Certificateless Public-Key Signature: Security Model and Efficient Construction

Conference Paper · June 2006

DOI: 10.1007/11767480_20 · Source: DBLP

CITATIONS
191

READS
292

4 authors, including:

Zhenfeng Zhang
Chinese Academy of Sciences
50 PUBLICATIONS **1,244** CITATIONS

SEE PROFILE

Duncan S. Wong
City University of Hong Kong
237 PUBLICATIONS **6,027** CITATIONS

SEE PROFILE

Jing Xu
Chinese Academy of Sciences
24 PUBLICATIONS **910** CITATIONS

SEE PROFILE

Certificateless Public-Key Signature: Security Model and Efficient Construction

Zhenfeng Zhang[1], Duncan S. Wong[2], Jing Xu[3], and Dengguo Feng[1]

[1] State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
[2] Department of Computer Science, City University of Hong Kong, Hong Kong, China
[3] Graduate University of Chinese Academy of Sciences, Beijing 100039, China
zfzhang@is.iscas.ac.cn, duncan@cityu.edu.hk

Abstract. "Certificateless public-key cryptosystem" is a new and attractive paradigm, which avoids the inherent key escrow property in identity-based public-key cryptosystems, and does not need expensive certificates as in the public key infrastructure. A strong security model for certificateless public key encryption was established by Al-Riyami and Paterson in 2003. In this paper, we first present a security model for certificateless public-key signature schemes, and then propose an efficient construction based on bilinear pairings. The security of the proposed scheme can be proved to be equivalent to the computational Diffie-Hellman problem in the random oracle model with a tight reduction.

1 Introduction

In traditional certificate-based public key cryptosystems, a user's public-key is generated randomly and is uncorrelated to his identity. The key therefore needs to be certified by some trusted authority with respect to the user's identity. This approach takes the form of digital certificates generated by some trusted Certification Authorities (CAs), which aim at vouching for the fact that a given public-key actually belongs to its alleged owner. Any other user who wants to use the public-key must first verify the corresponding certificate for the validity of the key. Currently, PKI (Public Key Infrastructure) is an important mechanism to maintain certificates and disseminate trust information among users in a hierarchical manner. However, PKIs in practice is very costly to deploy and cumbersome to use.

In [17], Shamir introduced the notion of identity-based (ID-based) public-key cryptography (ID-PKC). This notion is to use a binary string which can uniquely identify a user (or an entity in general) as the user's public key. Examples of such a binary string include email address, IP address, social security number, etc. The motivation of ID-PKC is to simplify key management and remove the need of public key certificates as much as possible. Certificates are only needed for some trusted authorities called ID-based Private Key Generators (PKGs) [15]. These PKGs are responsible for generating private keys for users (unlike

J. Zhou, M. Yung, and F. Bao (Eds.): ACNS 2006, LNCS 3989, pp. 293–308, 2006.

conventional public key schemes, users do not generate their own private keys). Although this does not completely remove the need of certificates, it drastically reduces the cost and complexity of the system as individual end users do not need to obtain certificates for their public keys.

An inherent problem of ID-based cryptography is the key escrow problem, i.e., the private key of a user is known to the PKG. The PKG can literally decrypt any ciphertext and forge signature on any message as any user. Therefore, it seems that ID-based cryptography may only be suitable for small private networks with low security requirements. To tackle this problem, several proposals have been made using multiple authority approach [5, 8]. If the master key of the PKG is distributed over several authorities and a private key is constructed in some threshold manner [5], key escrow problem may be alleviated. However, in many applications, multiple identifications for a user by multiple authorities can be quite a burden. Generating a new private key by adding multiple private keys [8] is another approach, but in this scheme, PKGs have no countermeasure against users' illegal usage. In [12], Gentry proposed an approach in which the private key is generated using some user-chosen secret information.

Independent of Gentry's work, in [1], Al-Riyami and Paterson successfully removed the necessity of certificates in a similar manner to that of user-chosen secrets, and they referred to it as certificateless public key cryptography (CL-PKC). Unlike ID-based cryptography, a user's private key in a CL-PKC scheme is not generated by the Key Generation Center (KGC) alone. Instead, it is a combination of some contribution of the KGC (called partial-private-key) and some user-chosen secret, in such a way that the key escrow problem can be solved. In particular, the KGC cannot obtain the user's private-key. Meanwhile, CL-PKC schemes are not purely ID-based, and there exists an additional public-key for each user. In order to encrypt a message, one has to know both the user's identity and this additional public key. Moreover and more importantly, this additional public key does not need to be certified by any trusted authority. The structure of the scheme ensures that the key can be verified without a certificate. In [1], a strong security model was established for certificateless public key encryption. The model has two types of adversaries. Type I adversary represents a malicious third party and Type II adversary represents a malicious KGC. More details of these two types of adversaries will be given in the later part of this paper. In [1, 2], efficient constructions of certificateless encryption based on bilinear pairings were proposed, and in [7], a scheme not using bilinear pairings was proposed.

Note that key escrow means that the KGC knows the key, not generates the key. In CL-PKC, the KGC can still generate keys but the KGC does not know the key if it is generated by the user. In ID-based cryptography, the user cannot generate a key for himself. Hence the key escrow problem is inherent. In certificate-based cryptography or CL-PKC, both the CA or the KGC, respectively, can generate the key. Therefore, we should assume the similar degree of trust on the CA to that of the KGC in this aspect. In other words, CL-PKC *retains* the beauty of PKI/CA that the KGC does not *know* the key generated by the user.

In conventional application of digital signature, no signer wants his signing key to be escrowed by others. So it seems to be more urgent to solve the key escrow problem as in ID-based setting. In [1], a certificateless public-key signature (CL-PKS) scheme was also proposed. However, a security model for analyzing the scheme's security in terms of unforgeability was not formalized. Also, the scheme was recently found to be vulnerable to key replacement attack [13]. The key replacement attack is launched by the malicious third party (i.e. Type I adversary) who replaces the additional public key of the targeting user with another key chosen by the adversary. By using this attack, the adversary can successfully forge signature on any message as the user. In [13], Huang et al. also proposed an improved scheme. In their security analysis, the Type I adversary can replace an entity's public-key, but is also required to provide a replacing secret value corresponding to the replacing public key. Hence the challenger (or game simulator) also learns the replaced secret. This restriction seems too strong to be reasonable. Due to the un-certified feature of user's public-key in the certificateless setting, a signer does not need to provide any proof about his knowledge of the corresponding secret value of the public-key. In addition, a signature verifier does not check whether a signer knows the secret either. The model in [18] did not deal with this either. We will discuss this further when presenting our security model..

