Efficient Certificateless Signcryption from Bilinear Pairings

Hongzhen Du

School of Mathematics and Information Science, Baoji University of Arts and Sciences, Baoji, 721013, China
hongzhendu@163.com

Abstract

Certificateless public key cryptography is an attractive paradigm which combines advantages of both traditional certificate-based and ID-based public key cryptosystems. In this paper, the author formalizes the definition and the security model for certificateless signcryption (CLSC), and proposes a CLSC scheme that is proved to be secure in the random oracle model. The CLSC scheme satisfies such desirable properties as message confidentiality, signature unforgeability, public verifiability and forward secrecy. Moreover, the scheme achieves significant efficiency in that it requires general cryptographic hash functions instead of the MapToPoint function which is inefficient and probabilistic, and the public/private key of a user in the scheme is a single group element, and it is unnecessary to check the validity of the public key, and signcryption algorithm needs no pairing operation while the reverse operation only requires two pairing computations.

Keywords: certificateless public key cryptography, signcryption, bilinear pairings, security

1. Introduction

Identity-based public key cryptosystem (ID-PKC) which was introduced by Shamir [1] in 1984 alleviates the burden of certificate management in a traditional public key cryptosystem (PKC). However, there inherently exists a defect named private key escrow problem in ID-PKC because it involves a Private Key Generator (PKG), which can generate a user’s private key based on his identity. So, the PKG can literally decrypt any ciphertext and forge any user’s signature on any message. To solve the inherent key escrow problem in ID-PKC, Al-Riyami and Paterson [2] introduced a new approach called certificateless public key cryptography (CL-PKC) in 2003. The CL-PKC is intermediate between traditional PKC and ID-PKC. In a CL-PKC, a user’s private key is not produced by the PKG alone. Instead, it is composed of a partial private key produced by the PKG and some secret value chosen by the user. As a result, the PKG does not know the user’s private key. In such a way, the key escrow problem of ID-PKC can be solved.

Due to the lack of authentication of public keys in CL-PKC, it is difficult to construct a secure and efficient certificateless scheme. Although many schemes [3-15] based on certificateless cryptosystem have been presented following the first work from Al-Riyami and Paterson [2], schemes in [3, 4, 6, 8, 9, 10] are proved to be insecure. Furthermore, most certificateless schemes are inefficient and unpractical with respect to the cost of computation and the length of public/private key. For example, the verification algorithm in certificateless signature schemes [4, 5, 10] requires four expensive pairing computations, and public key in schemes [2, 4, 7, 10] is two group elements and the validity of public key has to be verified, while the verification of public key is costly and requires at least two pairing computations.
The notion of signcryption was presented by Zheng[16] in 1997, and this primitive aims to efficiently perform encryption and signature in a single logical step. The first certificateless signcryption scheme was proposed by Barbosa and Farshim [17] in 2008. Li et al. [18] proposed a certificateless hybrid signcryption scheme. Unfortunately, Selvi et al. [19] show the scheme is existentially forgeable and then give an improved scheme. Wu et al. [20] and Xie et al. [21] separately put forward two certificateless signcryption schemes and give their security proofs in the random oracle. However, Selvi et al. [22, 23] point out the two schemes are insecure against the Type I adversaries. In 2009, Liu et al. [24] constructed a certificateless signcryption scheme in the standard model. Unfortunately, Weng et al. [25] point out that Liu et al.’s scheme is vulnerable to the Type II adversary’s attack. Jing et al. [26] proposed a CLSC scheme without bilinear pairings, but Shi et al. [28] prove the scheme does not meet the properties of confidentiality and unforgeability.

Contributions. This paper studies signcryption in certificateless public key cryptography. The author proposes a new CLSC scheme and provides detailed security proofs for the scheme. The CLSC scheme satisfies all the current requirements of the signcryption. In addition, the length of public/private key of a user in the scheme is only one group element. Moreover, the scheme never requires public key validity checking. And the scheme achieves significant efficiency in that it requires general cryptographic hash functions instead of the MapToPoint hash function, and signature verification requires only two pairing computations while no pairing operation is needed in signcryption algorithm.

The remaining sections are organized as follows: In the next section, the author gives a brief introduction to bilinear pairings and some mathematical theories. Section 3 provides the framework of CLSC. The author proposes an efficient CLSC scheme and gives its efficiency analysis, and provides its security proof in the random oracle model in Section 4. Conclusions are drawn in the last section.

2. Preliminaries

Let \( G_1 \) be a cyclic additive group of prime order \( p \), and \( G_2 \) be a cyclic multiplicative group of the same order \( p \). A bilinear pairing is a map \( e : G_1 \times G_1 \rightarrow G_2 \) which satisfies the following properties:

1) Bilinearity 
\[
e(aP, bQ) = e(P, Q)^{ab}, \text{ where } P, Q \in G_1, a, b \in \mathbb{Z}_p^*
\]

2) Non-degeneracy 
There exist \( P, Q \in G_1 \) such that \( e(P, Q) \neq 1 \)

3) Computability 
There is a computable algorithm to get \( e(P, Q) \) for all \( P, Q \in G_1 \).

It is intractable to solve the following problems within polynomial time.

