in Section V. In Section VI, we prove the security of the proposed scheme in Zhang's security model [3]. Finally, Section VII concludes the paper.

II. PRELIMINARIES

In this section, we will review some fundamental backgrounds required in this paper, namely bilinear pairing, the definition and security model of the certificateless signature scheme.

A. Bilinear Pairing

Let $\mathbb{G}_1$ denote an additive group of prime order q and $\mathbb{G}_2$ be a multiplicative group of the same order. Let P denote a generator in $\mathbb{G}_1$. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a bilinear mapping with the following properties:

- The map e is bilinear: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1, a, b \in \mathbb{Z}_q$.
- The map e is non-degenerate: $e(P, P) \neq 1_{\mathbb{G}_2}$.
- The map e is efficiently computable.

Definition 1: (Computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$) Given (P, aP, bP), for some $a, b \in \mathbb{Z}_q^*$, compute abP.

The success probability of any probabilistic polynomial-time algorithm $\mathcal{A}$ in solving CDH problem in $\mathbb{G}_1$ is defined to be

$$\mathtt{Succ}_{\mathcal{A},\mathbb{G}_1}^{CDH} = Pr[\mathcal{A}(P, aP, bP) = abP : a, b \in \mathbb{Z}_q^*].$$

The CDH assumption states that for every probabilistic polynomial-time algorithm $\mathcal{A}$, $\mathtt{Succ}_{\mathcal{A},\mathbb{G}_1}^{CDH}$ is negligible.

B. Certificateless Signature Schemes

A certificateless signature scheme is defined by seven algorithms (see literature [2]): Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key, Set-Public-Key, CL-Sign, and CL-Verify. The description of each algorithm is as follows.

- Setup: The master key and parameter generation algorithm is a probabilistic algorithm that accepts as input a security parameter 1^k and returns a master-key and a parameter list params.
- Partial-Private-Key-Extract: The partial private key issuance algorithm is a deterministic algorithm that accepts as input a user identity ID, a parameter list param and a master-key to produce the user's partial private key D_{ID}.
- Set-Secret-Value: The set secret value setup algorithm is a probabilistic algorithm that accepts as input a parameter list param and a user identity ID to produce the user's secret value s_{ID}.
- Set-Private-Key: The set private key setup algorithm is a probabilistic algorithm that accepts as input a parameter list param, the user's partial private key D_{ID} and the user's secret value s_{ID} to produce a private signing key SK_{ID}.
- Set-Public-Key: The public key generation algorithm is a deterministic algorithm that takes as input a parameter list param, a user identity ID and the user's secret value s_{ID} to produce a public key PK_{ID}.
- CL-Sign: The signing algorithm is a probabilistic algorithm that accepts a message $M \in \mathcal{M}$, $\mathcal{M}$ is the message space, a user's identity ID, a parameter list param and the user's signing key SK_{ID} to produce a signature σ.
- CL-Verify: The verification algorithm is a deterministic algorithm that accepts a message M, a signature σ, a parameter list param, the public key PK_{ID} and the user's identity ID to output `true` if the signature is correct, otherwise outputs $\perp$.

C. Adversarial Model of Certificateless Signature Schemes

As defined in [2], there are two types of adversary with different capabilities:

Type I Adversary: This type of adversary $\mathcal{A}_I$ does not have access to the master-key, but $\mathcal{A}_I$ has the ability to *replace* the public key of any entity with a value of his choice, because there is no certificate involved in certificateless signature schemes.

Type II Adversary: This type of adversary $\mathcal{A}_{II}$ has access to the master-key but cannot perform public keys replacement.

Nevertheless, no formal security model was presented in [2]. Huang *et al.* [11] and Zhang *et al.* [3] provided a formal definition of existential unforgeability of a certificateless signature

scheme under both two types of chosen message attack, respectively. In this section, we review the security model proposed by Zhang *et al.* [3], which are defined using the following two games between an adversary $\mathcal{A} \in \{\mathcal{A}_I, \mathcal{A}_{II}\}$ and a challenger.

Game-I: This is the game in which $\mathcal{A}_I$ interacts with the challenger:

- **Phase I-1:** The challenger runs Setup(1^k) for generating master-key and params. The challenger then gives params to $\mathcal{A}_I$ while keeping master-key secret.
- **Phase I-2:** $\mathcal{A}_I$ performs the following oracle-query operations:
  - **Extract Partial Private Key:** Each of which is denoted by (ID, "partial key extract"). On receiving such a query, the challenger computes D_{ID} = Partial-Private-Key-Extract(params, master-key, ID) and returns it to $\mathcal{A}_I$.
  - **Extract Private Key:** Each of which is denoted by (ID, "private key extract"). Upon receiving such a query, the challenger first computes D_{ID} = Partial-Private-Key-Extract(params, master-key, ID) and then s_{ID} = Set-Secret-Value (params, ID) as well as SK_{ID} = Set-Private-Key (params, D_{ID}, s_{ID}). It returns SK_{ID} to $\mathcal{A}_I$.
  - **Request Public Key:** Each of which is denoted by (ID, "public key request"). Upon receiving such a query, the challenger computes D_{ID} = Partial-Private-Key-Extract(params, master-key, ID), and s_{ID} = Set-Secret-Value (params, ID). It then computes PK_{ID} = Set-Public-Key (params, s_{ID}) and returns it to $\mathcal{A}_I$.

