Efficient Certificateless Signature Schemes

Kyu Young Choi, Jong Hwan Park, Jung Yeon Hwang, and Dong Hoon Lee

Center for Information Security Technologies(CIST),
Korea University, Seoul, Korea
{young,decartian,videmot}@cist.korea.ac.kr
donghlee@korea.ac.kr

Abstract. Recently, in order to eliminate the use of certificates in certified public key cryptography and the key-escrow problem in identity based cryptography, the notion of certificateless public key cryptography was introduced. In this paper, to construct an efficient certificateless signature (CLS) scheme, we present a new approach compactly and orthogonally combining short signatures using bilinear maps. Our approach is conceptually simple but effective to improve efficiency greatly. In the proposed CLS scheme a full private key of a user is a single group element and signature verification requires only one pairing operation. In addition, our CLS scheme has a flexible structure which can be easily extended to a certificateless signature scheme with additional properties such as certificateless ring and blind signature schemes.

1 Introduction

In a traditional public key cryptography (PKC), a random public key of a user is associated with the user by a certificate, that is, a signature of trusted Certificate Authority (CA) on the public key. Inevitably this feature causes CA to require a large amount of storage and computing time managing the certificates [8]. To simplify the certificate management process, Shamir introduced the concept of identity based cryptography (ID-PKC) where the certificate of a random public key does not be needed any more since publicly known information such as e-mail address is used as user's public key [15]. However, an inherent problem of ID-PKC is that a Key Generation Center (KGC) generates any user's private key using a master-key of KGC. Obviously a malicious KGC is able to forge the signature of any signer. This is called "key escrow" problem. In 2003, Al-Riyami and Paterson introduced the concept of certificateless public key cryptography (CL-PKC) which eliminates the use of certificates in PKC and solve the key escrow problem in ID-PKC [1]. The basic idea of CL-PKC is to construct a public/private key pair for a user by combining a master key of KGC with a random secret value generated by the user. In this paper we concentrate on a certificateless signature (CLS) scheme.

Despite of the usefulness of a CLS scheme, it is not easy to construct a secure and efficient CLS scheme because the construction of a CLS scheme conceptually involves mechanisms to authenticate an identity of a user, the public key of the user, and a message to be signed at the same time. For the security model of a CLS scheme reflecting such authentication mechanisms, unlike that of an ordinary signature scheme, we should consider two types of forgers, Type I and Type II forgers : A Type I forger represents a normal third party attacker who has no access to the master key but is allowed to replace public keys of users. A Type II forger represents a malicious KGC who is equipped with the master key but is not able to replace public keys. However, although many researches on a CLS scheme [1,17,12,9,10,20,11,6] are performed, only few schemes [10,20] are known to be secure against these forgers.

Naturally, such complicated authentication mechanisms should be a critical consideration to design an efficient CLS scheme. To improve efficiency it would be desirable to integrate the functionalities of authentication imbedded in a CLS scheme compactly while maintaining security.

Our Results. In the paper, to construct an efficient CLS scheme, we first present novel combinations of short signature schemes using a bilinear group: In the key setup phase, the Boneh-Shacham-Lynn short signature [5] or Boneh-Boyen's short signature [2] is used for KGC to generate a signature, that is, a partial private key corresponding to an identity of a user. The Boneh-Boyen short signature [2] is used for the user to generate a full private key, which plays a crucial role of a private *self-certificate* on the public-key of his/her choice, using the previous partial private key. In our scheme, signature verification requires only one pairing operation, compared to at least four pairing operations in the previous works [10,12,20], and signature generation requires no pairing operation. Moreover, a full private key for a user, which is computed by applying two short signature schemes sequentially, is just a single group element.

The compact feature of a full private key of a user, which aggregates two signatures, provides the minimum loss against key exposure. In other words, even if an adversary obtains a full private key, he cannot extract the partial private key from the full private key which is a signature on the partial private key. Because the partial private key is used as a long-lived key this provides a proactive property such that the user's public key can be replaced periodically.

We show that our CLS schemes are provably secure in the random oracle model. For security model we consider a realistic model where a Type I forger is not allowed to obtain a valid signature for the public key replaced by the forger. In practical environments, it is too strong to assume that the signer knows the private key associated with the replaced public key (by others). This model was already developed in several recent works [11,20].

Finally our method provides flexibility for extending to CLS schemes with additional properties such as certificateless blind and ring signature schemes. In fact, applying a similar method in [18,19,7] to our schemes we can directly construct certificateless ring and blind signature schemes.

Related Works. The first CLS scheme was proposed by Al-Riyami and Paterson [1]. Unfortunately, it was found insecure against a Type I forger by Huang et al. [10]. They also proposed a CLS scheme and proved its security in the random oracle model. In [17], Yum and Lee proposed a generic construction of CLS. However, Hu et al. presented that their construction is insecure against a Type I forger and improved it. Gorantla and Saxena [9] proposed an efficient CLS scheme. However, it was also found insecure against a Type I forger by Cao et al. [6]. In [12], Li et al. proposed a CLS scheme. It seems to be secure but a security analysis for the scheme was not formalized. Recently, Zhang et al. [20] proposed a CLS scheme and showed its security in the random oracle model.

Organization. The rest of this paper is organized as follows. In Section 2, we describe some fundamental backgrounds and define our security model for a CLS scheme. In Section 3 we propose an efficient CLS scheme and its security proofs. In Section 4 we propose a CLS scheme with a pairing operation and its security proofs. In Section 5, we analyze the performances of the proposed CLS schemes. In Section 6, we present extension of our CLS schemes. We conclude the paper in Section 7.

2 Preliminaries

We review some fundamental backgrounds required in this paper, namely bilinear pairing, certificateless signature scheme.

2.1 Bilinear Pairings and Some Problems

Let $\mathbb{G}_1$ be a cyclic additive group of prime order q and $\mathbb{G}_2$ be a cyclic multiplicative group of same order q. We assume that the discrete logarithm problems (DLP) in both $\mathbb{G}_1$ and $\mathbb{G}_2$ are intractable.

Admissible Bilinear Map. We call $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ an *admissible bilinear* map if it satisfies the following properties:

- Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$.
- Non-degenerancy: There exists $P \in \mathbb{G}_1$ such that $e(P, P) \neq 1$.
- Computability: There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

The modified Weil and Tate pairings in elliptic curve are examples of the admissible bilinear maps. We consider following problems in the group $\mathbb{G}_1$.