- k-CAA: For an integer \( k \), and \( s \in \mathbb{Z}_p^* \), \( P \in G_1 \). Given \{ \( P, sP, e_1, e_2, \ldots, e_k \in \mathbb{Z}_p^* \), and \( \frac{1}{s+e_1} P, \frac{1}{s+e_2} P, \ldots, \frac{1}{s+e_k} P \} \), to compute \( \frac{1}{s+e} P \), where \( e \notin \{e_1, e_2, \ldots, e_k \} \).

- Computational Diffie-Hellman Problem (CDHP): For unknown random values \( a, b \in \mathbb{Z}_p^* \), given \( aP, bP \), computing \( abP \).

- Computational Diffie-Hellman Inversion Problem (CDHIP): For an unknown value \( a \in \mathbb{Z}_p^* \), given \( P, aP \), to compute \( a^{-1}P \).

- Modified Computational Diffie-Hellman Inversion problem (MCDHIP):
Given \( b \in \mathbb{Z}_p^* \), \( P, aP \) for unknown value \( a \in \mathbb{Z}_p^* \), to compute \((a + b)^tP\).

- **q-Bilinear Diffie-Hellman Inversion Problem (q-BDHIP):** Given a tuple \((P, \alpha P, \alpha^2 P, \ldots, \alpha^q P)\) as input, computing \(e(P, P)^{1/\alpha}\).

- **Bilinear Diffie-Hellman Inversion problem (BDHIP):** Given \( P, \alpha P \), computing \(e(P, P)^{1/\alpha}\).

- **Modified Bilinear Diffie-Hellman Inversion problem (MBDHIP):** Given \( b \in \mathbb{Z}_p^* \), \( P, aP \) for unknown value \( a \in \mathbb{Z}_p^* \), computing \(e(P, P)^{1/(a+b)}\).

3. Frameworks of Certificateless Signcryption

(1) Definition of Certificateless Signcryption

A CLSC scheme is composed of seven algorithms as follows.

**Setup:** Taking a security parameter \( l \) as input and returns the system parameters, \( \text{params} \) and a master key \( s \).

**Partial-Private-Key-Extract:** On inputs \( \text{params}, s \) and a user’s identity \( ID \), the algorithm returns a partial private key \( d_{ID} \).

**Set-Secret-Value:** Taking as inputs \( \text{params} \) and a user’s identity \( ID \), the algorithm returns a secret value \( r \).

**Set-Private-Key:** Taking as inputs \( \text{params} \), a user’s partial private key \( d_{ID} \) and his secret value \( r \), and it returns the full private key \( sk_{ID} \).

**Set-Public-Key:** Taking as inputs \( \text{params} \) and a user’s secret value \( r \), and generates a public key \( pk_{ID} \) for the user.

**CL-Signcrypt:** is a probabilistic algorithm that takes as inputs \( \text{params} \), a message \( m \), a recipient’s identity \( ID_b \) and his public key \( pk_b \), and the sender’s private key \( sk_A \), and returns a ciphertext \( \sigma = \text{CL-Signcrypt}(m, sk_A, Q_{ID_b}, pk_b) \).

**CL-Unsigncrypt:** Taking as inputs \( \text{params} \), a ciphertext \( \sigma \), the recipient’s private key \( sk_b \), and the sender’s identity \( ID_A \) and his corresponding public key \( pk_A \). It returns a message-signature pair \((m, S)\) or a distinguished symbol \( \bot \) indicating an unsigncryption failure.

(2) Security Model for Certificateless Signcryption

The standard security notion for a signcryption scheme is indistinguishability of signcryption against an adaptive chosen-ciphertext attacker (IND-CCA2) and existentially unforgeable against an adaptive chosen-message attacker (EUF-CMA2).

As defined in security model for certificateless public key cryptography [2], there are two types of adversaries in a CLSC scheme: a Type I adversary \( A_1 \) and a Type II adversary \( A_2 \). A Type I adversary does not know the KGC’s master key but can replace public keys of arbitrary identities with other public keys of its choice. In contrast, a Type II adversary knows the KGC’s master key but is disallowed to replace public key of any user during the game. The formal security models for CLSC schemes were formalized by Barbosa and Farshim [17]. I refer the readers to literature [17].

4. An Efficient Certificateless Signcryption Scheme

4.1. The basic scheme

In this section, I present a CLSC scheme which is secure against both Type I and Type
II adversaries in the random oracle model.

Setup: Given a security parameter \( l \), the PKG chooses two groups \( G_i \) and \( G_s \) of same prime order \( p>2^l \) and a modified Weil pairing map \( e: G_1 \times G_1 \rightarrow G_2 \). \( P \) is a generator of group \( G_1 \). Let \( g=e(P,P) \), then PKG selects four distinct cryptographic hash functions such that \( H_1: \{0,1\} \rightarrow \mathbb{Z}_p^* \), \( H_2: G_1 \rightarrow \mathbb{Z}_p^* \), \( H_3: G_2 \rightarrow \{0,1\} \) and \( H_4: \{0,1\} \times G_2 \rightarrow \mathbb{Z}_p^* \). And picks a random \( s \in \mathbb{Z}_p^* \) as its master key and computes its public key \( P_0=sP \in G_1 \). Let \( E, D \) be the encryption and decryption algorithms of a secure symmetric cipher. Afterwards, PKG publishes the system public parameters, \( \text{params} = \{l, G_i, G_s, e, p, P, g, P_0, H_1, H_2, H_3, H_4, E, D\} \), but keeps \( s \) secret.