    Replace Public Key: $\mathcal{A}_I$ may replace a public key PK_{ID} with a value chosen by him. It is not required for $\mathcal{A}_I$ to provide the corresponding secret value when making this query.
  - **Signing Queries:** Each of which is of the form (ID, M, "signature"). On receiving such a query, the challenger finds SK_{ID} from its "query-answer" list, computes σ =CL-Sign(params, M, ID, SK_{ID}), and returns it to $\mathcal{A}_I$. If the public key PK_{ID} has been replaced by $\mathcal{A}_I$, then the challenger cannot find SK_{ID} and thus the signing oracle's answer may be incorrect. In such case, we assume that $\mathcal{A}_I$ may additionally submit the secret information s_{ID} corresponding to the replaced public-key PK_{ID} to the signing oracle.
- **Phase I-3:** Finally, $\mathcal{A}_I$ outputs a message M^*, and a signature σ^* corresponding to a target identity ID^* and a public key PK_{ID^*}. Note that ID^* cannot be an identity for which the private key has been extracted. Also, ID^* cannot be an identity for which both the public key has been replaced and the partial private key has been extracted. Moreover, M^* should not be queried to the signing oracle with respect to ID^* and PK_{ID^*}. However, in case the PK_{ID^*} is different from the original public key of the entity with identity ID^*, $\mathcal{A}_I$ does not need to provide the corresponding secret value to the challenger if it has not made signing queries for the identity ID^* and public key PK_{ID^*}.

Game II: This is a game in which $\mathcal{A}_{II}$ interacts with the challenger.

- **Phase II-1:** The challenger runs Setup($\cdot$) to generate master-key and params. The challenger gives both params and master-key to $\mathcal{A}_{II}$.
- **Phase II-2:** $\mathcal{A}_{II}$ performs the following operations:
  - Compute partial private key associated with ID: $\mathcal{A}_{II}$ computes D_{ID} = Partial-Private-Key-Extract(params, master-key, ID). This can be done by $\mathcal{A}_{II}$ since it holds the master key.
  - Make private key extraction queries: On receiving such a query, the challenger computes D_{ID} = Partial-Private-Key-Extract(params, master-key, ID), s_{ID} = Set-Secret-Value (params, ID), and SK_{ID} =Set-Private-Key(params, D_{ID}, s_{ID}). It then returns SK_{ID} to $\mathcal{A}_{II}$.
    Make public key request queries: On receiving such a query, the challenger sets D_{ID} = Partial-Private-Key-Extract(params, master-key, ID), s_{ID} = Set-Secret-Value (params, ID), and then computes PK_{ID} = Set-Public-Key(params, s_{ID}, ID). It returns PK_{ID} to $\mathcal{A}_{II}$.
    Make signing queries: On receiving such a query, the challenger finds SK_{ID} from its "query-answer" list, computes σ = CL-Sign (params, M, ID, SK_{ID}), and returns it to $\mathcal{A}_{II}$
- **Phase II-3:** $\mathcal{A}_{II}$ outputs a message M^*, and a signature σ^* corresponding to a target identity ID^* and a public key PK_{ID^*}. Note that ID^* has not been issued as a private key query. Moreover, M^* should not be queried to the signing oracle with respect to ID^* and PK_{ID^*}.

Definition 2: A certificateless signature scheme is existentially unforgeable against chosen-message attacks if for any probabilistic polynomial time adversary $\mathcal{A}$ ($\mathcal{A}_I$ or $\mathcal{A}_{II}$) succeeds in the above games (Game I or Game II)(i.e., CL-Verify(params,$PK_{ID^*}, M^*, ID^*, \sigma^*$) = 1) is negligible. In other words,

$$Succ_{\mathcal{A}}^{EF-CLS-CMA}(k) \leq \epsilon$$

where k is the system's security parameter.

III. REVIEW OF AN EFFICIENT CERTIFICATELESS SIGNATURE SCHEME

In this secion, we review an efficient certificateless signature Scheme [9] proposed by Yap *et al.* The scheme is as follows:

1. Setup: Given a security parameter k, the algorithm works as follows:
   (a) Let $(\mathbb{G}_1, \mathbb{G}_2)$ be groups with prime order q. e denotes the bilinear pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
   (b) Choose an arbitrary generator of $P \in \mathbb{G}_1$.
   (c) Select a random $s \in \mathbb{Z}_q^*$ and set $P_0 = sP$.
   (d) Choose a cryptographic hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$.
   The system parameters are params $=< \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_0, H_1, H_2 >$. The message space and master-key are $M = \{0,1\}^*$ and $s \in \mathbb{Z}_q^*$, respectively.
2. Set-Partial-Private-Key: Given params and master-key, this algorithm works as follows:
   Computes $Q_A = H_1(ID_A) \in \mathbb{G}_1$ and outputs a partial private key $D_A = sQ_A \in \mathbb{G}_1$.

3. Set-Secret-Key: Given params, select a random value $x_A \in \mathbb{Z}_q^*$, where x_A is the secret value.
4. Set-Private-Key: Set private key $S_A = (x_A Q_A + D_A)$.
5. Set-Public-Key: Given params and the secret value x_A, this algorithm computes $P_A = x_A P \in \mathbb{G}_1$.
6. Sign: Given params, ID_A, message m and private key S_A, the algorithm works as follows:
   (a) Compute $Q_A = H_1(ID_A) \in \mathbb{G}_1$.
   (b) Select a random value $r \in \mathbb{Z}_q^*$ and set $U = rQ_A \in \mathbb{G}_1$.
   (c) Set $h = H_2(m||U) \in \mathbb{Z}_q^*$.
   (d) Compute $V = (r + h)S_A$.
   (e) Set $\sigma = (U, V)$ as the signature of m.
7. Verify: Given signature σ, ID_A, m, P_A, this algorithm works as follows:
   (a) Compute $Q_A = H_1(ID_A) \in \mathbb{G}_1$.
   (b) Compute $h = H_2(m||U) \in \mathbb{Z}_q^*$.
   (c) Check whether $< P, P_0 + P_A, U + hQ_A, V >$ is a valid Diffie-Hellman tuple, i.e., by verifying whether $e(P, V) = e(P_0 + P_A, U + hQ_A)$. If not, then reject the signature else accept it.

In a CLS, Setup and Setup-Partial-Private-Key are performed by a KGC. A partial private key D_A is given to a user A by the KGC through a secure channel.