Computational Diffie-Hellman (CDH) problem: The CDH problem is to compute abP when given P, aP and bP for some $a, b \in \mathbb{Z}_q^*$.

Inverse Computational Diffie-Hellman (ICDH) problem: The ICDH problem is to compute $a^{-1}P$ when given P and aP for some $a \in \mathbb{Z}_q^*$.

Modified Inverse Computational Diffie-Hellman (mICDH) problem: The mICDH problem is to compute $(a + b)^{-1}P$ when given b, P and aP for some $a, b \in \mathbb{Z}_q^*$.

The CDH, ICDH and mICDH problems are polynomial time equivalent [16]. We assume that the CDH, ICDH and mICDH problems in $\mathbb{G}_1$ are intractable. That is, there is no polynomial time algorithm solving these problems with non-negligible probability.

2.2 Certificateless Signature Scheme

We briefly recall a formal definition of a certificateless signature scheme [10]. The CLS scheme is specified by seven polynomial time algorithms.

Setup: This algorithm takes a security parameter k as input and returns the system parameters `params` and a secret master key `master-key`.

Partial-Private-Key-Extract: This algorithm takes `params`, `master-key` and a user's identity ID as input. It returns a partial private key D_{ID} corresponding to the user.

Set-Secret-Value: This algorithm takes the security parameter k and a user's identity ID as input. It returns the user's secret value x_{ID}.

Set-Public-Key: This algorithm takes a user's secret value x_{ID} as input. It returns the user's public key PK_{ID}.

Set-Private-Key: This algorithm takes a user's partial private key D_{ID} and public key PK_{ID}, and his secret value x_{ID} as input. It returns the user's full private key SK_{ID}.

Sign: This algorithm takes `params`, a message m, and a user's full private key SK_{ID} as input. It returns a signature σ.

Verify: This algorithm takes `params`, a message m, a user's identity ID, a public key PK_{ID}, and a signature σ as input. It returns 0 or 1. With output value 1, we say that σ is a valid signature of a message m.

The **Setup** and **Partial-Private-Key-Extract** algorithms are performed by a Key Generation Center (KGC). Once a partial private key is given to a user via secure channel, the user runs the **Set-Secret-Value** algorithm and chooses a secret value to generate its own public/private key pair.

The security model of CLS is different from that of a normal signature scheme. As defined in [1,10,20], we should consider two types of forger for a CLS scheme, a Type I forger $\mathcal{F}_I$ and a Type II forger $\mathcal{F}_{II}$. The forger $\mathcal{F}_I$ represents a normal third party attacker against the CLS scheme. That is, $\mathcal{F}_I$ is not allowed to access to the master-key but $\mathcal{F}_I$ may request public keys and replace public keys with values of its choice. The forger $\mathcal{F}_{II}$ represents a malicious KGC who generates partial private key of users. The forger $\mathcal{F}_{II}$ is allowed to have access to the master-key but not replace a public key. We consider two games against the Type I and Type II forgers as follows.

Game I. The first game is performed between a challenger $\mathcal{C}$ and the Type I forger $\mathcal{F}_I$ for a certificateless signature scheme Π as follows.

- **Initialization:** $\mathcal{C}$ runs **Setup** algorithm and generates a master secret key `master-key`, public system parameters `params`. $\mathcal{C}$ keeps `master-key` secret and then gives `params` to $\mathcal{F}_I$. Note that $\mathcal{F}_I$ does not know the master key `master-key`.
- **Queries:** $\mathcal{F}_I$ may adaptively issue the following queries to $\mathcal{C}$.
  - ExtrPartSK(ID): When $\mathcal{F}_I$ requests the partial private key for a user with identity ID, $\mathcal{C}$ responds the user's partial private key D_{ID} running **Partial-Private-Key-Extract** algorithm.
  - ExtrFullSK(ID): When $\mathcal{F}_I$ requests the full private key for a user with identity ID, $\mathcal{C}$ responds the user's full private key SK_{ID} running **Partial-Private-Key-Extract**, **Set-Secret-Value** and **Set-Private-Key** algorithms.
  - ReqestPK(ID): When $\mathcal{F}_I$ requests the public key for a user with identity ID, the challenger $\mathcal{C}$ responds the user's public key PK_{ID} running **Set-Secret-Value** and **Set-Public-Key** algorithms.
  - RepalcePK(ID): $\mathcal{F}_I$ can replace the original public key PK_{ID} to a new public key PK'_{ID} chosen by him.
  - SIGN(m, ID): When $\mathcal{F}_I$ requests a signature on a message m for a user with identity ID, the challenger $\mathcal{C}$ responds a valid signature σ for m running **Sign** algorithm with the matching public key PK_{ID} for ID. If the public key PK_{ID} has been replaced earlier by $\mathcal{F}_I$, then $\mathcal{C}$ cannot know the corresponding private key SK_{ID} and thus the signing oracle's answer may not be correct. In such case, to correctness of the signing oracle's answer, we assume that $\mathcal{F}_I$ additionally submits the corresponding secret information to the signing oracle.
- **Output:** Eventually, $\mathcal{F}_I$ outputs (ID_t, m_t, σ_t), where ID_t is the identity of a target user, m_t is a message, and σ_t is a signature for m_t. $\mathcal{F}_I$ wins the game if
  1. ExtrPartSK(ID_t), ExtrFullSK(ID_t), and SIGN(m_t, ID_t) queries have never been queried.
  2. **Verify**(`params`, $m_t, ID_t, PK_t, \sigma_t$) outputs 1, that is, the signature σ_t for a message m_t is valid under PK_t which may be replaced by $\mathcal{F}_I$.

We define $\mathsf{Succ}^{\Pi}_{\mathcal{F}_I}$ to be the success probability that $\mathcal{F}_I$ wins in the above game.

Game II. The second game is performed between a challenger $\mathcal{C}$ and the Type II forger $\mathcal{F}_{II}$ for a certificateless signature scheme Π as follows.