Moreover, for a CL-PKS scheme, the validity of a certificateless signature and the validity of a un-certified public-key can be verified at the same time, which is different from certificateless public-key encryption, where the encryptor does not know whether the public-key used to encrypt is valid or not.

In this paper, we first develop a security model for CL-PKS schemes. The model captures the notion of existential unforgeability of certificateless signature against Type I and Type II adversaries. We then propose an efficient and simple certificateless public-key signature scheme and show its security in our model.

Paper organization. The rest of the paper is organized as follows. A security model for CL-PKS is given in Section 2. In Section 3, we propose a CL-PKS scheme based on bilinear pairings. Its security is analyzed in Section 4. We conclude the paper in Section 5.

2 Security Model for Certificateless Public-Key Signature

Definition 1 (CL-PKS). A CL-PKS (Certificateless Public Key Signature) scheme, Π, consists of the following probabilistic, polynomial-time algorithms:

- `Setup`: It takes as input a security parameter 1^k, and returns a list `params` of system parameters and a master private key `masterKey`.
  The parameter list `params` also defines message space $\mathcal{M}$, and is publicly known. The algorithm is assumed to be run by a Key Generation Center (KGC) for the initial setup of a certificateless system.
- `Partial-Private-Key-Extract`: It takes as inputs `params`, `masterKey` and a user identity $ID \in \{0,1\}^*$, and outputs a partial private key D_{ID}.

This algorithm is run by the KGC once for each user, and the partial private key generated is assumed to be distributed securely to the corresponding user.

- `Set-Secret-Value`: Taking as inputs `params` and a user's identity ID, this algorithm generates a secret value s_{ID}. This algorithm is supposed to be run by each user in the system.
- `Set-Private-Key`: This algorithm takes `params`, a user's partial private key D_{ID} and his secret value s_{ID}, and outputs the full private key SK_{ID}. This algorithm is run by each user.
- `Set-Public-Key`: It takes as inputs `params` and a user's secret value s_{ID}, and generates a public key PK_{ID} for that user. This algorithm is run by the user, and the resulting public key is assumed to be publicly known.
- `CL-Sign`: This is the certificateless signing algorithm. It takes as inputs `params`, a message m, a user's identity ID, and the user's full private key SK_{ID}, and outputs a signature σ.
- `CL-Verify`: This is the verification algorithm, a deterministic algorithm that takes as inputs `params`, a public key PK_{ID}, a message M, a user's identity ID, and a signature σ, and returns a bit b. $b = 1$ means that the signature is accepted, whereas $b = 0$ means rejected.

For security analysis of a CL-PKS, we extend the model for ID-based signature schemes so that the extension allows an adversary to extract partial private keys, or private keys, or both. We must also consider the ability of the adversary to replace the public key of any user with a value of his choice, because there is no certificate in a certificateless signature scheme.

Five oracles can be accessed by the adversary. The first is a partial private key exposure oracle that returns D_{ID} on input a user's identity ID. The second is a private key exposure oracle that returns SK_{ID} on input a user's identity ID if that user's public-key has not been replaced. The third is a public key request oracle that returns PK_{ID} on input an identity ID. The fourth is a public key replacement oracle that replaces the public key PK_{ID} with PK'_{ID} for a user with identity ID . The fifth is a signing oracle $O_{\texttt{CL-Sign}}(\cdot)$ that returns $\texttt{CL-Sign}(\texttt{params}, m, ID, SK_{ID})$ on input (m, ID).

Similar to Al-Riyami and Paterson's certificateless public-key encryption scheme [1], the security of a certificateless signature scheme can be analyzed by considering two types of adversaries. The first type of adversary (Type I) $\mathcal{A}^{\mathrm{I}}$ is meant to represent third party attacks against the existential unforgeability of the scheme. Due to the uncertified nature of the public-keys generated by the users, we must assume that the adversary is able to replace users' public-keys at will. This represents the adversary's ability to fool a user on accepting a signature by using a public key that is supplied by the adversary.

Al-Riyami and Paterson's security model for certificateless encryption allows the adversary to make decryption queries, even for public keys which have already been replaced. This means that the challenger must be able to correctly answer decryption queries for public keys where the corresponding secret keys may not be known to the challenger. This is a very strong security require-

ment, and it is unclear how realistic this restriction is. In fact, several papers [4, 9, 11, 18] have chosen to weaken this requirement so that the challenger is not forced to provide correct decryption of ciphertexts after the corresponding public-key has been replaced. Instead, they only require that ciphertexts can be decrypted if the public key is replaced while the corresponding secret value is also supplied by the adversary.

As for a certificateless public-key signature scheme, a "weakened" security definition *requires that the adversary can only request signatures on identities for which if the public key has been replaced with some value that is not equal to its original value, then the corresponding secret information is also provided during the key replacement.* However, it is not compulsory for the adversary to provide the corresponding secret information when the adversary replaces a public key.

It seems that this weakened requirement is more reasonable, at least for a certificateless signature scheme. First, we can never expect that a signer will produce a valid signature for a public key that he does not know the corresponding private key in the real world. Second, the cost of obtaining a partial private key is much more expensive than generating a pair of secret value s_{ID} and public key PK_{ID} for a user. Therefore, a certificateless signature scheme may be implemented in such a way that the partial-private-key is kept invariant for a long period, while the pair (s_{ID}, PK_{ID}) can be changed arbitrarily by the user. Thus an attack is allowed to replace (s_{ID}, PK_{ID}) with a key pair of adversary's choice when it has access to the terminal-devices..

The second type of adversary (Type II) $\mathcal{A}^{\mathrm{II}}$ for a certificateless signature scheme represents a malicious key generation center.. Here, the adversary is equipped with the key generation center's master key, but cannot replace any user's public key. In fact, if the Type II adversary is allowed to replace an user's public-key, then the adversary can definitely forge signatures of the user. This is the trivial case and is comparable to the damage caused by a malicious Certification Authority (CA) in the conventional certificate-based cryptosystem. Therefore, we do not consider this scenario in certificateless cryptography also.