IV. SECURITY ANALYSIS OF YAP *ET AL.*'S CERTIFICATELESS SIGNATURE SCHEME

Yap *et al.* claimed that their scheme is provably secure in the random oracle model [13] under the CDH assumption. Parallelled with the work of [18], as we will show in this section, their certificateless signature scheme does not resist against type I adversary as defined in [2]. By replacing the public key P_A of the user A, a type I adversary $\mathcal{A}_I$ can corrupt this user's private key S_A, and therefore, can output a forged signature on any message. The attack method is as follows:

1. Public-Key-Replacement: Firstly, $\mathcal{A}_I$ chooses a random $r \in \mathbb{Z}_q^*$, then he/she replaces the target user's public key as $P_A = rP - P_0$.
2. Private-Key-Corruption: Secondly, $\mathcal{A}_I$ computes the target user's private key $S_A = rQ_A$. S_A is the valid private key since

$$S_A = x_A Q_A + D_A = (r - s)Q_A + sQ_A = rQ_A.$$

3. Signature-Forgery: Finally, for a message m, $\mathcal{A}_I$ uses S_A to compute signature (U, V) by the same way as the Sign algorithm defined in Section III. Evidently (U, V) is a valid signature.

Remarks: In the common public key replacement attack, the adversary only replaces target user's public key and doesn't get corresponding private key. Our attack method can not only replace target user's public key but also obtain the full private key corresponding to modified public key of the user. Hence, anyone can forge a valid certificateless signature. The main cause that Yap *et al.*'s scheme does not resist against public key replacement attack if their key generation is not proper. In Yap *et al.*'s scheme [9], Adversary chooses a random $r' \in \mathbb{Z}_q^*$ and replaces public key as $P'_A = r'P - P_0$, which means corresponding secret key is $x'_A = r' - s$. Though the adversary does not know the master-key of the KGC, he/she can eliminate the master-key of the KGC and compute the full private key of the target user by the equation $S'_A = (x'_A Q_A + D_A) = ((r' - s)Q_A + D_A) = r'Q_A$. Therefore, the adversary with private key S'_A is easy to forge a valid certificateless signature according to signature step of the original scheme.

We also show Yap *et al.*'s scheme is insecure by the usual public key replacement attack. Our attack method is as follows: The adversary firstly chooses a random $r \in \mathbb{Z}_q^*$, then he/she replaces target user's public key as $P_A = rP - P_0$. Secondly, he/she randomly selects $U \in \mathbb{G}_1$, computes $Q_A = H_1(ID_A) \in \mathbb{G}_1$ and $h = H_2(m||U) \in \mathbb{Z}_q^*$. Finally, he/she computes $V = r(U + hQ_A)$. Then the tuple (U, V) is a valid certificateless signature scheme. We can see that verification will pass because the following verification equation holds

$$\begin{aligned} e(P_0 + P_A, U + hQ_A) &= e(rP, U + hQ_A) \\ &= e(P, r(U + hQ_A)) \\ &= e(P, V). \end{aligned}$$

V. A NEW AND EFFICIENT CERTIFICATELESS SIGNATURE SCHEME

In this secion, we present an efficient certificateless signature scheme. The scheme is as follows:

1. Setup: Given a security parameter k, the algorithm works as follows:
   (a) Let $(\mathbb{G}_1, \mathbb{G}_2)$ be groups with prime order q. e denotes the bilinear pairing $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
   (b) Choose an arbitrary generator of $P \in \mathbb{G}_1$.
   (c) Select a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$.
   (d) Choose a cryptographic hash function $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_1$, $H_3 : \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{Z}_q^*$.

   The system parameters are params $=< \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, H_1, H_2, H_3 >$. The message space is $M = \{0,1\}^*$. The master-key is $s \in \mathbb{Z}_q^*$.
2. Set-Partial-Private-Key : Given params and master-key, this algorithm works as follows: Compute $Q_A = H_1(ID_A) \in \mathbb{G}_1$ and output a partial private key $D_A = sQ_A \in \mathbb{G}_1$.
3. Set-Secret-Key: Given params, select a random value $x_A \in \mathbb{Z}_q^*$, where x_A is the secret value.
4. Set-Private-Key: Given params, the partial private key D_A, the secret value x_A and outputs full private key $S_A = (D_A, x_A)$.
5. Set-Public-Key: Given params and the secret value x_A, this algorithm computes $P_A = x_A P \in \mathbb{G}_1$.
6. Sign: Given params, ID_A, message m and private key S_A, the algorithm works as follows:
   (a) Compute $Q_A = H_1(ID_A) \in \mathbb{G}_1$.
   (b) Select a random value $r \in \mathbb{Z}_q^*$, compute $U = rQ_A \in \mathbb{G}_1$.
   (c) Compute

$$W = H_2(m||U||P_A) \in \mathbb{G}_1, \ h = H_3(m||U||P_A) \in \mathbb{Z}_q^*.$$

   (d) Compute $V = x_A W + (r + h)D_A$.

(e) Set $\sigma = (U, V)$ as the signature of m.

7. Verify: Given signature σ, ID_A, m, P_A, this algorithm works as follows:
   (a) Compute $Q_A = H_1(ID_A) \in \mathbb{G}_1$.
   (b) Compute

$$W = H_2(m||U||P_A) \in \mathbb{G}_1,\ h = H_3(m||U||P_A) \in \mathbb{Z}_q^*.$$

   (c) Check whether the following verification equation $e(P, V) = e(P_A, W)\, e(P_{pub}, U + hQ_A)$ holds. If not, then reject the signature else accept it.

The correctness of our scheme follows from the fact that $D_A = sQ_A \in \mathbb{G}_1$ and

$$\begin{aligned} e(P, V) &= e(P, x_A W + (r + h)D_A) \\ &= e(P, x_A W)e(P, (r + h)D_A) \\ &= e(P_A, W)e(P, s(r + h)Q_A) \\ &= e(P_A, W)e(P_{pub}, U + hQ_A). \end{aligned}$$

In a CLS, Setup and Set-Partial-Private-Key are performed by a KGC. A partial private key D_A is given to a user A by the KGC through a secure channel.

VI. SECURITY PROOF

Theorem 1: For the $\mathcal{A}_I$ adversary, the proposed certificateless signature scheme is existentially unforgeable against adaptive chosen-message attacks in the random oracle model under the assumption that the CDH problem in $\mathbb{G}_1$ is intractable.