- **Initialization:** $\mathcal{C}$ runs **Setup** algorithm and generates a master secret key `master-key`, public system parameters `params`. The challenger $\mathcal{C}$ gives public `params` and secret `master-key` to $\mathcal{F}_{II}$.
- **Queries:** $\mathcal{F}_{II}$ may adaptively issue the following queries to $\mathcal{C}$.
  - ExtrFullSK(ID): When $\mathcal{F}_{II}$ requests the full private key for a user with identity ID, $\mathcal{C}$ responds the user's full private key SK_{ID} running **Partial-Private-Key-Extract**, **Set-Secret-Value** and **Set-Private-Key** algorithms.

  - $\mathsf{RequestPK}(ID)$: When $\mathcal{F}_{II}$ requests the public key for a user with identity ID, the challenger $\mathcal{C}$ responds the user's public key PK_{ID} running **Set-Secret-Value** and **Set-Public-Key** algorithms.
  - $\mathsf{SIGN}(m, ID)$: When $\mathcal{F}_{II}$ requests a signature on a message m for a user with identity ID, the challenger $\mathcal{C}$ responds a valid signature σ for m running **Sign** algorithm with matching public key PK_{ID} for ID.
- **Output:** Eventually, $\mathcal{F}_{II}$ outputs (ID_t, m_t, σ_t), where ID_t is the identity of a target user, m_t is a message, and σ_t is a signature for m_t. $\mathcal{F}_{II}$ wins the game if
  1. $\mathsf{ExtrFullSK}(ID_t)$ and $\mathsf{SIGN}(m_t, ID_t)$ queries have never been issued.
  2. **Verify**$(\texttt{params}, m_t, ID_t, \sigma_t)$ outputs 1, that is, the signature σ_t for a message m_t is valid under PK_t.

We define $\mathsf{Succ}^{\Pi}_{\mathcal{F}_{II}}$ to be the success probability that $\mathcal{F}_{II}$ wins in the above game. Note that $\mathcal{F}_{II}$ does not need additional extraction query to obtain partial private keys since the master key `master-key` is given to $\mathcal{F}_{II}$.

Definition 1. *We say that a certificateless signature scheme Π is existentially unforgeable against chosen message attacks, if for any polynomially bounded forgers $\mathcal{F}_I$ and $\mathcal{F}_{II}$, the success probabilities of both $\mathcal{F}_I$ and $\mathcal{F}_{II}$ are negligible. In other words,*

$$\mathsf{Succ}^{\Pi}_{\mathcal{F}_I}(k) < \epsilon \quad \text{and} \quad \mathsf{Succ}^{\Pi}_{\mathcal{F}_{II}}(k) < \epsilon$$

where k is the security parameter.

3 New Certificateless Signature Scheme

In this section, we propose a new efficient CLS scheme and prove the security of the proposed scheme. We denote this CLS scheme by eCLS.

3.1 Our Construction

Setup. To generate system parameters and master key, run as follows:
1. Generate $(\mathbb{G}_1, \mathbb{G}_2, e)$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order q and e is an admissible bilinear map.
2. Choose a random $s \in \mathbb{Z}_q^*$ and a generator P of $\mathbb{G}_1$. Compute $P_{pub} = sP$.
3. Choose three cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \mathbb{G}_1 \to \mathbb{Z}_q^*$, and $H_3 : \{0,1\}^* \to \mathbb{Z}_q^*$.

Return the private `master-key`$= s$ and the system parameters `params`$= \{e, \mathbb{G}_1, \mathbb{G}_2, q, P, P_{pub}, H_1, H_2, H_3\}$. We assume that `params` is available to all users.

Partial-Private-Key-Extract. On input `params`, `master-key`, and identity ID_A of user A. Compute $Q_A = H_1(ID_A)$ and return a partial private key $D_A = sQ_A$ for user A.

Set-Secret-Value. On input k and ID_A, choose a random value $x_A \in \mathbb{Z}_q^*$ and return x_A as A's secret value.

Set-Public-Key. On input `params` and x_A, compute $R_A = x_A P$ and return the public key $PK_A = R_A$.

Set-Private-Key. On input x_A, R_A, and D_A. Compute $y_A = H_2(R_A)$ and $S_A = \frac{1}{x_A + y_A} D_A$. Return the (full) private key $SK_A = S_A$.

Sign. On input `params`, ID_A, S_A, and a message m, perform the following steps:

1. Choose a random $r \in \mathbb{Z}_q^*$.
2. Compute $U = rQ_A = rH_1(ID_A)$.
3. Set $h = H_3(m, U)$.
4. Compute $V = (r + h)S_A$.
5. Return $\sigma = (U, V)$ as the signature on the message m.

Verify. On input `params`, ID_A, R_A, m, and $\sigma = (U, V)$. Compute $Q_A = H_1(ID_A)$, $y_A = H_2(R_A)$, and $h = H_3(m, U)$. Check if $e(V, R_A + y_A P) = e(U + hQ_A, P_{pub})$ holds. If the equation holds, it outputs 1, otherwise 0.

We can easily show that our CLS scheme satisfies completeness property as follows:

$$\begin{aligned} e(V, R_A + y_A P) &= e((r+h)S_A, x_A P + y_A P) \\ &= e((r+h)(x_A + y_A)^{-1} sQ_A, (x_A + y_A)P) \\ &= e((r+h)sQ_A, P) \\ &= e((rQ_A + hQ_A, sP) = e(U + hQ_A, P_{pub}). \end{aligned}$$

3.2 Security Analysis

Theorem 1. *Our certificateless signature scheme* eCLS *is existentially unforgeable against a Type I forger in random oracle model under the CDH assumption.*

Proof. Suppose there exists a forger $\mathcal{F}_I$ which has advantage in attacking our CLS scheme eCLS. We want to build an algorithm $\mathcal{C}$ that uses $\mathcal{F}_I$ to solve the CDH problem. $\mathcal{C}$ receives a CDH instance (P, aP, bP) for randomly chosen $a, b \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$. Its goal is to compute abP. $\mathcal{C}$ runs $\mathcal{F}_I$ as a subroutine and simulates its attack environment. $\mathcal{C}$ sets $P_{pub} = aP$ where a is the master key, which is unknown to $\mathcal{C}$, and gives system parameters to $\mathcal{F}_I$. Without loss of generality, we assume that any extraction (ExtrPartSK, RequestPK, ExtrFullSK) and signature (SIGN) queries are preceded by H_1 query, and the SIGN and ExtrFullSK queries are preceded by RequestPK query. To avoid collision and consistently respond to these queries, $\mathcal{C}$ maintains four lists $L_{H_1}, L_{H_2}, L_{H_3}, L_K = \{\langle ID, PK_{ID}, x_{ID}, c(= 0 \text{ or } 1)\rangle\}$ which are initially empty. $\mathcal{C}$ then simulates the oracle queries of $\mathcal{F}_I$ as follows:

- H_1 query: Suppose $\mathcal{F}_I$ makes at most q_{H_1} queries to H_1 oracle. First, $\mathcal{C}$ chooses $j \in [1, q_{H_1}]$ randomly. When $\mathcal{F}_I$ makes an H_1 query on ID_i where $1 \le i \le q_{H_1}$, if $i = j$ (we let $ID_i = ID^*$ at this point), $\mathcal{C}$ returns $Q_{ID_i} = bP$ and adds $\langle ID_i, Q_{ID_i}, k_i = \perp\rangle$ to L_{H_1}. Otherwise $\mathcal{C}$ picks a random $k_i \in \mathbb{Z}_q^*$ and returns $Q_{ID_i} = k_i P$, and adds $\langle ID_i, Q_{ID_i}, k_i\rangle$ to L_{H_1}.

- H_2 query: When $\mathcal{F}_I$ makes this query on PK_{ID_i}, if the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns y_{ID_i}. Otherwise, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns y_{ID_i}, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.
- H_3 query: When $\mathcal{F}_I$ makes this query on (m_i, U_i), if the list L_{H_3} contains $\langle m_i, U_i, h_i\rangle$, $\mathcal{C}$ returns h_i. Otherwise $\mathcal{C}$ picks a random $h_i \in \mathbb{Z}_q^*$ and returns h_i, and adds $\langle m_i, U_i, h_i\rangle$ to L_{H_3}.
- ExtrPartSK(ID_i) query: When $\mathcal{F}_I$ makes this query on ID_i, if $ID_i \neq ID^*$, $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, k_i\rangle$ in L_{H_1}, and returns $D_{ID_i} = k_i aP$. Otherwise $\mathcal{C}$ outputs FAIL and aborts the simulation.
- RequestPK(ID_i) query: When $\mathcal{F}_I$ makes this query on ID_i, if the list L_K contains $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$, $\mathcal{C}$ returns PK_{ID_i}. Otherwise, $\mathcal{C}$ picks a random $x_{ID_i} \in \mathbb{Z}_q^*$. Then $\mathcal{C}$ returns $PK_{ID_i} = x_{ID_i}P$ and adds $\langle ID_i, PK_{ID_i}, x_{ID_i}, 1\rangle$ to L_K.
- ExtrFullSK(ID_i) query: When $\mathcal{F}_{II}$ makes this query on ID_i, if $ID_i = ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, k_i\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$ in L_{H_1} and L_K, respectively. $\mathcal{C}$ performs as follows:
  - If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns $SK_{ID_i} = \frac{1}{x_{ID_i}+y_{ID_i}} k_i aP$.
  - If the list L_{H_2} does not contain $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns $SK_{ID_i} = \frac{1}{x_{ID_i}+y_{ID_i}} k_i aP$, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.
- ReplacePK(ID_i, PK'_{ID_i}) query: When $\mathcal{F}_I$ makes this query on (ID_i, PK'_{ID_i}), $\mathcal{C}$ performs as follows:
  - If the list L_K contains $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$, $\mathcal{C}$ sets $PK_{ID_i} = PK'_{ID_i}$ and $c = 0$.
  - If the list L_K does not contain $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$, $\mathcal{C}$ makes a RequestPK query on ID_i itself. Then $\mathcal{C}$ sets $PK_{ID_i} = PK'_{ID_i}$ and $c = 0$.
- SIGN(m, ID_i) query: When $\mathcal{F}_I$ makes this query on (ID_i, m), $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, k_i\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$ in L_{H_1} and L_K, respectively. Then $\mathcal{C}$ performs as follows:
  - If $c = 1$, $\mathcal{C}$ picks two random $r_i, h_i \in \mathbb{Z}_q^*$ and finds $\langle PK_{ID_i}, y_{ID_i}\rangle$ in L_{H_2}. If it does not exist, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}. $\mathcal{C}$ computes $U_i = r_i(x_{ID_i} + y_{ID_i})P - h_i Q_{ID_i}$ and $V_i = r_i aP$. $\mathcal{C}$ then returns (U_i, V_i) and adds $\langle m_i, U_i, h_i\rangle$ to L_{H_3} ($\mathcal{C}$ outputs FAIL and aborts the simulation if the $\langle m_i, U_i, h_i\rangle$ has already been defined in the list L_{H_3}).
  - If $c = 0$, $\mathcal{C}$ gets additionally information x'_{ID_i} from $\mathcal{F}_I$. Using the x'_{ID_i}, $\mathcal{C}$ then simulates as in the above case ($c = 1$).

Eventually, $\mathcal{F}_I$ outputs a valid signature $(ID_t, m_t, \sigma_t = (U_t, V_t))$. If $ID_t \neq ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle m_t, U_t, h_t\rangle$ in L_{H_3}. Then by replays of $\mathcal{C}$ with the same random tape but different choices of H_3 (it is to apply the 'forking' technique formalized in [14]), $\mathcal{C}$ gets another valid signature tuple $(ID_t, m_t, h'_t, \sigma_t = (U_t, V'_t))$ such that $h_t \neq h'_t$. $\mathcal{C}$ finds $\langle ID_t, PK_{ID_t}, x_{ID_t}, c\rangle$ in L_K. If $c = 0$, that is, $\mathcal{F}_I$ generated a public/private key pair and replaced the public key of ID_t. In such case, as the proof in [10], we assume that $\mathcal{C}$ keeps track of the public/private key pair generated by $\mathcal{F}_I$. Hence, after $\mathcal{C}$ finds $\langle PK_{ID_i}, y_{ID_i}\rangle$ in L_{H_2}, he can compute as follows:

$$(x_{ID_t} + y_{ID_t})\frac{V_t - V_t'}{h_t - h_t'} = SK_{ID_t} = abP.$$

Therefore, if a Type I forger who can break our scheme eCLS exists, then an attacker who solves the CDH problem exists. □

Theorem 2. *Our certificateless signature scheme* eCLS *is existentially unforgeable against a Type II forger in random oracle model under the mICDH assumption.*