Definition 2 (EUF-CMA of CL-PKS). Let $\mathcal{A}^{\mathrm{I}}$ and $\mathcal{A}^{\mathrm{II}}$ denote a Type I attacker and a Type II attacker, respectively. Let Π be a CL-PKS scheme. We consider two games "Game I" and "Game II" where $\mathcal{A}^{\mathrm{I}}$ and $\mathcal{A}^{\mathrm{II}}$ interact with their "Challenger" in these two games, respectively. We say that a CL-PKS scheme is existentially unforgeable against adaptive chosen message attacks, if the success probability of both $\mathcal{A}^{\mathrm{I}}$ and $\mathcal{A}^{\mathrm{II}}$ is negligible. Note that the Challenger keeps a history of "query-answer" while interacting with the attackers.

Game-I: This is the game in which $\mathcal{A}^{\mathrm{I}}$ interacts with the "Challenger":

Phase I-1: The Challenger runs $\mathtt{Setup}(1^k)$ for generating `masterKey` and `params`. The Challenger then gives `params` to $\mathcal{A}^{\mathrm{I}}$ while keeping `masterKey` secret.

Phase I-2: $\mathcal{A}^{\mathrm{I}}$ 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`, `masterKey`, ID) and returns it to $\mathcal{A}^{\mathrm{I}}$.
- **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`, `masterKey`, 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 $\mathcal{A}^{\mathrm{I}}$.
- **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`, `masterKey`, ID), and s_{ID}= `Set-Secret-Value` (`params`, ID). It then computes PK_{ID}= `Set-Public-Key` (`params`, s_{ID}) and returns it to $\mathcal{A}^{\mathrm{I}}$.
- **Replace Public Key:** $\mathcal{A}^{\mathrm{I}}$ may replace a public key PK_{ID} with a value chosen by him. It is not required for $\mathcal{A}^{\mathrm{I}}$ 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 $\mathcal{A}^{\mathrm{I}}$. If the public key PK_{ID} has been replaced by $\mathcal{A}^{\mathrm{I}}$, then the Challenger cannot find SK_{ID} and thus the signing oracle's answer may be incorrect. In such case, we assume that $\mathcal{A}^{\mathrm{I}}$ *may additionally submit the secret information* s_{ID} *corresponding to the replaced public-key* PK_{ID} *to the signing oracle.*

Phase I-3: Finally, $\mathcal{A}^{\mathrm{I}}$ 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 that the* PK_{ID^*} *is different from the original public key of the entity with identity* ID^*, $\mathcal{A}^{\mathrm{I}}$ *needs not 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 $\mathcal{A}^{\mathrm{II}}$ interacts with the "Challenger".

- **Phase II-1:** The challenger runs `Setup`($\cdot$) to generate `masterKey` and `params`. The challenger gives both `params` and `masterKey` to $\mathcal{A}^{\mathrm{II}}$.
- **Phase II-2:** $\mathcal{A}^{\mathrm{II}}$ performs the following operations:
  - Compute partial private key associated with ID: $\mathcal{A}^{\mathrm{II}}$ computes D_{ID} = `Partial-Private-Key-Extract`(`params`, `masterKey`, ID). This can be done by $\mathcal{A}^{\mathrm{II}}$ 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`, `masterKey`, 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 $\mathcal{A}^{\mathrm{II}}$.

- Make public key request queries: On receiving such a query, the Challenger sets $D_{ID} = \texttt{Partial-Private-Key-Extract}(\texttt{params}, \texttt{masterKey}, ID)$, $s_{ID} = \texttt{Set-Secret-Value}(\texttt{params}, ID)$, and then computes $PK_{ID} = \texttt{Set-Public-Key}(\texttt{params}, s_{ID}, ID)$. It returns PK_{ID} to $\mathcal{A}^{\mathrm{II}}$.
- Make signing queries: On receiving such a query, the Challenger finds SK_{ID} from its "query-answer" list, computes $\sigma = \texttt{CL-Sign}(\texttt{params}, M, ID, SK_{ID})$, and returns it to $\mathcal{A}^{\mathrm{II}}$.

– **Phase II-3:** $\mathcal{A}^{\mathrm{II}}$ 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^*}.

We say that an adversary $\mathcal{A}$ ($\mathcal{A}^{\mathrm{I}}$ or $\mathcal{A}^{\mathrm{II}}$) succeeds in the above games (Game I or Game II) if $\texttt{CL-Verify}(\texttt{params}, PK_{ID^*}, M^*, ID^*, \sigma^*)$=1. Denote the probability of $\mathcal{A}$'s success by $\texttt{Succ}_{\mathcal{A}}(k)$. If for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, the success probability $\texttt{Succ}_{\mathcal{A}}(k)$ is negligible, then we say that a CL-PKS scheme is existentially unforgeable against chosen message attacks.

Remark: The definition of security against a Type II adversary is as strong as Al-Riyami and Paterson's security notion for certificateless public-key encryption, where $\mathcal{A}^{\mathrm{II}}$ can requesting for private-keys of its own choices. In fact, this is also a very strong security notion, and it is unclear how realistic it is. As $\mathcal{A}^{\mathrm{II}}$ has the knowledge of the master key and hence can compute the partial private-key of any user, it gives the same degree of damage as a malicious KGC in the traditional certificate-based setting. Therefore, some authors [9] have chosen to weaken this notion and taken the security against $\mathcal{A}^{\mathrm{II}}$ as the traditional public-key cryptosystems, i.e., the private-key extraction query is not allowed to make by a Type II adversary $\mathcal{A}^{\mathrm{II}}$.

3 An Efficient CL-PKS Scheme Based on Bilinear Pairings

In this section, we propose an efficient certificateless public-key signature scheme based on bilinear pairings. Some definitions and properties of bilinear pairings are first reviewed in the following.

3.1 Preliminaries

Let $\mathcal{G}_1$ be a cyclic additive group generated by P, whose order is a prime q, and $\mathcal{G}_2$ be a cyclic multiplicative group of the same order. Let $e : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$ be a pairing which satisfies the following conditions:

1. Bilinearity: For any $P, Q, R \in \mathcal{G}_1$, we have $e(P + Q, R) = e(P, R)\ e(Q, R)$ and $e(P, Q + R) = e(P, Q)e(P, R)$. In particular, for any $a, b \in \mathbf{Z}_q^*$,

$$e(aP, bP) = e(P, P)^{ab} = e(P, abP) = e(abP, P).$$

2. Non-degeneracy: There exists $P, Q \in \mathcal{G}_1$, such that $e(P, Q) \neq 1$.

3. Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathcal{G}_1$.