Proof: Let $\mathcal{A}_I$ be a Type I adversary who can break the proposed scheme under adaptive chosen-message attacks. We show how $\mathcal{A}_I$ can be used by a probabilistic polynomial-time (PPT) algorithm $\mathcal{B}$ to solve the CDH problem in $\mathbb{G}_1$. It is interesting to note that the reductionist proof can be obtained without the requirement that $\mathcal{A}_I$ should have submitted the secret information s_{ID} corresponding to the replaced public-key PK_{ID} when querying the signing oracle. Let $(X = aP, Y = bP) \in \mathbb{G}_1 \times \mathbb{G}_1$ be a random instance of the CDH problem taken as input by $\mathcal{B}$. The algorithm $\mathcal{B}$ initializes $\mathcal{A}_I$ with $P_{pub} = X$, and then starts performing oracle simulation. Without loss of generality, we assume that, for any key extraction query or signature query involving an identity, an $H_1(\cdot)$ oracle query has previously been made on that identity. And $\mathcal{B}$ maintains a list $L = (ID, D_{ID}, PK_{ID}, s_{ID})$ while $\mathcal{A}_I$ is making queries throughout the game. $\mathcal{B}$ responds to $\mathcal{A}_I$'s oracle queries as follows.

Queries on Oracle H_1: $\mathcal{B}$ maintains a list $L_1 = (ID, t_1, T)$, where $T \in \{0, 1, \ldots, q_{H_1}\}$, q_{H_1} is the most times for H_1 query.

- If an identity ID has appeared on list $L_1 = (ID, t_1, T)$, then $\mathcal{B}$ responds with $H_1(ID)$ according to T value.
- If an identity ID has not appeared on the list and ID is the I^{th} distinct H_1 query made by $\mathcal{A}_I$, then $\mathcal{B}$ picks $t_1 \in \mathbb{Z}_q^*$ at random and returns $H_1(ID) = t_1 Y \in \mathbb{G}_1$. Otherwise the hash value $H_1(ID)$ is defined as $t_1 P \in \mathbb{G}_1$. In both cases, $\mathcal{B}$ inserts a tuple (ID, t_1, T) in a list $L_1 = (ID, t_1, T)$ to keep track the way it answered the queries.

Partial Private Key Queries: Suppose the request is on an identity ID. $\mathcal{B}$ recovers the corresponding (ID, t_1, T) from the list L_1 (recall that such a tuple must exist because of the aforementioned assumption). If $T = I$, then $\mathcal{B}$ outputs "failure" and halts because it is unable to coherently answer the query. Otherwise, $\mathcal{B}$ looks up the list L and performs as follows.

- If the list L contains $(ID, D_{ID}, PK_{ID}, s_{ID})$, $\mathcal{B}$ checks whether $D_{ID} = \perp$. If $D_{ID} \neq \perp$, $\mathcal{B}$ returns D_{ID} to $\mathcal{A}_I$. If $D_{ID} = \perp$, $\mathcal{B}$ recovers the corresponding (ID, t_1, T) from the list L_1. Noting $T \neq I$ means that $H_1(ID)$ was previously defined to be $t_1 P \in \mathbb{G}_1$ and $D_{ID} = t_1 P_{pub} = t_1 X \in \mathbb{G}_1$ is the partial private key associated to ID. Thus $\mathcal{B}$ returns D_{ID} to $\mathcal{A}_I$ and writes D_{ID} in the list L.
- If the list L does not contain $(ID, D_{ID}, PK_{ID}, s_{ID})$, $\mathcal{B}$ recovers the corresponding (ID, t_1, T) from the list L_1, sets $D_{ID} = t_1 P_{pub} = t_1 X \in \mathbb{G}_1$ and returns D_{ID} to $\mathcal{A}_I$. B also sets $PK_{ID} = s_{ID} = \perp$ and adds an element $(ID, D_{ID}, PK_{ID}, s_{ID})$ to the list L.

Public Key Queries: Suppose the query is made on an identity ID.

- If the list L contains $(ID, D_{ID}, PK_{ID}, s_{ID})$, $\mathcal{B}$ checks whether $PK_{ID} = \perp$. If $PK_{ID} \neq \perp$, $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ randomly chooses $w \in \mathbb{Z}_q^*$ and sets $PK_{ID} = wP$ and $s_{ID} = w$. $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}_I$ and saves (PK_{ID}, s_{ID}) into the list L.
- If the list L does not contain $(ID, D_{ID}, PK_{ID}, s_{ID})$, $\mathcal{B}$ sets $D_{ID} = \perp$, and then randomly chooses $w \in \mathbb{Z}_q^*$ and sets $PK_{ID} = wP$ and $s_{ID} = w$. $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}_I$ and adds $(ID, D_{ID}, PK_{ID}, s_{ID})$ into the list L.

Private Key Extraction Queries: Suppose the query is made on an identity ID. $\mathcal{B}$ recovers the corresponding (ID, t_1, T) from the list L_1. If $T = I$, then $\mathcal{B}$ outputs "failure" and halts because it is unable to coherently answer the query. Otherwise, $\mathcal{B}$ looks up the list L and performs as follows.

- If the list L contains $(ID, D_{ID}, PK_{ID}, s_{ID})$, $\mathcal{B}$ checks whether $D_{ID} = \perp$ and $PK_{ID} = \perp$. If $D_{ID} = \perp$, $\mathcal{B}$ makes a partial private key query itself to obtain D_{ID}. If $PK_{ID} = \perp$, $\mathcal{B}$ makes a public key query itself to generate $(PK_{ID} = wP, s_{ID} = w)$. Then $\mathcal{B}$ saves these values in the list L and returns $SK_{ID} = (D_{ID}, w)$ to $\mathcal{A}_I$.
- If the list L does not contain an item $(ID, D_{ID}, PK_{ID}, s_{ID})$, $\mathcal{B}$ makes a partial private key query and a public key query on ID itself, and then adds $(ID, D_{ID}, PK_{ID}, s_{ID})$ to the list L and returns $SK_{ID} = (D_{ID}, s_{ID})$.