Proof. Suppose there exists a forger $\mathcal{F}_{II}$ which has advantage in attacking our CLS scheme eCLS. We want to build an algorithm $\mathcal{C}$ that uses $\mathcal{F}_{II}$ to solve the mICDH problem. $\mathcal{C}$ receives a mICDH instance (P, aP, b) for randomly chosen $a, b \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$. Its goal is to compute $(a+b)^{-1}P$. $\mathcal{C}$ runs $\mathcal{F}_{II}$ as a subroutine and simulates its attack environment. $\mathcal{C}$ picks a random $s \in \mathbb{Z}_q^*$ and sets `master-key`$= s$. $\mathcal{C}$ then gives system parameters with `master-key` to $\mathcal{F}_{II}$. Without loss of generality, we assume that any extraction (RequestPK, ExtrFullSK) and signature (SIGN) queries are preceded by H_1 query, and the SIGN and ExtrFullSK queries are preceded by RequestPK query. To avoid collision and consistently respond to these queries, $\mathcal{C}$ maintains four lists L_{H_1}, L_{H_2}, L_{H_3}, $L_K = \{\langle ID, PK_{ID}, x_{ID}\rangle\}$ which are initially empty. $\mathcal{C}$ then simulates the oracle queries of $\mathcal{F}_{II}$ as follows:

– H_1 query: When $\mathcal{F}_{II}$ makes this query on ID_i, $\mathcal{C}$ picks a random $k_i \in \mathbb{Z}_q^*$ and returns k_iP, and adds $\langle ID_i, k_i\rangle$ to L_{H_1}.
– H_2 query: When $\mathcal{F}_{II}$ makes this query on PK_{ID_i}, if $PK_{ID_i} = aP$, $\mathcal{C}$ sets $y_{ID_i} = b$ and returns y_{ID_i}, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}. If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns y_{ID_i}. Otherwise, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns y_{ID_i}, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.
– H_3 query: When $\mathcal{F}_{II}$ makes this query, $\mathcal{C}$ performs as in the proof of Theorem 1.
– RequestPK(ID_i) query: Suppose $\mathcal{F}_{II}$ makes at most q_{PK} queries to public key request oracle. First, $\mathcal{C}$ chooses $j \in [1, q_{PK}]$ randomly. When $\mathcal{F}_{II}$ makes a RequestPK query on ID_i, if $i = j$ (we let $ID_i = ID^*$ at this point), $\mathcal{C}$ sets $PK_{ID_i} = aP$ and returns PK_{ID_i}, and adds $\langle ID_i, PK_{ID_i}, x_{ID_i} = \perp\rangle$ to L_K. Otherwise $\mathcal{C}$ picks a random x_{ID_i} and returns $PK_{ID_i} = x_{ID_i}P$, and adds $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ to L_K.
– ExtrFullSK(ID_i) query: When $\mathcal{F}_{II}$ makes this query on ID_i, if $ID_i = ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle ID_i, k_i\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ in L_{H_1} and L_K, respectively. $\mathcal{C}$ performs as follows:
  • If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns $SK_{ID_i} = \frac{1}{x_{ID_i}+y_{ID_i}}k_isP$.
  • If the list L_{H_2} does not contain $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns $SK_{ID_i} = \frac{1}{x_{ID_i}+y_{ID_i}}k_isP$, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.

– $\mathsf{SIGN}(m, ID_i)$ query: When $\mathcal{F}_{II}$ makes this query on (ID_i, m), $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, k_i\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ in L_{H_1} and L_K, respectively. $\mathcal{C}$ picks two random r_i, $h_i \in \mathbb{Z}_q^*$ and finds $\langle PK_{ID_i}, y_{ID_i}\rangle$ in L_{H_2} (if it does not exist, $\mathcal{C}$ makes a H_2 query on PK_{ID_i} itself). $\mathcal{C}$ computes $(U_i = r_i(PK_{ID_i} + y_{ID_i}P) - h_i k_i P$, $V_i = r_i sP$ and returns (U_i, V_i), and adds $\langle m_i, U_i, h_i\rangle$ to L_{H_3} ($\mathcal{C}$ outputs FAIL and aborts the simulation if the $\langle m_i, U_i, h_i\rangle$ has already been defined in the list L_{H_3}).

Eventually, $\mathcal{F}_{II}$ outputs a valid signature $(ID_t, m_t, \sigma_t = (U_t, V_t))$. If $ID_t \neq ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle m_t, U_t, h_t\rangle$ in L_{H_3}. Then, as in the proof of Theorem 1, $\mathcal{C}$ gets another valid signature tuple $(ID_t, m_t, h_t', \sigma_t = (U_t, V_t'))$ such that $h_t \neq h_t'$. Since $\mathcal{C}$ knows the master key s, after $\mathcal{C}$ finds $\langle ID_t, k_t\rangle$ in L_{H_1}, he can compute as follows:

$$\frac{V_t - V_t'}{k_t s(h_t - h_t')} = \frac{1}{a+b}P.$$

Therefore, if a Type II forger who can break our scheme eCLS exists, then an attacker who solves the CDH problem exists. □

4 Certificateless Signature Scheme with a Pairing Operation

In this section, we propose a CLS scheme with a pairing operation, and prove the security of the proposed scheme. We denote this CLS scheme by oCLS.

4.1 Our Construction

Setup. To generate system parameters and master key, run as follows:

1. Generate $(\mathbb{G}_1, \mathbb{G}_2, e)$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order q and e is an admissible bilinear map.
2. Choose a random $s \in \mathbb{Z}_q^*$ and a generator P of $\mathbb{G}_1$. Compute $P_{pub} = sP$ and $g = e(P, P)$.
3. Choose three cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_2 : \mathbb{G}_1 \to \mathbb{Z}_q^*$, and $H_3 : \{0,1\}^* \to \mathbb{Z}_q^*$.

Return the private `master-key`$= s$ and the system parameters `params`$= \{e, \mathbb{G}_1, \mathbb{G}_2, q, g, P, P_{pub}, H_1, H_2, H_3\}$. We assume that `params` is available to all users.

Partial-Private-Key-Extract. On input `params`, `master-key`, and identity ID_A of user A. Compute $q_A = H_1(ID_A)$ and return a partial private key $D_A = \frac{1}{s+q_A}P$ for user A.