We write $\mathcal{G}_1$ with an additive notation and $\mathcal{G}_2$ with a multiplicative notation, since in general implementation $\mathcal{G}_1$ will be the group of points on an elliptic curve and $\mathcal{G}_2$ will denote a multiplicative subgroup of a finite field. Typically, the map e will be derived from either the Weil or Tate pairing on an elliptic curve over a finite field. We refer readers to [5, 3] for a more comprehensive description on how these groups, pairings and other parameters should be selected for efficiency and security. Interested readers may also refer to [16] for a comprehensive bibliography of cryptographic works based on pairings.

The computational Diffie-Hellman (CDH) problem in $\mathcal{G}_1$ states that, given P, aP, bP for randomly chosen $a, b \in \mathbf{Z}_q^*$, it is computationally infeasible to compute abP.

3.2 Our Construction

The proposed certificateless public-key signature scheme comprises the following seven algorithms.

Setup:

1. On input a security parameter 1^k where $k \in \mathbb{N}$, the algorithm first generates $\langle \mathcal{G}_1, \mathcal{G}_2, e \rangle$, where $(\mathcal{G}_1, +)$ and $(\mathcal{G}_2, \cdot)$ are cyclic groups of prime order q and $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ is a bilinear pairing.

2. Arbitrarily choose a generator $P \in \mathcal{G}_1$.

3. Select a master-key $s \in_{\mathcal{R}} \mathbf{Z}_q^*$ uniformly and set $P_{pub} = sP$.

4. Choose three distinct hash functions H_1, H_2 and H_3, each of them maps from $\{0, 1\}^*$ to $\mathcal{G}_1$.

The system parameter list is $\texttt{params}=\langle \mathcal{G}_1, \mathcal{G}_2, e, q, P, P_{pub}, H_1, H_2, H_3 \rangle$. The master-key is s.

Partial-Private-Key-Extract: This algorithm takes as inputs `params`, master-key s, $ID_A \in \{0, 1\}^*$, and carries out the following for generating a partial private key D_A for a user A with identity ID_A.

1. Compute $Q_A = H_1(ID_A)$.

2. Output the partial private key $D_A = sQ_A$.

It is easy to see that D_A is actually a signature [6] on ID for the key pair (P_{pub}, s), and user A can check its correctness by checking whether $e(D_A, P) = e(Q_A, P_{pub})$.

Set-Secret-Value: This algorithm picks $x \in \mathbf{Z}_q^*$ at random and sets x as user A's secret value.

Set-Private-Key: This algorithm takes as inputs `params`, A's partial private-key D_A and A's secret value x, and outputs a pair is A's full private key $SK_A = \langle D_A, x \rangle$. So, the private key for A is just the pair consisting of the partial private key and the secret value.

Set-Public-Key: This algorithm takes as inputs `params` and A's secret value x, and generates A's public-key as $PK_A = xP$.

CL-Sign. On inputs `params`, a message $m \in \{0,1\}^*$, signer A's identity ID_A and his private key $SK_A = \langle D_A, x\rangle$, the signer randomly picks $r \in \mathbf{Z}_q^*$, computes $U = rP$ and

$$V = D_A + rH_2(m, ID_A, PK_A, U) + xH_3(m, ID_A, PK_A),$$

where $PK_A = xP$. The signature is $\sigma = (U, V)$.

CL-Verify. Given `params`, PK_A, message m, ID_A and signature $\sigma = (U, V)$, the algorithm computes $Q_A = H_1(ID_A)$ and accepts the signature if the following equation holds:

$$e(V,P) = e(Q_A, P_{pub})e(H_2(m, ID_A, PK_A, U), U)e(H_3(m, ID_A, PK_A), PK_A) \tag{1}$$

The correctness of the scheme follows from the fact that $D_A = sQ_A$ and

$$\begin{aligned} e(V,P) &= e(sQ_A, P)e(rH_2(m, ID_A, PK_A, U), P)e(xH_3(m, ID_A, PK_A), P) \\ &= e(Q_A, sP)e(H_2(m, ID_A, PK_A, U), rP)e(H_3(m, ID_A, PK_A), xP) \\ &= e(Q_A, P_{pub})e(H_2(m, ID_A, PK_A, U), U)e(H_3(m, ID_A, PK_A), PK_A). \end{aligned}$$

The current set up of our construction allows a user to create more than one public key for the same partial private key. This can be a useful property in some applications, but may be not desirable in others. In the latter case, an alternative technique of [1] can be used to generate users' key. An entity A first generate its secret value x_A and public key $PK_A = x_AP$, and Q_A is defined as $Q_A = H_1(ID_A \| PK_A)$. The partial private key is still $D_A = sQ_A$ and the private key is $SK_A = (D_A, x_A)$. In this technique, Q_A binds a user's identifier ID_A and its public keyPK_A, and thus a user can only create one public key for which he knows the corresponding private key.

4 Security Proof

Theorem 1. *In the random oracle model, our certificateless public key signature scheme is existentially unforgeable against adaptive chosen-message attacks under the assumption that the CDH problem in $\mathcal{G}_1$ is intractable.*

The theorem follows at once from Lemmas 1 and 2, according to Definition 2.

Lemma 1. *If a probabilistic polynomial-time forger $\mathcal{A}^{\mathrm{I}}$ has an advantage ε in forging a signature in an attack modelled by **Game I** of Definition 2 after running in time t and making q_{H_i} queries to random oracles H_i for $i = 1, 2, 3$, q_{ParE} queries to the partial private-key extraction oracle, q_{PK} queries to the public-key request oracle, and q_{Sig} queries to the signing oracle, then the CDH problem can be solved with probability*

$$\varepsilon' > \big(\varepsilon - (q_S(q_{H_2} + q_S) + 2)/2^k\big)/e\big(q_{ParE} + 1\big),$$

within time $t' < t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{ParE} + q_{PK} + q_{Sig})t_m + (2q_{Sig} + 1)t_{mm}$, *where* t_m *is the time to compute a scalar multiplication in* $\mathcal{G}_1$ *and* t_{mm} *is the time to perform a multi-exponentiation in* $\mathcal{G}_1$.