Public Key Replacement Queries: Suppose $\mathcal{A}_I$ makes the query with an input (ID, PK'_{ID}).

- If the list L contains an element $(ID, D_{ID}, PK_{ID}, s_{ID})$, $\mathcal{B}$ sets $PK_{ID} = PK'_{ID}$ and $s_{ID} = \perp$.
- If the list L does not contain an element $(ID, D_{ID}, PK_{ID}, s_{ID})$, $\mathcal{B}$ sets $D_{ID} = \perp$, $PK_{ID} = PK'_{ID}$, $s_{ID} = \perp$, and adds an item $(ID, D_{ID}, PK_{ID}, s_{ID})$ to L.

Queries on Oracle H_2**:** Suppose (m, U, PK_{ID}) is submitted to oracle $H_2(\cdot)$. $\mathcal{B}$ first scans a list $L_2 = (m, U, PK_{ID}, H_2, h_2)$ to check whether H_2 has already been defined for that input. If so, the previously defined value is returned. Otherwise, $\mathcal{B}$ picks at random $h_2 \in \mathbb{Z}_q^*$, and returns $H_2 = h_2 P \in \mathbb{G}_1$ as a hash value of $H_2(m, U, PK_{ID})$ to $\mathcal{A}_I$, and also stores the values in the list L_2.

Queries on Oracle H_3**:** Suppose (m, U, PK_{ID}) is submitted to oracle $H_3(\cdot)$. $\mathcal{B}$ first scans a list $L_3 = (m, U, PK_{ID}, h_3)$ to

check whether H_3 has already been defined for that input. If so, the previously defined value is returned. Otherwise, $\mathcal{B}$ picks at random $h_3 \in \mathbb{Z}_q^*$ as a hash value of $H_3(m, U, PK_{ID})$ and returns it to $\mathcal{A}_I$, and also stores the values in the list L_3.

Signing Oracle Queries: Suppose that $\mathcal{A}_I$ queries the oracle with an input (m, ID). Without loss of generality, we assume that the list L contains an item $(ID, D_{ID}, PK_{ID}, s_{ID})$, and $PK_{ID} \neq \perp$. (If the list L does not contain such an item, or if $PK_{ID} = \perp$, $\mathcal{B}$ runs a public key query to get (PK_{ID}, s_{ID}).) Note that at any time during the simulation, equipped the item $(ID, D_{ID}, PK_{ID}, s_{ID})$ for any $T \neq I$ and $s_{ID} \neq \perp$, $\mathcal{B}$ is easy to generate signature on any message.

For the circumstance $T = I$ or $s_{ID} = \perp$, $\mathcal{B}$ firstly picks at random two numbers $r, h_3 \in \mathbb{Z}_q^*$, sets $U = rP - h_3 Q_{ID}$ and defines the hash value of $H_3(m, U, PK_{ID})$ as h_3 ($\mathcal{B}$ halts and outputs "failure" if $H_3(m, U, PK_{ID})$ turns out to have already been defined for (m, ID, PK_{ID}, U)). Then $\mathcal{B}$ looks up the list L_2 for $(m, U, PK_{ID}, H_2, h_2)$ such that the hash value of $H_2(m, U, PK_{ID})$ has been defined to $H_2 = h_2 P$ (If such an item does not exist, $\mathcal{B}$ makes a query on oracle H_2). Finally, $\mathcal{B}$ sets $V = h_2 PK_{ID} + rP_{pub}$. Now (U, V) is returned to $\mathcal{A}_I$, which appears to be a valid signature since

$$\begin{aligned} & e(PK_{ID}, H_2(m||U||PK_{ID})) \times \\ & \qquad e(P_{pub}, U + H_3(m||U||PK_{ID})Q_{ID}) \\ &= e(PK_{ID}, h_2 P) e(P_{pub}, rP - h_3 Q_{ID} + h_3 Q_{ID}) \\ &= e(P, h_2 PK_{ID}) e(P, rP_{pub}) \\ &= e(P, h_2 PK_{ID} + rP_{pub}) \\ &= e(P, V). \end{aligned}$$

Note that, the above simulation for signing queries works even in the strong case that $\mathcal{B}$ does not know the secret value s_{ID} corresponding to the public key PK_{ID} of a user with identity ID.

Forgery: The next step of the simulation is to apply the forking technique formalized in [14]. Let $(m, (U, V), ID_I, PK_{ID_I})$ be a forgery output by $\mathcal{A}_I$ at the end of the attack. If $\mathcal{A}_I$ does not output $ID = ID_I$ as a part of the forgery then $\mathcal{B}$ aborts (the probability that $\mathcal{B}$ does not abort the simulation is $O(1/q_{H_1})$). $\mathcal{B}$ then replays $\mathcal{A}_I$ with the same random tape, but the different choice of the hash function H_3 and the same choice of the hash function H_2 to get another forgery $(m, (U, V'), ID_I, PK_{ID_I})$. Notice that the hash values $h_3 \neq h_3'$ on (m, U, PK_{ID_I}) for the two choices of H_3. From the forking lemma, after a polynomial reply of the $\mathcal{A}_I$ adversary, we obtain two valid signatures $(m, (U, V), ID_I, PK_{ID_I})$ and $(m, (U, V'), ID_I, PK_{ID_I})$ with $h_3 \neq h_3'$. Then we have the following equalities,

$$\begin{aligned} e(P, V) &= e(PK_{ID_I}, H_2) e(P_{pub}, U + h_3 Q_I), \\ e(P, V') &= e(PK_{ID_I}, H_2) e(P_{pub}, U + h_3' Q_I), \end{aligned}$$

from which we obtain

$$e(P, V - V') = e(P_{pub}, (h_3 - h_3')Q_I) = e(aP, (h_3 - h_3')t_1 bP),$$

i.e., $V - V' = (h_3 - h_3')t_1 abP$. Thus, it is not difficult to see that $abP = (h_3 - h_3')^{-1} t_1^{-1} (V - V')$. This completes our proof. □

Theorem 2: If a PPT forger $\mathcal{A}_{II}$ has an advantage ε in forging a signature in an attack modeled by Game II after running in time t and making at most q_{H_i} queries to random oracles H_i for $i = 2, 3$, at most q_E queries to the private-key extraction oracle, at most q_{PK} queries to the public-key request oracle, at most q_S queries to the signing oracle, and e is the base of the natural logarithm, then the CDH problem can be solved with probability $\varepsilon' \geqslant \frac{1}{eq_E}(1 - (q_S q_{H_2} - 2)/2^k)\varepsilon$, within time $t' < t + (q_{H_2} + q_{PK} + 3q_S)t_m$, where t_m is the time to compute a scalar multiplication in $\mathbb{G}_1$.