**Partial-Private-Key-Extract:** Given an identity \( ID_i \in \{0,1\}^l \), \( i \in \{A,B\} \), PKG computes \( Q_i = H_1(ID_i) \), \( d_i = \frac{1}{s+Q_i}P \), and sends \( d_i \) to a user with identity \( ID_i \) as his partial private key by a secure channel.

**Set-Secret-Value:** The user with identity \( ID_i \) picks a random value \( x_i \in \mathbb{Z}_p^* \) as his secret value.

**Set-Public-Key:** Taking as inputs \( \text{params} \) and a user’s secret value \( x_i \), the algorithm generates the user’s public key as \( pk_i = x_i(P_0 + Q_iP) \). For simplicity, we set \( T_i = P_0 + Q_iP \).

**Set-Private-Key:** Given \( \text{params} \), a user’s partial private key \( d_i \) and his secret value \( x_i \), the algorithm outputs \( sk_i = \frac{1}{(x_i + H_2(pk_i))(s+Q_i)}P \) as the user’s private key.

**CL-Signcrypt.** Given a message \( m \in \{0,1\}^l \), a recipient’s identity \( ID_B \) and his public key \( pk_B \), and a sender’s private key \( sk_A \), the sender with the identity \( ID_A \) and his public key \( pk_A \) works as follows:

1) Pick a random \( r \in \mathbb{Z}_p^* \), and compute \( R = g^r \), set \( K = H_3(R) \) and compute \( c = E_A(m \parallel ID_A \parallel pk_A) \);
2) Set \( h = H_4(m, R, ID_A, ID_B) \);
3) Compute \( S = (r+h)sk_A = \frac{(r+h)}{(x_A + H_2(pk_A))(s+Q_A)}P \);
4) Compute \( U = r(pk_B + H_2(pk_B)T_B) \in G_i \).

The ciphertext is \( \sigma = (c, U, S) \).

**CL-Unsigncrypt.** Given a ciphertext \( \sigma = (c, U, S) \) and a sender’s identity \( ID_A \) and his public key \( pk_B \), the recipient with identity \( ID_B \) and his public key \( pk_B \) acts as follows:

1) The recipient uses his private key \( sk_B \) to compute \( R \).

\[
e(U, sk_B) = e \left( x_B T_B + H_2(pk_B)T_B, \frac{P}{(x_B + H_2(pk_B))(s+Q_B)} \right)
= e \left( x_B + H_2(pk_B)(s+Q_B)P, \frac{P}{(x_B + H_2(pk_B))(s+Q_B)} \right)
= e(P, P)^r = R.
\]
And sets $K = H_a(R)$, and then recovers $D_k(c) = m \parallel ID_A \parallel pk_A$.

2) Set $h = H_4(m, R, ID_A, ID_B)$.

3) Accept the message $m$ and return the message-signature pair $(m, (h, S))$ if and only if the following equation holds.

$$h = H_4\left( m, e(S, pk_A + H_2(pk_A)T_A) \cdot g^{-h}, ID_A, ID_B \right)$$

The consistency of the scheme is easy to verify as follows:

$$e\left( H_2(pk_A)T_A, \right) e\left( P, \right) e\left( P, e^{h} \right) = e\left( H_2(pk_A)T_A, \right) e\left( P, \right) e\left( P, e^{h} \right) = R$$

Hence, $h = H_4\left( m, e(S, pk_A + H_2(pk_A)T_A) \cdot g^{-h}, ID_A, ID_B \right)$

Public Verifiability

To prove to a third party that the sender actually signcrypted a message $m$, the recipient forwards the tuple $(m, (h, S))$ to the third party. Then the third party accepts the proof by verifying whether the following equation holds

$$h = H_4\left( m, e(S, pk_A + H_2(pk_A)T_A) \cdot g^{-h}, ID_A, ID_B \right)$$

4.2. Security

**Theorem 1** In the random oracle model, the proposed CLSC scheme is secure against an adaptive chosen-ciphertext attack (IND-CLSC-CCA2) under the $q$-BDHIP and MBDHIP assumptions.

The theorem follows from Lemmas 1 and 2.

**Lemma 1.** Assume that an IND-CLSC-CCA2 adversary $A_1$ has an advantage $\varepsilon$ against the above scheme when running in a time $t$, making $q_h$ queries to random oracles $H_i$ ($i = 1, 2, 3, 4$), $q_{sc}$ queries to the Signcryption oracle and $q_{us}$ queries to the Unsigncryption oracle. Then there exists an algorithm $X$ to solve the $q$-BDHIP for $q = q_h$ with probability

$$\varepsilon' \geq \frac{\varepsilon(q_h - 1)}{q_h(1 - q_{sc}q_h + 2q_h)}(1 - q_{sc}q_h + 2q_h)\left(1 - \frac{q_{us}}{2^j}\right),$$

with a time $t' < t + (O(q^2) + (3q + 2q_{sc} + 2q_{us}))t_{ms} + (q_h + 2)t_{exp} + (4q_{us} + 1)t_{pr}$, where $t_{ms}$, $t_{exp}$ and $t_{pr}$ denote respectively the costs of a scalar multiplication in $G_1$, an exponentiation in $G_2$ and a pairing computation.