Proof. Let $\mathcal{A}^{\mathrm{I}}$ be a Type I adversary who can break our CL-PKS scheme. Suppose $\mathcal{A}^{\mathrm{I}}$ has a success probability ε and running time t. We show how $\mathcal{A}^{\mathrm{I}}$ can be used by a PPT algorithm $\mathcal{B}$ to solve the CDH problem in $\mathcal{G}_1$. It is interesting to note that the reductionist proof can be obtained without the requirement that $\mathcal{A}^{\mathrm{I}}$ should have submitted the secret information s_{ID} corresponding to the replaced public-key PK_{ID} when querying the signing oracle. And a tight security reduction similar to that of the ID-based signature scheme [14] can be obtained.

Let $(X = aP, Y = bP) \in \mathcal{G}_1 \times \mathcal{G}_1$ be a random instance of the CDH problem taken as input by $\mathcal{B}$. The algorithm $\mathcal{B}$ initializes $\mathcal{A}^{\mathrm{I}}$ 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 $\mathcal{A}^{\mathrm{I}}$ is making queries throughout the game. $\mathcal{B}$ responds to $\mathcal{A}^{\mathrm{I}}$'s oracle queries as follows.

Queries on Oracle H_1**:** The proof technique of Coron [10] is used to answer such queries. When an identity ID is submitted to oracle H_1, $\mathcal{B}$ first flips a coin $T \in \{0, 1\}$ that yields 0 with probability ζ and 1 with probability $1 - \zeta$, and picks $t_1 \in \mathbf{Z}_q^*$ at random. If $T = 0$, then the hash value $H_1(ID)$ is defined as $t_1 P \in \mathcal{G}_1$. If $T = 1$, then $\mathcal{B}$ returns $t_1 Y \in \mathcal{G}_1$. 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 $T = 1$, 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 $\mathcal{A}^{\mathrm{I}}$. If $D_{ID} = \perp$, $\mathcal{B}$ recovers the corresponding (ID, t_1, T) from the list L_1. Noting $T = 0$ means that $H_1(ID)$ was previously defined to be $t_1 P \in \mathcal{G}_1$ and $D_{ID} = t_1 P_{pub} = t_1 X \in \mathcal{G}_1$ is the partial private key associated to ID. Thus $\mathcal{B}$ returns D_{ID} to $\mathcal{A}^{\mathrm{I}}$ 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 $D_{ID} = t_1 P_{pub} = t_1 X$ and returns D_{ID} to $\mathcal{A}^{\mathrm{I}}$. $\mathcal{B}$ 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 $\mathcal{A}^{\mathrm{I}}$. Otherwise, $\mathcal{B}$ randomly chooses $w \in \mathbf{Z}_q^*$

and sets $PK_{ID} = wP$ and $s_{ID} = w$. $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}^{\mathrm{I}}$ 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 $w \in \mathbf{Z}_q^*$ and sets $PK_{ID} = wP$ and $s_{ID} = w$. $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}^{\mathrm{I}}$ and adds $(ID, D_{ID}, PK_{ID}, s_{ID})$ to 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 $T = 1$, 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 $\mathcal{A}^{\mathrm{I}}$.
– 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 $\mathcal{A}^{\mathrm{I}}$ 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 item $(ID, D_{ID}, PK_{ID}, s_{ID})$, $\mathcal{B}$ sets $D_{ID} = \perp$, $PK_{ID} = PK'_{ID}$, $s_{ID} = \perp$, and adds an element $(ID, D_{ID}, PK_{ID}, s_{ID})$ to L.

Queries on Oracle H_2: Suppose (m, ID, PK_{ID}, U) is submitted to oracle $H_2(\cdot)$. $\mathcal{B}$ first scans a list $L_2 = \{(m, ID, PK_{ID}, U, H_2, t_2)\}$ 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 $t_2 \in \mathbf{Z}_q^*$, and returns $H_2 = t_2 P \in \mathcal{G}_1$ as a hash value of $H_2(m, ID, PK_{ID}, U)$ to $\mathcal{A}^{\mathrm{I}}$ (we abuse the notation H_2 here), and also stores the values in the list L_2.

Queries on Oracle H_3: Suppose (m, ID, PK_{ID}) is submitted to oracle $H_3(\cdot)$. $\mathcal{B}$ first scans a list $L_3 = \{(m, ID, PK_{ID}, H_3, t_3)\}$ 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 $t_3 \in \mathbf{Z}_q^*$, and returns $H_3 = t_3 P \in \mathcal{G}_1$ as a hash value of $H_3(m, ID, PK_{ID})$ to $\mathcal{A}^{\mathrm{I}}$ (we abuse the notation H_3 here), and also stores the values in the list L_3.

Signing Oracle Queries: Suppose that $\mathcal{A}^{\mathrm{I}}$ 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}).)

Then $\mathcal{B}$ picks at random two numbers $v, u \in \mathbf{Z}_q^*$, sets $U = vP_{pub}$, and defines the hash value of $H_2(m, ID, PK_{ID}, U)$ as $H_2 = v^{-1}(uP - Q_{ID}) \in \mathcal{G}_1$

($\mathcal{B}$ halts and outputs "failure" if H_2 turns out to have already been defined for (m, ID, PK_{ID}, U)). Then $\mathcal{B}$ looks up the list L_3 for $(m, ID, PK_{ID}, H_3, t_3)$ such that the hash value of $H_3(m, ID, PK_{ID})$ has been defined to $H_3 = t_3 P$ (If such an item does not exist, $\mathcal{B}$ makes a query on oracle H_3). Finally, $\mathcal{B}$ sets $V = uP_{pub} + t_3 PK_{ID}$. Now (U, V) is returned to $\mathcal{A}^{\mathrm{I}}$, which appears to be a valid signature since

$$\begin{aligned}
& e(Q_{ID}, P_{pub}) e(H_2, U) e(H_3, PK_{ID}) \\
&= e(Q_{ID}, P_{pub}) e(v^{-1}(uP - Q_{ID}), vP_{pub}) e(t_3 P, PK_{ID}) \\
&= e(Q_{ID}, P_{pub}) e(uP - Q_{ID}, P_{pub}) e(P, t_3 PK_{ID}) \\
&= e(Q_{ID}, P_{pub}) e(P, uP_{pub}) e(Q_{ID}, P_{pub})^{-1} e(P, t_3 PK_{ID}) \\
&= e(P, uP_{pub} + t_3 PK_{ID}) = e(V, P).
\end{aligned}$$

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.