Set-Secret-Value. On input k and ID_A, choose a random value $x_A \in \mathbb{Z}_q^*$ and return x_A as A's secret value.

Set-Public-Key. On input `params` and x_A, compute $Q_A = P_{pub} + H_1(ID_A)P$ and $R_A = x_A Q_A$. Return the public key $PK_A = R_A$.

Set-Private-Key. On input x_A, R_A, and D_A. Compute $y_A = H_2(R_A)$ and $S_A = \frac{1}{x_A+y_A} D_A$. Return the (full) private key $SK_A = S_A$.

Sign. On input `params`, ID_A, S_A, and a message m, perform the following steps:

1. Choose a random $r \in \mathbb{Z}_q^*$.
2. Compute $U = g^r = e(P, P)^r$.
3. Set $h = H_3(m, U)$.
4. Compute $V = (r + h)S_A$.
5. Return $\sigma = (U, V)$ as the signature on the message m.

Verify. On input `params`, ID_A, R_A, m, and $\sigma = (U, V)$. Compute $Q_A = (s + q_A)P = P_{pub} + H_1(ID_A)P$, $y_A = H_2(R_A)$, and $h = H_3(m, U)$. Check if $e(V, R_A + y_A Q_A) = Ug^h$ holds. If the equation holds, it outputs 1, otherwise 0.

We can easily show that our CLS scheme satisfies completeness property as follows:

$$\begin{aligned} e(V, R_A + y_A Q_A) &= e((r+h)S_A, x_A(P_{pub} + q_A P) + y_A(P_{pub} + q_A P)) \\ &= e\Big((r+h)\frac{1}{(x_A + y_A)(s + q_A)}P, (x_A + y_A)(s + q_A)P\Big) \\ &= e((r+h)P, P) \\ &= e(P, P)^{r+h} = Ug^h. \end{aligned}$$

4.2 Security Analysis

To prove the security of the oCLS, we review the k-CAA (Collusion Attack Algorithm with k traitor) problem.

k-CAA [13] problem: The k-CAA problem is to compute $\frac{1}{s+t_0}P$ for some $t_0 \in \mathbb{Z}_q^*$ when given

$$P, sP, \quad t_1, t_2 ..., t_k \in \mathbb{Z}_q^*, \quad \frac{1}{s+t_1}P, \ \frac{1}{s+t_2}P, ..., \frac{1}{s+t_k}P.$$

Theorem 3. *Our certificateless signature scheme oCLS is existentially unforgeable against a Type I forger in random oracle model under the k-CAA assumption.*

Proof. Suppose there exists a forger $\mathcal{F}_I$ which has advantage in attacking our CLS scheme oCLS. We want to build an algorithm $\mathcal{C}$ that uses $\mathcal{F}_I$ to solve the k-CAA problem. $\mathcal{C}$ receives a k-CAA instance $(P, sP, t_1, ..., t_k, \frac{1}{s+t_1}P, ..., \frac{1}{s+t_k}P)$ where $k \geq q_{H_1}$ (we suppose $\mathcal{F}_I$ makes at most q_{H_1} queries to H_1 oracle). Its goal is to compute $\frac{1}{s+t_0}P$ for some t_0. $\mathcal{C}$ runs $\mathcal{F}_I$ as a subroutine and simulates its attack environment. $\mathcal{C}$ sets $g = e(P, P)$ and $P_{pub} = sP$ where s is the master key, which is unknown to $\mathcal{C}$, and gives system parameters to $\mathcal{F}_I$. Without loss of generality, we assume that any extraction (ExtrPartSK, RequestPK, ExtrFullSK) and signature (SIGN) queries are preceded by H_1 query, and the

SIGN and ExtrFullSK queries are preceded by RequestPK query. To avoid collision and consistently respond to these queries, $\mathcal{C}$ maintains four lists L_{H_1}, L_{H_2}, L_{H_3}, $L_K = \{\langle ID, PK_{ID}, x_{ID}, c(= 0 \text{ or } 1)\rangle\}$ which are initially empty. $\mathcal{C}$ then simulates the oracle queries of $\mathcal{F}_I$ as follows:

- When $\mathcal{F}_I$ makes H_2, H_3, and ReplacePK queries, $\mathcal{C}$ performs as in the proof of Theorem1.
- H_1 query: $\mathcal{F}_I$ makes an H_1 query on ID_i where $1 \le i \le q_{H_1}$, $\mathcal{C}$ chooses $j \in [1, q_{H_1}]$ randomly. If $i = j$ (we let $ID_i = ID^*$ at this point), $\mathcal{C}$ returns $q_{ID_i} = t_0$, otherwise $q_{ID_i} = t_i$. $\mathcal{C}$ then computes $Q_{ID_i} = sP + q_{ID_i}P$ and adds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ to L_{H_1}.
- ExtrPartSK(ID_i) query: When $\mathcal{F}_I$ makes this query on ID_i, if $ID_i \ne ID^*$, $\mathcal{C}$ returns $D_{ID_i} = \frac{1}{s+t_i}P$. Otherwise $\mathcal{C}$ outputs FAIL and aborts the simulation.
- RequestPK(ID_i) query: When $\mathcal{F}_I$ makes this query on ID_i, if the list L_K contains $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$, $\mathcal{C}$ returns PK_{ID_i}. Otherwise, $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ in L_{H_1}, and picks a random $x_{ID_i} \in \mathbb{Z}_q^*$. $\mathcal{C}$ then returns $PK_{ID_i} = x_{ID_i}Q_{ID_i}$ and adds $\langle ID_i, PK_{ID_i}, x_{ID_i}, 1\rangle$ to L_K.
- ExtrFullSK(ID_i) query: When $\mathcal{F}_{II}$ makes this query on ID_i, if $ID_i = ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$ in and L_K, respectively. $\mathcal{C}$ performs as follows:
  - If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns $SK_{ID_i} = (x_{ID_i} + y_{ID_i})^{-1}\frac{1}{s+q_{ID_i}}P$.
  - If the list L_{H_2} does not contain $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns $SK_{ID_i} = (x_{ID_i} + y_{ID_i})^{-1}\frac{1}{s+q_{ID_i}}P$, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.
- SIGN(m, ID_i) query: When $\mathcal{F}_I$ makes this query on (ID_i, m), $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, k_i\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}, c\rangle$ in L_{H_1} and L_K, respectively. Then $\mathcal{C}$ performs as follows:
  - If $c = 1$, $\mathcal{C}$ picks two random $r_i, h_i \in \mathbb{Z}_q^*$ and finds $\langle PK_{ID_i}, y_{ID_i}\rangle$ in L_{H_2}. If it does not exist, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}. $\mathcal{C}$ computes $U_i = g^{-h_i}e((r_i + h_i)P, Q_{ID_i})$ and $V_i = (r_i + h_i)\frac{1}{s+q_{ID_i}}P$. $\mathcal{C}$ then returns (U_i, V_i) and adds $\langle m_i, U_i, h_i\rangle$ to L_{H_3} ($\mathcal{C}$ outputs FAIL and aborts the simulation if the $\langle m_i, U_i, h_i\rangle$ has already been defined in the list L_{H_3}).
  - If $c = 0$, $\mathcal{C}$ gets additionally information x'_{ID_i} from $\mathcal{F}_I$. Using the $Q'_{ID_i} = x'_{ID_i}(sP + q_{ID_i}P)$, $\mathcal{C}$ then simulates as in the above case ($c = 1$).