**Proof.** Suppose that $X$ is given a challenge: Given $<P, sP, s^2P, \ldots, s^tP>$, for an unknown value $s \in \mathbb{Z}_p^*$, $X$’s task is to output $e(P, P)^{t/s}$ after interacting with $A_1$. $X$ will interact with $A_1$ as follows:
Setup: X uses its input to compute a generator $Q \in G_i$ and sets $P_0 = xQ$ for $x = k_i^{-1} s - w$ such that it knows $q$ pairs $(w_i, \frac{1}{w_i + x})$ for $i \in \{1, 2, \ldots, q-1\}$ as in the proof technique of [29]. X acts as follows:

1) Choose $w_1, w_2, \ldots, w_{q-1} \in Z_p^*$ at random and construct a polynomial $f(t) = \prod_{i=0}^{q-1} (t + w_i)$ , and expand it to obtain $c_0, c_1, \ldots, c_{q-1} \in Z_p^*$ so that $f(t) = \sum_{i=0}^{q-1} c_i t^i$.

2) Pick values $k_i \in Z_p^*$ for $I \in \{1, 2, \ldots, q_h\}$ and $w \in Z_p^*$, where $w \notin \{w_1, w_2, \ldots, w_{q-1}\}$. Suppose $x = k_i^{-1} s - w$, X computes the following:

$$Q = f(x)P = \sum_{i=0}^{q-1} c_i (xP) = \sum_{i=0}^{q-1} c_i (k_i^{-1} s - w)^i P = \sum_{i=0}^{q-1} c_i' (s^i P)$$

and compute $sQ = sf(x)P = \sum_{i=0}^{q-1} c_i' (s^{i+1} P)$.

3) Set $P_0 = k_i^{-1} \cdot sQ - wQ = xQ$ and $g = e(Q, Q)$. X sends params = $(G_i, G_2, Q, P_0, g, H_1, H_2, H_3, H_4, E, D)$ to $A_1$.

$H_1$-Queries: X maintains a hash list $H_i^{iua}$ of tuple $(ID_i, Q_i)$ ($i \in \{1, 2, \ldots, q_h\}$) as explained below. The list is initially empty. When $A_1$ makes a hash oracle query on $ID_i$, X returns the previously defined value if it exists. Otherwise, X acts as follows: If $ID_i = ID_j$, X sets $Q_i = w$ and returns $Q_i$ to $A_j$. Otherwise, X answers $Q_i = w_i$ for $w_i \in \{w_1, w_2, \ldots, w_{q-1}\}$. In both cases, X inserts $(ID_i, Q_i)$ into $H_i^{iua}$.

Partial-Private-Key-Extraction Queries: X maintains a list $E^{iua}$ of tuple $(ID_i, Q_i, d_i)$ is initially empty. For any given identity $ID_i$, X recovers the corresponding tuple $(ID_i, Q_i)$ from the list $H_i^{iua}$, if $ID_i \neq ID_j$, then returns $d_i = \frac{1}{x + Q_i}$ to $A_1$ and adds $(ID_i, Q_i, d_i)$ to the $E^{iua}$. Otherwise, X aborts and outputs “failure” (denote the event by $E_1$).

Public-Key-Extraction Queries: X maintains a list $pk^{list}$ of tuple $(ID_i, Q_i, pk_i, x_i)$ which is initially empty. For on input $ID_j$, X picks a random $x_i \in Z_p^*$, computes $pk_i = x_i (P_0 + Q_iQ) = x_i T_i$ and returns $pk_i$. Then adds $(ID_i, Q_i, pk_i, x_i)$ to the list $pk^{list}$.

$H_2$-Queries: For a query on input $ID_j$ and $pk_i$, If $ID_j = ID_i$, X recovers the corresponding tuple $(ID_j, Q_j, pk_j, x_j)$ from the list $pk^{list}$ and sends $h_{2j} = k_j - x_j \in Z_p^*$ to $A_1$. (If the identity $ID_j$’s public key $pk_j$ has been replaced by $A_1$, we assume that $C$ can additionally obtain the secret value corresponding to the replaced public key). Otherwise, X picks a random $h_{2j} \in Z_p^*$ and returns it to $A_1$. In both cases, X adds tuple $(ID_j, pk_j, h_{2j})$ to the list $H_2^{iua}$.

Private-Key-Extraction Queries: For a query on input $ID_j$, If $ID_j = ID_i$, X stops and outputs “failure” (denotes the event by $E_2$). Otherwise, X performs as follows:
- If lists $E_{\text{list}}$, $pk_{\text{list}}$ and $H_{2\text{list}}$ contain the corresponding tuple $(ID_i, Q_i, d_i)$, $(ID_i, Q_i, pk_i, x_i)$ and $(ID_i, pk_i, h_{2i})$ respectively, $X$ sets $sk_i = \frac{d_i}{(x_i + h_{2i})}$ and sends it to $A_i$.

- Otherwise, $X$ makes a Partial Private Key Extraction query, a Public Key Extraction query and a $H_3$ query on $ID_i$, then simulates as the above process and sends $sk_i = \frac{d_i}{(x_i + h_{2i})}$ to $A_i$.

**Public-Key-Replacement Queries:** When $A_i$ queries on input $(ID_i, pk_i)$, $X$ checks whether an element $(ID_i, Q_i, pk_i, x_i)$ is contained in the $pk_{\text{list}}$. If it does, $X$ sets $pk_i = pk_i'$ and adds $(ID_i, Q_i, pk_i, x_i = \bot)$ to the $pk_{\text{list}}$. Otherwise, $X$ executes public key extraction to generate $(ID_i, Q_i, pk_i, x_i)$, then sets $pk_i = pk_i'$ and adds $(ID_i, Q_i, pk_i, x_i = \bot)$ to the $pk_{\text{list}}$.