Eventually, $\mathcal{A}^{\mathrm{I}}$ outputs a forgery $\tilde{\sigma} = (\tilde{U}, \tilde{V})$ on a message $\tilde{m}$, for an identity $\tilde{ID}$ with public key $PK_{\tilde{ID}}$..[1] Now $\mathcal{B}$ recovers the triple $(\tilde{ID}, \tilde{t}_3, \tilde{T})$ from L_1. If $\tilde{T} = 0$, then $\mathcal{B}$ outputs "failure" and stops.. Otherwise, it goes on and finds out an item $(\tilde{m}, \tilde{ID}, PK_{\tilde{ID}}, \tilde{U}, \tilde{H}_2, \tilde{t}_2)$ in the list L_2, and an item $(\tilde{m}, \tilde{ID}, PK_{\tilde{ID}}, \tilde{H}_3, \tilde{t}_3)$ in the list L_3. Note that the list L_2 and L_3 must contain such entries with overwhelming probability (otherwise, $\mathcal{B}$ stops and outputs "failure"). Note that $\tilde{H}_2 = H_2(\tilde{m}, \tilde{ID}, PK_{\tilde{ID}}, \tilde{U})$ is $\tilde{t}_2 P \in \mathcal{G}_1$, and $\tilde{H}_3 = H_3(\tilde{m}, \tilde{ID}, PK_{\tilde{ID}})$ is $\tilde{t}_3 P \in \mathcal{G}_1$. If $\mathcal{A}^{\mathrm{I}}$ succeeds in the game, then

$$e(\tilde{V}, P) = e(Q_{\tilde{ID}}, X) e(\tilde{H}_2, \tilde{U}) e(\tilde{H}_3, PK_{\tilde{ID}})$$

with $\tilde{H}_2 = \tilde{t}_2 P$, $\tilde{H}_3 = \tilde{t}_3 P$ and $Q_{\tilde{ID}} = \tilde{t}_1 Y$ for known elements $\tilde{t}_1$, $\tilde{t}_2$, $\tilde{t}_3 \in \mathbf{Z}_q^*$. Therefore,

$$e(\tilde{V} - \tilde{t}_2 \tilde{U} - \tilde{t}_3 \tilde{PK}_{ID}, P) = e(\tilde{t}_1 Y, X),$$

and thus $\tilde{t}_1^{-1}(\tilde{V} - \tilde{t}_2 \tilde{U} - \tilde{t}_3 PK_{\tilde{ID}})$ is the solution to the target CDH instance $(X, Y) \in \mathcal{G}_1 \times \mathcal{G}_1$.

Now we evaluate $\mathcal{B}$'s probability of failure.. $\mathcal{B}$'s simulation of oracle H_3 is perfect. One can also readily check that the probability of failure in handling a signing query because of a conflict on H_2 is at most $q_S(q_{H_2} + q_S)/2^k$, as L_2 never has more than $q_{H_2} + q_S$ entries, while the probability for $\mathcal{A}^{\mathrm{I}}$ to output a valid forgery $\tilde{\sigma}$ on a message $\tilde{m}$ for an identity $\tilde{ID}$ with public key $PK_{\tilde{ID}}$, without asking the corresponding $H_2(\tilde{m}, \tilde{ID}, PK_{\tilde{ID}}, \tilde{U})$ query or $H_3(\tilde{m}, \tilde{ID}, PK_{\tilde{ID}})$ query, is at most $2/2^k$. And, by an analysis similar to Coron's technique [10], the probability $\zeta^{q_{\mathtt{ParE}}}(1 - \zeta)$ for $\mathcal{B}$ not to fail in key extraction queries or because $\mathcal{A}^{\mathrm{I}}$ produces its forgery on a 'bad' identity $\tilde{ID}$ is greater than $1 - 1/e(q_{\mathtt{ParE}} + 1)$ when the optimal probability $\zeta_{\mathrm{opt}} = q_{\mathtt{ParE}}/(q_{\mathtt{ParE}} + 1)$ is taken. Therefore,

[1] We remark again that the Challenger $\mathcal{B}$ may not know the secret value $s_{\tilde{ID}}$ corresponding to $PK_{\tilde{ID}}$, while a reduction can be given even if $\mathcal{B}$ does not know $s_{\tilde{ID}}$.

it results that $\mathcal{B}$'s advantage in solving the CDH problem in $\mathcal{G}_1$ is at least $\big(\varepsilon - (q_S(q_{H_2} + q_S) + 2)/2^k\big)/e\big(q_{\mathtt{ParE}} + 1\big)$.

Lemma 2. *If a PPT forger $\mathcal{A}^{\mathrm{II}}$ has an advantage ε in forging a signature in an attack modelled by* ***Game II*** *of Definition 2 after running in time t and making q_{H_i} queries to random oracles H_i for $i = 2, 3$, q_E queries to the private-key extraction oracle, q_{PK} queries to the public-key request oracle, and q_{Sig} queries to the signing oracle, then the CDH problem can be solved with probability*

$$\varepsilon' > \big(\varepsilon - (q_S(q_{H_2} + q_S) + 2)/2^k\big)/e\big(q_E + 1\big),$$

within time $t' < t + (q_{H_2} + q_{H_3} + q_{PK} + q_{Sig})t_m + (2q_{Sig} + 1)t_{mm}$, where t_m is the time to compute a scalar multiplication in $\mathcal{G}_1$ and t_{mm} is the time to perform a multi-exponentiation in $\mathcal{G}_1$.

Proof. Suppose $\mathcal{A}^{\mathrm{II}}$ 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 $\mathcal{G}_1$ with probability at least ε'. Let $(X = aP, Y = bP) \in \mathcal{G}_1 \times \mathcal{G}_1$ be a random instance of the CDH problem taken as input by $\mathcal{B}$.

$\mathcal{B}$ randomly chooses $s \in \mathbf{Z}_q^*$ as the master key, and then initializes $\mathcal{A}^{\mathrm{II}}$ with $P_{pub} = sP$ and also the master key s. The adversary $\mathcal{A}^{\mathrm{II}}$ then starts making oracle queries such as those described in Definition 2. Note that the partial private key $D_{ID} = sH_1(ID)$ can be computed by both $\mathcal{B}$ and $\mathcal{A}^{\mathrm{II}}$, thus the hash function $H_1(\cdot)$ is not modelled 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 $\mathcal{A}^{\mathrm{II}}$ makes certain queries specified below.