Eventually, $\mathcal{F}_I$ outputs a valid signature $(ID_t, m_t, \sigma_t = (U_t, V_t))$. If $ID_t \ne ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ performs as in the proof of Theorem 1., and then he computes as follows:

$$(x_{ID_t} + y_{ID_t})\frac{V_t - V_t'}{h_t - h_t'} = SK_{ID_t} = \frac{1}{s + q_0}P.$$

Therefore, if a Type I forger who can break our scheme oCLS exists, then an attacker who solves the k-CAA problem exists. □

Theorem 4. *Our certificateless signature scheme* oCLS *is existentially unforgeable against a Type II forger in random oracle model under the mICDH assumption.*

Proof. This proof is same as the proof of Theorem 2. The different points are as follows:

- H_1 query: When $\mathcal{F}_{II}$ makes this query on ID_i, $\mathcal{C}$ picks a random $q_{ID_i} \in \mathbb{Z}_q^*$ and returns q_{ID_i}. $\mathcal{C}$ then computes $Q_{ID_i} = sP + q_{ID_i}P$ and adds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ to L_{H_1}.
- H_2 query: When $\mathcal{F}_{II}$ makes this query on PK_{ID_i}, if $PK_{ID_i} = saP + q_{ID_i}aP$, $\mathcal{C}$ sets $y_{ID_i} = b$ and returns y_{ID_i}, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}. If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns y_{ID_i}. Otherwise, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns y_{ID_i}, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.
- H_3 query: When $\mathcal{F}_{II}$ makes this query, $\mathcal{C}$ performs as in the proof of Theorem 1.
- $\mathsf{RequestPK}(ID_i)$ query: Suppose $\mathcal{F}_{II}$ makes at most q_{PK} queries to public key request oracle. First, $\mathcal{C}$ chooses $j \in [1, q_{PK}]$ randomly. When $\mathcal{F}_{II}$ makes a $\mathsf{RequestPK}$ query on ID_i, $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ in L_{H_1}. If $i = j$ (we let $ID_i = ID^*$ at this point), $\mathcal{C}$ sets $PK_{ID_i} = saP + q_{ID_i}aP$ and returns PK_{ID_i}, and adds $\langle ID_i, PK_{ID_i}, x_{ID_i} = \perp\rangle$ to L_K. Otherwise $\mathcal{C}$ picks a random x_{ID_i} and returns $PK_{ID_i} = x_{ID_i}(sP + q_{ID_i}P)$, and adds $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ to L_K.
- $\mathsf{ExtrFullSK}(ID_i)$ query: When $\mathcal{F}_{II}$ makes this query on ID_i, if $ID_i = ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ in L_{H_1} and L_K, respectively. $\mathcal{C}$ performs as follows:
  - If the list L_{H_2} contains $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ returns $SK_{ID_i} = (x_{ID_i} + y_{ID_i})^{-1}\frac{1}{s+q_{ID_i}}P$.
  - If the list L_{H_2} does not contain $\langle PK_{ID_i}, y_{ID_i}\rangle$, $\mathcal{C}$ picks a random $y_{ID_i} \in \mathbb{Z}_q^*$ and returns $SK_{ID_i} = (x_{ID_i} + y_{ID_i})^{-1}\frac{1}{s+q_{ID_i}}P$, and adds $\langle PK_{ID_i}, y_{ID_i}\rangle$ to L_{H_2}.
- $\mathsf{SIGN}(m, ID_i)$ query: When $\mathcal{F}_{II}$ makes this query on (ID_i, m), $\mathcal{C}$ finds $\langle ID_i, Q_{ID_i}, q_{ID_i}\rangle$ and $\langle ID_i, PK_{ID_i}, x_{ID_i}\rangle$ in L_{H_1} and L_K, respectively. $\mathcal{C}$ picks two random r_i, $h_i \in \mathbb{Z}_q^*$ and finds $\langle PK_{ID_i}, y_{ID_i}\rangle$ in L_{H_2} (if it does not exist, $\mathcal{C}$ makes a H_2 query on PK_{ID_i} itself). $\mathcal{C}$ computes $U_i = g^{-h_i} \cdot e(PK_{ID_i}, r_iP) \cdot e(y_{ID_i}(s + q_{ID_i})P, r_iP)$, $V_i = r_iP$ and returns (U_i, V_i), and adds $\langle m_i, U_i, h_i\rangle$ to L_{H_3} ($\mathcal{C}$ outputs FAIL and aborts the simulation if the $\langle m_i, U_i, h_i\rangle$ has already been defined in the list L_{H_3}).

Eventually, $\mathcal{F}_{II}$ outputs a valid signature $(ID_t, m_t, \sigma_t = (U_t, V_t))$. If $ID_t \neq ID^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, $\mathcal{C}$ finds $\langle m_t, U_t, h_t\rangle$ in L_{H_3}. Then, as in the proof of Theorem 1, $\mathcal{C}$ gets another valid signature tuple $(ID_t, m_t, h'_t, \sigma_t = (U_t, V'_t))$ such that $h_t \neq h'_t$. Since $\mathcal{C}$ knows the master key s, after $\mathcal{C}$ finds $\langle ID_t, Q_{ID_t}, q_{ID_t}\rangle$ in L_{H_1}, he can compute as follows:

$$(s + q_{ID_t})\frac{V_t - V'_t}{h_t - h'_t} = \frac{1}{a+b}P.$$

Therefore, if a Type II forger who can break our scheme oCLS exists, then an attacker who solves the CDH problem exists. □

5 Performance Analysis

We now compare our CLS schemes with other previously known CLS schemes [10,12,20] in Table 1.