$H_3$-Queries: For an input $R \in G_2$, $X$ picks a random value $h_3 \in Z_p^*$ and returns it to $A_i$, and inserts $(R, h_3)$ into the list $H_{3\text{list}}$.

$H_4$-Queries: For an input $(m, R, ID_u, ID_v)$ with identities $ID_u / ID_v$ of sender/recipient, $X$ answers the defined value if it exists and a random value $h_4 \in Z_p^*$ otherwise. Additionally, $X$ computes $\delta = R \cdot e(Q, Q)^{h_3}$ and inserts a record $(m, R, ID_u, ID_v, h_4, \delta)$ into a list $H_{\text{list}}^4$.

**Signcrypt Queries:** When a Signcrypt query on a message $m$ for sender/recipient with identity $ID_u / ID_v$ for $u, v \in \{1, \cdots, q_h\}$ and corresponding public key $pk_u / pk_v$ is coming, $X$ acts as follows:

- If $ID_u = ID_i$, $X$ does not know the sender’s private key $sk_u$, but he knows the recipient’s private key $sk_v$ since $ID_v \neq ID_i$ (If the recipient’s public key $pk_v$ has been replaced by $A_i$, we assume that $X$ can keep track of the secret value generated by $A_i$). $X$ finds the corresponding tuple $(ID_i, pk_i, h_{2i})$ and the tuple $(ID_i, pk_i, h_{2i})$ from $H_{2\text{list}}$, and chooses two random values $a, h_4 \in Z_p^*$, computes $U = a(pk_i + h_{2i}) - h_4(pk_i + h_{2i})$, $S = a \cdot sk_v$, $R = e(a(pk_i + h_{2i}), sk_v) / e(Q, Q)^{h_3}$ and then backpatches to define the value $H_4(m, R, ID_u, ID_v)$ as $h_4$ ($X$ aborts in the unlikely event that $H_4(m, R, ID_u, ID_v)$ is already defined, denote the event by $E_3$). Then, $X$ finds the corresponding tuple $(R, h_3)$ from $H_{3\text{list}}$ and computes $c = E_{h_3}(m \parallel ID_i \parallel pk_i)$. At last, $X$ outputs a ciphertext $\sigma = (c, U, S)$.

- Otherwise, $X$ has the sender’s private key and can answer the query according to the specification of the Signcrypt algorithm.

**Unsigncrypt Queries:** Suppose the request is on a ciphertext $\sigma = (c, U, S)$ for identities $(ID_u, ID_v)$ for $u, v \in \{1, \cdots, q_h\}$ and with corresponding public key $(pk_u, pk_v)$. $C$ acts as follows:

- If $ID_u = ID_i$, $X$ has not the recipient’s private key $sk_v$, but he knows the sender’s private key $sk_u$ since $ID_u \neq ID_i$. Obviously, the equation
\[ e(U, sk_u) = e(pk_i + h_2T_i, S - h_3sk_u) \] holds, where \( H_d(m, R, ID_u, ID_v) \) is the hash value obtained in the Signcrypt algorithm. \( X \) computes \( \delta = e(pk_i + h_2T_u, S) \) and recovers the tuple \((m, R, ID_u, ID_v, h_4, \delta)\) from \( H^*_3 \). If this tuple does not exist, \( \sigma = (c, U, S) \) is rejected, denote the event by \( E_4 \) and further examines whether the following equality holds:
\[ e(U, sk_u) / e(pk_i + h_2T_i, S) \neq e(pk_i + h_2T_i, -h_3sk_u) . \]

If the equation holds, \( X \) outputs the message-signature pair \((m, (h_4, S))\), otherwise, \( \sigma = (c, U, S) \) is rejected.

- If \( ID_v \neq ID_i \), \( X \) knows the recipient’s private key and can normally run the Unsigncrypt algorithm.

Finally, \( A_1 \) outputs messages \((m_0, m_1)\) and identities \((ID_k^*, ID_r^*)\) with public keys \((pk_s^*, pk_r^*)\), where the private key of identity \( ID_k^* \) was not corrupted. Here note \( ID_k^* \) does not have been submitted to both oracles Public-Key-Replacement and Partial-Private-Key-Extract. If \( ID_r^* \neq ID_i \), \( X \) outputs “failure” and aborts (denote the event by \( E_5 \)). Otherwise, \( X \) chooses \( \lambda \in \mathbb{Z}_p^* \) and computes \( U^* = \lambda Q \), picks randomly \( c^* \in \mathbb{G}_1 \) (\( \mathbb{G}_1 \) denotes the ciphertext space) and \( S^* \in \hat{G}_1 \) to return the challenge \( \sigma^* = (c^*, S^*, U^*) \). Here if we define \( \lambda = rs \) for \( r \in \mathbb{Z}_p^* \) and we can check that \( U^* = \lambda Q = rsQ = r(P_0 + Q_0Q)(x_1 + h_2x_2) = r(pk_i + h_2T_i) \).