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

- If the list L contains (ID, PK_{ID}, s_{ID}, T), $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}^{\mathrm{II}}$.
- If the list L does not contain (ID, PK_{ID}, s_{ID}), as in Coron's proof [10], $\mathcal{B}$ flips a coin $T \in \{0, 1\}$ that yields 0 with probability ζ and 1 with probability $1 - \zeta$. $\mathcal{B}$ also picks a number $w \in \mathbf{Z}_q^*$ at random. If $T = 0$, the value of PK_{ID} is defined as $wP \in \mathcal{G}_1$. If $T = 1$, $\mathcal{B}$ returns $wY \in \mathcal{G}_1$. In both cases, $\mathcal{B}$ sets $s_{ID} = w$, and inserts a tuple (ID, PK_{ID}, s_{ID}, T) into a list $L_1 = \{(ID, PK_{ID}, s_{ID}, T)\}$ to keep track the way it answered the queries. $\mathcal{B}$ returns PK_{ID} to $\mathcal{A}^{\mathrm{II}}$.

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

Queries on Oracle H_2: When a tuple (m, ID, PK_{ID}, U) is submitted to oracle $H_2(\cdot)$, $\mathcal{B}$ first scans a list $L_2 = \{(m, ID, PK_{ID}, U, H_2, t_2)\}$ to check whether H_2

has already been defined for that input. If so, the existing value is returned. Otherwise, $\mathcal{B}$ picks a random number $t_2 \in \mathbf{Z}_q^*$, and returns $H_2 = t_2 P \in \mathcal{G}_1$ as the hash value of $H_2(m, ID, PK_{ID}, U)$ to $\mathcal{A}^{\mathrm{II}}$, and also stores the values in the list L_2.

Queries on Oracle H_3**:** When a tuple (m, ID, PK_{ID}) is submitted to oracle $H_3(\cdot)$, $\mathcal{B}$ first scans a list $L_3 = \{(m, ID, PK_{ID}, H_3, t_3)\}$ 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 $t_3 \in \mathbf{Z}_q^*$, and returns $H_3 = t_3 Y \in \mathcal{G}_1$ as a hash value of $H_3(m, ID, PK_{ID})$ to $\mathcal{A}^{\mathrm{II}}$, and also stores the values in the list L_3.

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

First, $\mathcal{B}$ picks $v, u \in \mathbf{Z}_q^*$ at random, sets $U = uPK_{ID}$, $V = vPK_{ID} + D_{ID}$ and defines the hash value of $H_2(ID, M, PK_{ID}, U)$ as $H_2 = u^{-1}(vP - H_3)$, where $H_3 = H_3(m, ID, PK_{ID})$ ($\mathcal{B}$ halts and outputs "failure" if H_2 turns out to have already been defined for (m, ID, PK_{ID}, U)). Now (U, V) is returned to $\mathcal{A}^{\mathrm{II}}$, which appears to be a valid signature since

$$\begin{aligned}
& e(Q_{ID}, P_{pub}) e(H_2, U) e(H_3, PK_{ID}) \\
&= e(Q_{ID}, P_{pub}) e(u^{-1}(vP - H_3), uPK_{ID}) e(H_3, PK_{ID}) \\
&= e(sQ_{ID}, P) e(vP, PK_{ID}) e(-H_3, PK_{ID}) e(H_3, PK_{ID}) \\
&= e(D_{ID}, P) e(vPK_{ID}, P) \\
&= e(vPK_{ID} + D_{ID}, P) = e(V, P).
\end{aligned}$$

Eventually, $\mathcal{A}^{\mathrm{II}}$ outputs a forgery $\tilde{\sigma} = (\tilde{U}, \tilde{V})$ on a message $\tilde{m}$, for an identity $I\tilde{D}$ with public key $PK_{I\tilde{D}}$. Then $\mathcal{B}$ recovers $(I\tilde{D}, PK_{I\tilde{D}}, s_{I\tilde{D}}, \tilde{T})$ from L_1 and evaluates $\tilde{T}$. If $\tilde{T} = 0$, then $\mathcal{B}$ outputs "failure" and stops. Otherwise, it looks up an item $(\tilde{m}, I\tilde{D}, PK_{I\tilde{D}}, \tilde{U}, \tilde{H}_2, \tilde{t}_2)$ in the list L_2 such that the value of $\tilde{H}_2 = H_2(\tilde{m}, I\tilde{D}, PK_{I\tilde{D}}, \tilde{U})$ has been defined to be $\tilde{t}_2 P$. $\mathcal{B}$ also looks up an item $(\tilde{m}, I\tilde{D}, PK_{I\tilde{D}}, \tilde{H}_3, \tilde{t}_3)$ in the list L_3 such that the value of $\tilde{H}_3 = H_3(\tilde{m}, I\tilde{D}, PK_{I\tilde{D}})$ has been defined to be $\tilde{t}_3 Y$. Note that the lists L_2 and L_3 must contain such entries with overwhelming probability. If $\mathcal{A}^{\mathrm{II}}$ succeeds in the game, then

$$e(\tilde{V}, P) = e(Q_{I\tilde{D}}, P_{pub}) e(\tilde{H}_2, \tilde{U}) e(\tilde{H}_3, PK_{I\tilde{D}})$$

with $\tilde{H}_2 = \tilde{t}_2 P$, $\tilde{H}_3 = \tilde{t}_3 Y$, $P_{pub} = sP$ and $PK_{I\tilde{D}} = s_{I\tilde{D}} X$, for known elements $\tilde{t}_2, \tilde{t}_3, s, s_{I\tilde{D}} \in \mathbf{Z}_q^*$. Therefore,

$$e(\tilde{V} - sQ_{I\tilde{D}} - \tilde{t}_2 \tilde{U}, P) = e(\tilde{t}_3 Y, s_{I\tilde{D}} X),$$

and thus $(s_{I\tilde{D}} \tilde{t}_3)^{-1}(\tilde{V} - sQ_{I\tilde{D}} - \tilde{t}_2 \tilde{U})$ is the solution to the CDH instance (X, Y).