Table 1. Comparison of Certificateless Signature Schemes

Schemes	Sign			Verify		
	e	G$_1$	G$_2$	e	G$_1$	G$_2$
HSMZ05 [10]	2	2	0	5	1	0
LCS05 [12]	0	2	0	4	1	0
ZWXF06 [20]	0	3	0	4	0	0
eCLS	0	2	0	2	2	0
oCLS	0	1	1	1	1	1

(e: pairing operation, G$_1$: multiplication in $\mathbb{G}_1$, G$_2$: exponentiation in $\mathbb{G}_2$)

According to the result in [3,4], the pairing operation is several times more expensive than the scalar multiplication in $\mathbb{G}_1$. Hence reducing the number of pairing operations is critical. As we shown in Table 1, our CLS schemes are more efficient than other previous schemes. In particular, our verification procedure requires only one (or two) pairing operation because checking the validity of the public key is not done separately.

6 Extension

The ring signature guarantees the anonymity of the signer. In certificateless ring signature (CLRS) schemes, to guarantee the signer anonymity, a signature generated should not reveal any information about both a signer's identity and his public key. This is different from the previous (certificate or identity-based) ring signature schemes, because the ring signature schemes hide a public key or identity of the actual signer. Using the similar techniques as in [19,7], it is possible that our CLS schemes are expanded to CLRS schemes. Also, applying the method in [18] to our CLS scheme, we can easily modify our eCLS to a certificateless blind signature scheme.

7 Conclusion

The certificateless public key cryptography is receiving significant attention because it is a new paradigm that simplifies the public key cryptography. In this paper, we proposed two efficient CLS schemes which are provably secure in the random oracle model. Particularly, our signature verification requires only one pairing operation. In addition, our CLS scheme can be easily extended to certificateless ring and blind signature schemes.

References

1. S. Al-Riyami and K. Paterson, *Certificateless public key cryptography*, Asiacrypt 2003, LNCS 2894, pp. 452-473, Springer-Verlag, 2003.
2. D. Boneh and X. Boyen, *Short Signatures Without Random Oracles*, Eurocrypt 2004, LNCS 3027, pp. 56-73, 2004.
3. P. S. L. M. Barreto, S. Galbraith, C. O. hEigeartaigh, and M. Scott, *Efficient Pairing Computation on Supersingular Abelian Varieties*, Crypto 2002, LNCS 2442, pp. 354-368, Springer-Verlag, 2002.
4. P. S. L. M. Barreto, B. Lynn, and M. Scott, *Efficient implementation of pairing-based cryptosystems*, Journal of Cryptology, pp. 321-334, 2004.
5. D. Boneh, H. Shacham, and B. Lynn, *Short signatures from the Weil pairing*, Journal of Cryptology, Vol. 17, No. 4, pp. 297-319, 2004.
6. X. Cao, K. G. Paterson and W. Kou *An Attack on a Certificateless Signature Scheme*, http://eprint.iacr.org/2006/367.
7. Sherman S.M. Chow, S.M. Yiu, and Lucas C.K. Hui, *Efficient Identity Based Ring Signature*, ACNS 2005, LNCS 3531, pp. 499-512, Springer-Verlag, 2005.
8. P. Gutmann, *PKI: It's not dead, just resting*, IEEE Computer, 35(8), pp. 41-49, 2002.
9. M. C. Gorantla and A. Saxena, *An Efficient Certificateless Signature Scheme*, CIS 2005, LNAI 3802, pp. 110-116, Springer-Verlag, 2005.
10. X. Huang, W. Susilo, Y. Mu, and F. Zhang, *On the security of certificateless signature schemes from asiacrypt 2003*, CANS 2005, LNCS 3810, pp. 13-25, Springer-Verlag, 2005.
11. B. C. Hu, D. S. Wong, Z. Zhang, and X. Deng, *Key Replacement Attack Against a Generic Construction of Certificateless Signature*, ACISP 2006, LNCS 4058, pp. 235-246, Springer-Verlag, 2006.
12. X. Li, K. Chen, and L. Sun, *Certificateless signature and proxy signature schemes from bilinear pairings*, Lithuanian Mathematical Journal, Vol. 45, No. 1, pp. 76-83, 2005.
13. S. Mitsunari, R. Sakai and M. Kasahara, *A new traitor tracing*, Proc. of IEICE Trans. Vol. E85-A, No.2, pp.481-484, 2002.
14. D. Pointcheval and J. Stern, *Security Proofs for Signature Schemes*, Eurocrypt 1996, LNCS 1070, pp. 387-398, Springer-Verlag, 1996.
15. A. Shamir, *Identity based cryptosystems and signature schemes*, Crypto 1984, LNCS, Vol.196, pp. 47-53, Springer-Verlag, 1984.
16. A. R. Sadeghi and M. Steiner, *Assumptions related to discrete logarithms: why subtleties make a real difference*, Eurocrypt 2001, LNCS 2045, pp. 243-260, Springer-Verlag, 2001.
17. D. H. Yum and P. J. Lee, *Generic Constructin of Certificateless Signature*, ACISP 2004, LNCS 3108, pp. 200-211, Springer-Verlag, 2004.

18. F. Zhang and K. Kim, *Efficient ID-Based Blind Signature and Proxy Signature from Bilinear Pairings*, ACISP 2003, LNCS 2727, pp. 312-323, Springer-Verlag, 2003.
19. F. Zhang, R. Safavi-Naini and W. Susilo, *An Efficient Signature Scheme from Bilinear Pairings and Its Applications*, PKC 2004, LNCS 2947, pp. 277-290, Springer-Verlag, 2004.
20. Z. Zhang, D. Wong, J. Xu and D. Feng, *Certificateless Public-Key Signature: Security Model and Efficient Construction*, ACNS 2006, LNCS 3989, pp. 293-308, Springer-Verlag, 2006.