To omit \( A_1 \)’s eventual output, \( X \) fetches tuple \((R, h_4)\) or \((m, R, h_4, c, \delta)\) from list \( H^*_3 \) or \( H^*_4 \), the chosen tuple will contain the right element
\[ R = e(Q, Q') = e(P, P')^{(x_2^2) / (x_2^3)} \] . Then the q-BDHP solution can be extracted. Since
\[ R = e(Q, Q') = e(Q, Q)^{(x_2^3) / (x_2^3)} = e((1 / s) f(x)P, f(x)P)^{x_2^3} = e((1 / s) \sum_{i=0}^{q-1} c_i (s^iP), \sum_{i=0}^{q-1} c_i (s^iP))^{x_2^3} = e(c_0'P + \sum_{i=0}^{q-1} c_i (s^iP), \sum_{i=0}^{q-1} c_i (s^iP))^{x_2^3} . \]

Hence, \( X \) can successfully compute and output
\[ e(P, P')^{x_2^3} = \left\{ R | e(c_0'P + \sum_{i=0}^{q-1} c_i (s^iP), \sum_{i=0}^{q-1} c_i (s^iP))^{x_2^3} \right\} \]
as a solution to \( A_1 \)’s challenge.

Now analyze the advantage of \( X \) in this game. The responses to \( A_1 \)’s \( H_i \) for \( i \in \{1, 2, 3, 4\} \) queries are indistinguishable from the real life. We note that \( X \) only fails in providing a consistent simulation because one of the events \( E_1, E_2, E_3, E_4, E_5 \). Hence, if none of the events \( E_1, E_2, E_3, E_4 \) and \( E_5 \) happens, \( X \) can solve the q-BDHP successfully.

Now we assess the probability for these events. From the description above, we have \( \Pr[E_1] = 1/5 \), we can observe that \( -E_5 \) means \( -E_2 \) and we also know that \( \Pr[E_1] \leq q_m(q_m + q_h) / 2^l \), \( \Pr[E_4] \leq q_m / 2^l \) and \( \Pr[-E_3] = 1/5 \).

So, we can compute as follows:
Pr(\neg E_1 \land \neg E_2 \land \neg E_3 \land \neg E_4) \geq \frac{q_h}{q_h - 1}(1 - \frac{q_w}{2^j})(1 - \frac{q_w}{2^j})(1 - \frac{q_w}{2^j})(1 - \frac{q_w}{2^j}).

We obtain the announced bound by noting that X fetches the correct element from list \( H_{3^{\text{list}}} \) or \( H_{4^{\text{list}}} \) with probability \( \frac{1}{q_h + 2q_h} \).

Taking the above analysis on these events, we know that challenger X's advantage

\[ \epsilon' \geq \frac{\epsilon (q_h - 1)}{q_h^2(q_h + 2q_h)(1 - \frac{q_w}{2^j})}. \]

From the above description of X, we can conclude that the running time of X is bound by

\[ t' < t + (O(q^2) + (3q + 3q_w + 2q_w)t_{sm} + (q_h + 2)t_{exp} + (4q_w + 1)t_{pr}}. \]

**Lemma 2.** Assume that an IND-CLSC-CCA2 adversary \( A_2 \) has an advantage \( \epsilon \) against our scheme when running in time \( t \), asking \( q_{hs} \) queries to random oracles \( H_i \) \((i = 1, 2, 3, 4)\), \( q_w \) queries to signcryption oracle and \( q_{at} \) queries to the unsigncryption oracle. Then there is an algorithm X to solve the MBDHIP with probability

\[ \epsilon' \geq \frac{\epsilon}{q_h(q_h + 2q_h)(1 - \frac{q_w}{2^j})}. \]

within a time \( t' < t + (3q_h + 3q_w + q_w)t_{sm} + (q_h + q_w + q_w)t_{exp} + 2q_w + t_{pr} \), where \( t_{sm}, t_{exp} \) and \( t_{pr} \) also denote respectively the costs of a scalar multiplication in \( G_1 \), an exponentiation in \( G_2 \) and a pairing computation.

**Proof.** Suppose X is given a challenge of a random instance of the MBDHIP:

Given \( h_{32} \equiv Z_p^*, P, aP \) and \((a + h_{23})P \in G_1\), where \( a \) is unknown to X. X's goal is to output \( e(P, P)^{\lambda(a + h_{23})} \) by interacting with adversary \( A_2 \).

**Setup:** X runs algorithm Setup, randomly picks a value \( s \in Z_p^* \) as the system master key, sets \( g = e(P, P) \) and \( P_r = sP \), and picks an identity \( ID_2 \) at random as the challenge ID in this game, and gives \( \text{params} \{ P, g, P_r, H_1, H_2, H_3 \} \) and the system master key \( s \) to \( A_2 \).

**H_1-Queries:** X maintains a hash list \( H_{1^{\text{list}}} \) of tuple \((ID_1, Q)\) as explained below. The list is initially empty. When \( A_2 \) makes a hash oracle query on \( ID_1 \), if the query \( ID_1 \) has already appeared on the \( H_{1^{\text{list}}} \), then the previously defined value is returned. Otherwise, X randomly picks a value \( Q \in Z_p^* \) and returns it to \( A_2 \). Then, adds \((ID_1, Q)\) to \( H_{1^{\text{list}}} \).