Proof: Suppose $\mathcal{A}_{II}$ is a Type II adversary that (t, ε)-breaks our certificateless signature scheme. We show how to construct a t'-time algorithm $\mathcal{B}$ that solves the CDH problem on G_1 with probability at least ε'. Let $(X = aP, Y = bP \in \mathbb{G}_1 \times \mathbb{G}_1)$ be a random instance of the CDH problem taken as input by $\mathcal{B}$. $\mathcal{B}$ randomly chooses $s \in \mathbb{Z}_q^*$ as the master key, and then initializes $\mathcal{A}_{II}$ with $P_{pub} = sP$ and also the master key s. The adversary $\mathcal{A}_{II}$ then starts making oracle queries such as those described in Game II. Note that the partial private key $D_{ID} = sH_1(ID)$ can be computed by both $\mathcal{B}$ and $\mathcal{A}_{II}$, thus the hash function $H_1(\cdot)$ is not modeled as a random oracle in this case. $\mathcal{B}$ maintains a list $L = \{(ID, PK_{ID}, s_{ID}, T)\}$, which does not need to be made in advance and is populated when $\mathcal{A}_{II}$ makes certain queries specified below.

Queries on Oracle H_2: Suppose a tuple (m, U, PK_{ID}) is submitted to oracle $H_2(\cdot)$. $\mathcal{B}$ first scans a list $L_2 = \{(m, U, PK_{ID}, H_2, h_2)\}$ to check whether H_2 has already been defined for that input. If so, the previously defined value is returned. Otherwise, $\mathcal{B}$ selects at random $h_2 \in \mathbb{Z}_q^*$ and defines the value of $H_2(m, U, PK_{ID})$ as $H_2 = h_2 Y \in G_1$. $\mathcal{B}$ adds the tuple $(m, U, PK_{ID}, H_2, h_2)$ into the list L_2 and responds to $\mathcal{A}_{II}$ by seting $H_2(m, U, PK_{ID}) = H_2$.

Queries on Oracle H_3: When a tuple (m, U, PK_{ID}) is submitted to oracle $H_3(\cdot)$. $\mathcal{B}$ first scans a list $L_3 = \{(m, U, PK_{ID}, h_3)\}$ to check whether H_3 has already been defined for that input. If so, the existing value is returned. Otherwise, $\mathcal{B}$ picks at random $h_3 \in \mathbb{Z}_q^*$ as a hash value of $H_3(m, U, PK_{ID})$ and returns it to $\mathcal{A}_{II}$, and also stores the values in the list L_3.

Public Key Queries: Suppose the query is made on an identity ID.

- If the list L contains (ID, PK_{ID}, s_{ID}, T), $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}_{II}$.
- If the list L does not contain (ID, PK_{ID}, s_{ID}, T), $\mathcal{B}$ picks a number $w \in \mathbb{Z}_q^*$ at random. $\mathcal{B}$ flips a coin $T \in \{0, 1\}$ that yields 0 with probability δ and 1 with probability $1 - \delta$. If $T = 0$, the value of PK_{ID} is defined as $wP \in \mathbb{G}_1$. If $T = 1$, the value of PK_{ID} is defined as $wX \in \mathbb{G}_1$. In both cases, $\mathcal{B}$ sets $s_{ID} = w$, and inserts a tuple (ID, PK_{ID}, s_{ID}, T) into a list $L = \{(ID, PK_{ID}, s_{ID}, T)\}$ to keep track the way it answered the queries. $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}_{II}$.

Private Key Extraction Queries: Suppose the query is made on an identity ID.

- If the list L contains (ID, PK_{ID}, s_{ID}, T), $\mathcal{B}$ returns $SK_{ID} = (D_{ID}, s_{ID})$ to $\mathcal{A}_{II}$ if $T = 0$, and fails otherwise.
- If the list L does not contain (ID, PK_{ID}, s_{ID}, T), $\mathcal{B}$ makes a public key query on ID itself, and adds (ID, PK_{ID}, s_{ID}, T) into the list L. Then $\mathcal{B}$ returns $SK_{ID} = (D_{ID}, s_{ID})$ to $\mathcal{A}_{II}$ if $T = 0$, and fails otherwise.

Signing Oracle Queries: Suppose $\mathcal{A}_{II}$ makes the query with an input (m, ID, PK_{ID}). Without loss of generality, we assume that there is an item (ID, PK_{ID}, s_{ID}, T) in the list L.

- If $T = 0$, due to knowing $SK_{ID} = (D_{ID}, s_{ID})$, $\mathcal{B}$ is easy to generate signature on any message.
- If $T = 1$, $\mathcal{B}$ firstly chooses at random $h_2 \in \mathbb{Z}_q^*$, $U \in \mathbb{G}_1$ and defines the hash value of $H_2(m, U, PK_{ID})$ as $h_2 P$ ($\mathcal{B}$ halts and outputs "failure" if $H_2(m, U, PK_{ID})$ turns out to have already been defined for (m, ID, PK_{ID}, U)). Then $\mathcal{B}$ looks up the list L_3 for (m, U, PK_{ID}, h_3) such that the hash value of $H_3(m, U, PK_{ID})$ has been defined to $H_3 = h_3$ (If such an item does not exist, $\mathcal{B}$ makes a query on oracle H_3). Finally, $\mathcal{B}$ sets $V = h_2 PK_{ID} + sU + h_3 D_{ID}$. Now (U, V) is returned to $\mathcal{A}_{II}$, which appears to be a valid signature since

$$\begin{aligned} e(P, V) &= e(P, h_2 PK_{ID} + sU + h_3 D_{ID}) \\ &= e(P, h_2 PK_{ID}) \cdot e(P, sU + sh_3 Q_{ID}) \\ &= e(PK_{ID}, H_2(m||U||PK_{ID})) \\ &\quad \times e(P_{pub}, U + H_3(m||U||PK_{ID}) Q_{ID}). \end{aligned}$$