Now we evaluate the failure probability of $\mathcal{B}$. Our simulation for oracle H_3 is perfect. Also, the probability for $\mathcal{B}$ to fail in handling a signing query because of a conflict on H_2 is at most $q_S(q_{H_2} + q_S)/2^k$. The probability for $\mathcal{A}^{\mathrm{II}}$ to output

a valid forgery $\tilde{\sigma}$ on a message $\tilde{m}$ for identity $\tilde{ID}$ with public key $\tilde{PK}_{ID}$, without asking the corresponding $H_2(\tilde{m}, \tilde{ID}, \tilde{PK}_{ID}, \tilde{U})$ query or $H_3(\tilde{m}, \tilde{ID}, \tilde{PK}_{ID})$ query, is at most $2/2^k$. And, by an analysis similar to Coron's [10], we can see that the probability $\zeta^{q_E}(1-\zeta)$ for $\mathcal{B}$ not to fail in a private key extraction query or because $\mathcal{A}^{\mathrm{II}}$ produces its forgery on a 'bad' identity $\tilde{ID}$ is greater than $1 - 1/e(q_E + 1)$ when the optimal probability $\zeta_{\mathrm{opt}} = q_E/(q_E + 1)$ is taken. Hence, $\mathcal{B}$'s advantage in solving the CDH problem in $\mathcal{G}_1$ is at least $\big(\varepsilon - (q_S(q_{H_2} + q_S) + 2)/2^k\big)/e\big(q_E + 1\big)$.

5 Conclusion

Al-Riyami and Paterson introduced the new paradigm of certificateless public key cryptography in 2003. They established a security model for certificateless public key encryption and proposed some efficient constructions. In this paper, we proposed a security model for certificateless public-key signature, and an efficient construction based on bilinear pairings. We also showed that the proposed scheme is tightly equivalent to the computational Diffie-Hellman problem in the random oracle model.

Acknowledgements

The work is supported by National Natural Science Foundation of China under Granted No. 60373039, 90604018, and National Grand Fundamental Research Project of China under Granted No.G1999035802. The second author was supported by a grant from CityU (Project No. 7001844).

References

1. S. Al-Riyami and K. Paterson, Certificateless public key cryptography, Advances in Cryptology-Asiacrypt'2003, *Lecture Notes in Computer Science*, vol. 2894, pages 452-473, Springer-Verlag, 2003.
2. S. Al-Riyami and K. Paterson, "CBE from CL-PKE: A generic construction and efficient schemes", Public Key Cryptography-PKC'05, *Lecture Notes in Computer Science*, vol. 3386, pages 398-415, Springer-Verlag, 2005.
3. P. Barreto, H. Kim, B. Lynn and M. Scott, Efficient algorithms for pairing-based cryptosystems, Advances in Cryptology-Crypto'2002, *Lecture Notes in Computer Science*, vol. 2442, pages 354-368, Springer-Verlag, 2002.
4. K. Bentahar, P. Farshim, J. Malone-Lee, and N.P. Smart, Generic constructions of identity-based and certificateless KEMs. IACR *Cryptology ePrint Archive*, Report 2005/058, 2005.
5. D. Boneh and F. Franklin, Identity-based encryption from the Weil pairing, Advances in Cryptology-Crypto'2001, *Lecture Notes in Computer Science*, vol. 2139, pages 213-229, Springer-Verlag, 2001; *SIAM J. COMPUT.*, 32(3): 586-615, 2003.
6. D. Boneh, B. Lynn and H. Shacham, Short signatures from the Weil pairing, Advances in Cryptology-Asiacrypt'2001, *Lecture Notes in Computer Science*, vol. 2248, pages 514-532, Springer-Verlag; *J. Cryptology*, 17(4): 297-319, 2004.

7. J. Baek, R. Safavi-Naini and W. Susilo, Certificateless public key encryption without pairing, Proc. of the 8th Information Security Conference (ISC 2005), *Lecture Notes in Computer Science*, vol. 3650, pages 134-148, Springer-Verlag, 2005.
8. L. Chen, K. Harrison, N. P. Smart, and D. Soldera, Applications of multiple trust authorities in pairing based cryptosystems, InfraSec 2002, *Lecture Notes in Computer Science*, vol. 2437, pages 260-275, Springer-Verlag, 2002.
9. Z.H. Cheng and R. Comley. Efficient certificateless public key encryption, IACR *Cryptology ePrint Archive*, Report 2005/012, 2005
10. J.S. Coron. On the exact security of Full Domain Hash. Advances in Cryptology-Crypto'00, *Lecture Notes in Computer Science*, vol.1880, pages 229-235, Springer-Verlag, 2000.
11. A.W. Dent and C. Kudla, On proofs of security for certificateless cryptosystems, IACR *Cryptology ePrint Archive*, Report 2005/348, 2005
12. C. Gentry, Certificate-based encryption and the certificate revocation problem, Advances in Cryptology-Eurocrypt'2003, *Lecture Notes in Computer Science*, vol. 2656, pages 272-293, Springer-Verlag, 2000.
13. X.Y. Huang, W. Susilo, Y.. Mu and F.T. Zhang, On the security of a certificateless signature scheme. Cryptology and Network Security: 4th International Conference, *Lecture Notes in Computer Science*, vol.3810, pages 13-25, Springer-Verlag, 2005
14. B. Libert and J.J. Quisquater. The exact security of an identity based signature and its applications, IACR *Cryptology ePrint Archive*, Report 2004/102, 2004.
15. B. Libert and J.J. Quisquater, What is possible with identity based cryptography for PKIs and what still must be improved, EuroPKI 2004, *Lecture Notes in Computer Science*, vol. 3093, pages 57-70, Springer-Verlag, 2004.
16. The pairing-Based Crypto Lounge.. Web page maintained by Paulo Barreto. Available at: http://planeta.terra.com.br/informatica/paulobarreto/pblounge.html
17. A.Shamir, Identity based cryptosystems and signature schemes, Advances in Cryptology-Crypto'84, *Lecture Notes in Computer Science*, vol.196, pages 47-53, Springer-Verlag, 1984.
18. D.H. Yum and P.J. Lee, Generic construction of certificateless signature. Proc. of Information Security and Privacy: 9th Australasian Conference, ACISP 2004, *Lecture Notes in Computer Science*, vol.3108, pages 200-211, Springer-Verlag, 2004