**Public Key Extraction Queries:** X maintains a list \( pk_{1^{\text{list}}} \) of tuple \((ID_1, Q, pk, x)\) which is initially empty. When \( A_2 \) queries on input \( ID_1 \), X checks whether \( pk_{1^{\text{list}}} \) contains a tuple for this input. If it does, the previously defined value is returned. Otherwise, X works as follows:

- If \( ID_1 = ID_2 \), X finds the tuple \((ID_1, Q)\) in \( H_{1^{\text{list}}} \) and sets \( pk_i = saP + Q, aP \) and sends \( pk_i \) to \( A_2 \), and inserts \((ID_1, Q, pk, \bot)\) into \( pk_{1^{\text{list}}} \).

- Otherwise, X recovers the tuple \((ID_1, Q)\) from \( H_{1^{\text{list}}} \) and picks a random \( x_i \in Z_p^* \), computes \( pk_i = x_i(P_r + Q, P) \) and returns \( pk_i \) and adds \((ID_1, Q, pk_i, x)\) into the \( pk_{1^{\text{list}}} \).

**H_2-Queries:** For query on input \( ID_2 \) and \( pk_i \), X picks a random \( h_{23} \in Z_p^* \) and returns it to \( A_2 \), then adds tuple \((ID_2, pk, h_{23})\) to the list \( H_{2^{\text{list}}} \).

**Private Key Extraction Queries:** For query on input \( ID_j \), if \( ID_j = ID_1 \), X stops and outputs “failure”. Otherwise, X performs as follows:
- If the lists $pk^{list}_u$ and $H^{list}_2$ contain the corresponding tuple $(ID_i, Q_i, pk_i, x_i)$ and $(ID_i, pk_i, h_{2i})$ respectively, $X$ sets $sk_i = \frac{P}{(x_i + h_{2i})(s + Q_i)}$ and sends it to $A_2$.

- Otherwise, $X$ makes public key extraction query and $H_2$ query on $ID_i$, then simulates as the above process and sends $sk_i = \frac{P}{(x_i + h_{2i})(s + Q_i)}$ to $A_2$.

$H_3$-Queries: For an input $R \in G_2$, $X$ returns the defined value if it exists and a random $h_3 \in Z_p^*$ otherwise, and then inserts $(R, h_3)$ into the list $H^{list}_3$.

$H_4$-Queries: For an input $(m, R)$ with identities $ID_u/ID_v$ of sender/recipient, $X$ answers the defined value if it exists and a random $h_4 \in Z_p^*$ otherwise. Additionally, $X$ recovers the list $H^{list}_3$ and finds $h_3 = H_3(R) \in Z_p^*$ (If $h_3$ does not exist, $X$ simulates $H_3$ oracle on its own to obtain $h_3 = H_3(R) \in Z_p^*$) and inserts the elements $(m, R, h_4, c, \delta = R \cdot e(P, P)^h_4)$ into the list $H^{list}_4$.

Signcrypt Queries: When $A_2$ asking a signcrypt query on a message $m$ for sender/recipient with identities $ID_u/ID_v$, for $u, v \in \{1, \cdots, q_h\}$ and corresponding public key $pk_u/pk_v$, $X$ acts as follows:

If $ID_{u} = ID_{v}$, $X$ does not know the sender’s private key $sk_u$, but he knows the recipient’s private key $sk_v$ since $ID_u \neq ID_v$. $X$ randomly chooses $\varphi \in Z_p^*$ and $\psi \in Z_p^*$ and computes $U = \varphi(pk_j + h_{2j}T_j) - \psi(pk_v + h_{2v}T_v)$, $S = \varphi \cdot sk_v$, where $T_i = P_0 + Q_iP(i \in \{I, v\})$ and backpatches to define the value $H_4(m, R)$ as $\psi$ ($X$ aborts in the unlikely event that $H_4(m, R)$ is already defined). The ciphertext is $\sigma = (c = E_{h_i}(m \| Q_i), U, S)$.

- Otherwise, $X$ knows the sender’s private key and can answer the query according to the specification of signcrypt algorithm.

Unsigncrypt Queries: Suppose the request is on a ciphertext $\sigma = (c, U, S)$ for identities $(ID_u, ID_v)$ for $u, v \in \{1, \cdots, q_h\}$ and with corresponding public key $(pk_u, pk_v)$. $X$ acts as follows:

- If $ID_v = ID_i$, $X$ has not the recipient’s private key $sk_v$, but he knows the sender’s private key $sk_u$ since $ID_u \neq ID_v$. Obviously, the equation $e(U, sk_u) = e(pk_i + h_{2i}T_i, S - h_u sk_u)$ holds, where $h_i = H_4(m, R)$ is the hash value obtained in the signcrypt algorithm. $X$ computes $\delta = e(pk_u + h_{2u}T_u, S)$ and recovers the tuple $(m, R, h_u, c, \delta)$ from $H^{list}_4$ (If this tuple does not exist, $\sigma = (c, U, S)$ is rejected) and further examines whether the following equality holds: $e(U, sk_u)/e(pk_i + h_{2i}T_i, S) = e(pk_i + h_{2i}T_i, -h_u sk_u)$.

If the equation holds, $X$ outputs the message-signature pair $(m, (h_i, S))$, otherwise, $\sigma = (c, U, S)$ is rejected.

- If $ID_v \neq ID_i$, $X$ knows the recipient’s private key and can normally run the unsigncrypt algorithm.