Output: Eventually $\mathcal{A}_{II}$ outputs a spurious signature $\sigma^* = (U^*, V^*)$ on a message M^* corresponding to a target identity ID^* and a public key PK_{ID^*} such that ID^* has not been issued as a private key query. Moreover, M^* should not be queried to the signing oracle with respect to ID^* and PK_{ID^*}. Then $\mathcal{B}$ recovers $(ID^*, PK_{ID^*}, s_{ID^*}, T^*)$ from L and evaluates T^*. If $T^* = 0$, then $\mathcal{B}$ outputs failure and stops. Otherwise, it looks up an item $(m^*, U^*, PK_{ID^*}, H_2^*, h_2^*)$ in the list L_2 such that the value of $H_2(m^*, U^*, PK_{ID^*})$ has been defined as $h_2^* Y \in G_1$. Then $\mathcal{B}$ looks up an item $(m^*, U^*, PK_{ID^*}, h_3^*)$ in the list L_3 such that the value of $H_3(m^*, U^*, PK_{ID^*})$ has been defined to be h_3^*. Note that the lists L_2 and L_3 must contain such entries with overwhelming probability. If $\mathcal{A}_{II}$ succeeds in the game, then the following equation is satisfied:

$$\begin{aligned} e(P, V^*) &= e(PK_{ID^*}, H_2(m^*||U^*||PK_{ID^*})) \\ &\quad \times e(P_{pub}, U^* + H_3(m^*||U^*||PK_{ID^*}) Q_{ID^*}) \\ &= e(s_{ID^*} X, h_2^* Y) e(sP, U^* + h_3^* Q_{ID^*}) \\ &= e(P, s_{ID^*} h_2^* abP) e(P, sU^* + h_3^* D_{ID^*}). \end{aligned}$$

Therefore, $e(P, V^* - sU^* - h_3^* D_{ID^*}) = e(P, s_{ID^*} h_2^* abP)$. Due to knowing $s, h_3^*, h_2^*, s_{ID^*} \in \mathbb{Z}_q^*$, $abP = (s_{ID^*} h_2^*)^{-1}(V^* - sU^* - h_3^* D_{ID^*})$ is the solution to the CDH instance (X, Y).

This completes the description of $\mathcal{B}$. It remains to show that $\mathcal{B}$ solves the given instance of the CDH problem in $\mathbb{G}_1$ with probability at least ε'. To do so, we analyze the following three events needed for $\mathcal{B}$ to succeed:

- ξ_1: $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}_{II}$'s private key extraction queries and signature queries.
- ξ_2: $\mathcal{A}_{II}$ generates a valid message-signature forgery (ID^*, M^*, σ^*).
- ξ_3: Event ξ_2 and $T^* = 1$ for the tuple containing M^* on the list L.

$\mathcal{B}$ succeeds if all of these events happen. The probability $Pr[\xi_1 \wedge \xi_2 \wedge \xi_3]$ decomposes as $Pr[\xi_1 \wedge \xi_2 \wedge \xi_3] = Pr[\xi_1] Pr[\xi_2|\xi_1] Pr[\xi_3|\xi_1 \wedge \xi_2]$. The following claims give a lower bound for each of these terms.

Claim 1. $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}_{II}$'s private key extraction queries and signature queries is at least $\delta^{q_E}(1 - (q_S q_{H_2} - 2)/2^k)$, hence

$$Pr[\xi_1] \geqslant \delta^{q_E}(1 - (q_S q_{H_2} - 2)/2^k).$$

Proof: The probability that $\mathcal{B}$ answers to all private key extraction queries is at least δ^{q_E}. Next we compute the probability that $\mathcal{B}$ answers to all signature queries. Our simulation for oracle H_3 is perfect. Due to a conflict on H_2, the probability that $\mathcal{B}$ fails to handle a signature query is at most $q_S q_{H_2}/2^k$. The probability for $\mathcal{A}_{II}$ to output a valid forgery σ^* on a message M^* for identity ID^* with public key PK_{ID}^*, without asking the corresponding $H_2(m^*||U^*||PK_{ID^*})$ query or $H_3(m^*||U^*||PK_{ID^*})$ query, is at most $2/2^k$. Therefore, the probability that $\mathcal{B}$ answers to all signature queries is $1 - (q_S q_{H_2} - 2)/2^k$. Thus, $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}_{II}$'s private key extraction queries and signature queries is $\delta^{q_E}(1 - (q_S q_{H_2} - 2)/2^k)$. □

Claim 2. If $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}_{II}$'s private key extraction queries and signature queries then $\mathcal{A}_{II}$'s view is identical to its view in the real attack. Hence, $Pr[\xi_2|\xi_1] \geqslant \varepsilon$.

Claim 3. The probability that $\mathcal{B}$ does not abort after $\mathcal{A}_{II}$ outputs a valid forgery is at least $1 - \delta$. Hence, $Pr[\xi_3|\xi_1 \wedge \xi_2] \geqslant 1 - \delta$.

The proofs of the claim 2 and claim 3 are similar with literature [15], the interested reader is referred to it. From above claims, we obtain $Pr[\xi_1 \wedge \xi_2 \wedge \xi_3] \geqslant \delta^{q_E}(1 - \delta)(1 - (q_S q_{H_2} - 2)/2^k)\varepsilon$. And, by an analysis similar in [16], we can see that the maximal value of the probability $\delta^{q_E}(1 - \delta)$ is $\frac{1}{q_E}(1 - \frac{1}{q_E + 1})^{q_E + 1} \simeq \frac{1}{eq_E}$ (for sufficiently larger q_E) when the $\delta = 1 - 1/(q_E + 1)$ is taken. Thus,

$$Pr[\xi_1 \wedge \xi_2 \wedge \xi_3] \geqslant \frac{1}{eq_E}(1 - (q_S q_{H_2} - 2)/2^k)\varepsilon.$$

This completes our proof. □

VII. CONCLUSION

In this paper, we reviewed the security of the efficient certificateless signature scheme proposed in [9]. we show that their scheme does not resist against public key replacement attacks, and gives the main cause suffered from attack. Furthermore, we proposed an improved certificateless signature scheme, which is existential unforgeable under adaptive chosen message attacks in the random oracle model and proves the security of the proposed scheme in Zhang's [3] security model.

REFERENCES

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. Advances in Cryptology-Crypto'84, Lecture Notes in Computer Science 196*, Aug. 1984, pp. 47–53.

[2] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. Advances in Cryptography-Asiacrypt2003, Lecture Notes in Computer Science 2894*, Dec. 2003, pp. 452–473.

[3] Z. F. Zhang, D. S. Wong, J. Xu, and D. G. Feng, "Certificateless public-key signature: Security model and efficient construction," in *Proc. ACNS2006, Lecture Notes in Computer Science 3989*, June 2006, pp. 293-308.

[4] S. S. Al-Riyami and K. G. Paterson, "CBE from CLPKE: A generic construction and efficient schemes," in *Proc. Public Key Cryptography,*

PKC2005, Lecture Notes in Computer Science 3386, Jan. 2005, pp. 398–415.

[5] J. Baek, R. Safavi-Naini, and W. Susilo,"Certificateless public key encryption without pairing," in *Proc. 8th Information Security Conference, ISC2005, Lecture Notes in Computer Science 3650*, Sept. 2005, pp. 134–148.

[6] D. H. Yum and P. J. Lee,"Generic construction of certificateless encryption," in *Proc. ICCSA2004, Lecture Notes in Computer Science 3043*, May 2004, pp. 802–811.

[7] Z. Cheng and R. Comley, "Efficient certificateless public key encryption," Cryptology ePrint Archive. [Online]. Available: http://eprint.iacr.org/2005/012

[8] D. H. Yum and P. J. Lee, "Generic construction of certificateless signature," in *Proc. Information Security and Privacy, ACISP2004, Lecture Notes in Computer Science 3108*, July 2004, pp. 200–211.

[9] W. S. Yap, S. H. Heng, and B. M. Goi, "An efficient certificateless signature scheme," in *Proc. EUC Workshops2006, Lecture Notes in Computer Science 4097*, Aug. 2006, pp. 322–331.

[10] L. C. Wang, Z. F. Cao, X. X. Li, and H. F. Qian, "Certificateless threshold signature schemes," in *Proc. CIS2005, Lecture Notes in Artifical Intelligence 3802*, Dec. 2005, pp. 104–109.

[11] X. Y. Huang, W. Susilo, Y. Mu, and F. T. Zhang, "On the security of certificateless signature schemes from asiacrypt 2003," in *Proc. CANS2005, Lecture Notes in Computer Science 3810*, Dec. 2005, pp. 13–25.

[12] B. C. Hu, D. S. Wong, Z. F. Zhang, and X. T. Deng, "Key replacement attack against a generic construction of certificateless signature," in *Proc. ACISP2006, Lecture Notes in Computer Science 4058*, July 2006, pp. 235–246.

[13] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. ACM CCS'93*, Nov. 1993, pp. 62–73.

[14] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Proc. Advanced in Cryptology-Eurocrypt'96, Lecture Notes in Computer Science 1070*, 1996, pp. 387–398.

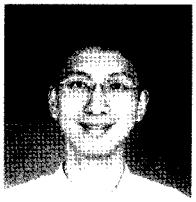
[15] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the weil pairing," in *Proc. Advances in Cryptology - Asiacrypt2001, Lecture Notes in Computer Science 2248*, Dec. 2001, pp. 514–532.

[16] J.S. Coron, "On the exact security of full domain hash," in *Proc. Advances in Cryptology- Crypto2000, Lecture Notes in Computer Science 1880*, Aug. 2000, pp. 229-235.

[17] X. Y. Huang, Y. Mu, W. Susilo, D. S. Wong, and W. Wu, "Certificateless signature revisited," in *Proc. ACISP2007, Lecture Notes in Computer Science 4586*, July 2007, pp. 308–322.

[18] Z. F. Zhang, and D. G. Feng, "Key replacement attack on a certificateless signature scheme," Cryptology ePrint Archive. [Online]. Available: http//eprint.iacr.org/2006/453

Jiguo Liwas born in Heilongjiang Province, China, on December 17, 1970. He received his Bachelor degree from Heilongjiang University, China in 1996. He recieved his Master degree and Ph.D. degree from Harbin Institute of Technology, China, in 2000 and 2003, respectively. He is currently associate professor at the College of Computer and Information Engineering, Hohai University, Nanjing, China. His major interests are cryptography theory and technology, cryptography protocol and network, and information security.



Xinyi Huang received his Bachelor and Master degrees from Nanjing Normal University, China. He is currently a Ph.D. candidate at the School of Computer Science and Software Engineering, University of Wollongong, Australia. His research interest is about cryptography, in particular digital signatures with additional properties.

Yi Mu received his Ph.D. degree from the Australian National University in 1994. He currently is an associate professor in School of Computer Science and Software Engineering and the director of Centre for Computer and Information Security Research, University of Wollongong. Prior to joining University of Wollongong, he worked as a lecturer in the School of Computing and IT, University of Western Sydney, and later as a senior lecturer in the Department of Computing, Macquarie University. His current research interests include network security, computer security, and cryptography. He is the editor-in-chief of International Journal of Applied Cryptography and serves as editor for six other international journals. He has served in program committees for a number of international conferences. He is a senior member of the IEEE and a member of the IACR.

Wei Wu received her Bachelor and Master degrees from Nanjing Normal University, China. She is currently a research fellow at the School of Computer Science and Software Engineering, University of Wollongong, Australia. Her research interest is about cryptography, in particular new public key cryptography system.