Finally, $A_2$ outputs messages $(m_0, m_1)$ and identities $(ID_{i0}, ID_{i1})$ with public keys $(pk_{i0}, pk_{i1})$. Here $ID_{i1}$ may not have been submitted to Private-Key-Extract query.
If $ID^*_R \neq ID^*_I$, X outputs “failure” and aborts. Otherwise, X chooses $\lambda \in Z^*_p$ and computes $U^* = \lambda(P_0 + Q_I P)$, picks randomly $c^* \in \mathbb{G}$ (here $\mathbb{G}$ denotes the ciphertext space) and $S^* \in G_1$, and returns the challenge $\sigma^* = (c^*, S^*, U^*)$. Here if we define $\lambda = r(a + h_{2_1})$ for $r \in Z^*_p$ and we can check that

$$U^* = \lambda(P_0 + Q_I P) = r(a + h_{2_1})(P_0 + Q_I P) = r(pk_I + h_{2_1}T_I).$$

A2 then makes a new series of queries but is not permitted to ask for the unsigncrypt oracle of $\sigma^* = (c^*, S^*, U^*)$ for the combination $(ID_{s^*}^*, ID_{R^*}^*)$ and $(pk_{s^*}, pk_{R^*})$. Moreover no private key exposure query can be made on $ID_{R^*}^*$ at any time.

To ignore A2’s eventual output, X fetches tuple $(R, h_3)$ or $(m, R, h_4, c, \delta)$ from list $H^*_3$ or $H^*_4$, the chosen tuple will contain the right element $R = e(P, P)^{\lambda(a + h_{2_1})}$. Hence, X can successfully compute $e(P, P)^{\lambda(a + h_{2_1})} = R^{x^{-1}}$ and output $R^{x^{-1}}$ as a solution to $A_2$’s challenge. So, X breaks MBDHIP. The analysis of X’s advantage is similar to that of Lemma 1. This completes my proof.

**Theorem 2.** In the random oracle model, the proposed CLSC scheme is existentially unforgeable against adaptive chosen messages attacks (EUF-CLSC-CMA2) under the $k$-CAA and MCDHIP assumptions.

The theorem follows from Lemma 3 and lemma 4.

**Lemma 3.** In the random oracle model, the CLSC scheme is secure against any EUF-CLSC-CMA2 adversary $A_1$ if the $k$-CAA problem is hard in $G_1$.

**Proof.** Suppose that X receives a challenge: Given \{ $P, sP, e_1, e_2, \ldots, e_k \in \mathbb{G}^*$, and $\frac{1}{s + e_1} P, \frac{1}{s + e_2} P, \ldots, \frac{1}{s + e_k} P$ \}, for an unknown value $s \in \mathbb{Z}^*_p$, X’s task is to compute $\frac{1}{s + e_0}$ for some $e_0 \in \{ e_1, e_2, \ldots, e_k \}$. X will interact with $A_1$ as follows:

**Setup:** X sets $g = e(P, P)$ and $P_0 = sP$, and picks an identity $ID_I$ at random as the challenge ID in this game, and gives params \{ $P, g, P_0, H_1, H_2, H_3, H_4$ \} to $A_1$.

**$H_i$-Queries:** X maintains a hash list $H^*_{1,i}$ of tuple $(ID, Q_i)$ $i \in \{ 1, \ldots, q_k \}$ as explained below. The list is initially empty. When $A_1$ makes a hash oracle query on $ID_I$, X returns the previously defined value if it exists. Otherwise, X acts as follows: If $ID_I = ID_J$, X sets $Q_j = e_i$ and returns $Q_j$ to $A_1$. Otherwise, X answers $Q_j = e_i$ for $i \in \{ e_1, e_2, \ldots, e_k \}$. In both cases, X inserts $(ID_I, Q_j)$ into $H^*_{1,i}$.

- Partial Private Key Extraction Queries, Public Key Extraction Queries, $H_2, H_3, H_4$ -Queries, Private Key Extraction Queries, Public Key Replacement, Signcrypt Queries and Unsigncrypt Queries: The responses to these queries are similar to the corresponding answers in Lemma 1.

Eventually, $A_1$ outputs a ciphertext $\sigma^* = (c^*, U^*, S^*)$ for identities $(ID_{s^*}^*, ID_{R^*}^*)$ with public keys $(pk_{s^*}, pk_{R^*})$, where the private key of identity $ID_{s^*}^*$ was not corrupted. Here note $ID_{s^*}^*$ does not have been submitted to both oracles Public-Key-Replacement and Partial-Private-Key-Extract. If $ID_{s^*}^* \neq ID_I$, X outputs “failure” and aborts. Otherwise, X computes $R^* = e(U^*, sk_{R^*})$ and finds the tuple $(m, R^*, h_4)$ in $H^*_4$. Then by replays of X
with the same random tape but different choices of $H_4$, X can obtain another valid signature $(R^*, h_4^*, S^*)$ for $h_4^* \neq h_4^\omega$. Then, we can get

$$S^* = (r^* + h_4^*)s_k^* = \frac{(r^* + h_4^*)}{(x_5^* + h_2^*)(s + e_0)} P$$

$$S^\omega = (r^* + h_4^\omega)s_k^* = \frac{(r^* + h_4^\omega)}{(x_5^* + h_2^*)(s + e_0)} P.$$

$$S^* - S^\omega = \frac{h_4^* - h_4^\omega}{x_5^* + h_2^*} \frac{1}{s + e_0} P.$$

As a result, X can output $\frac{1}{s + e_0} P = \frac{(x_5^* + h_2^*)(S^* - S^\omega)}{h_4^* - h_4^\omega}$ as a solution to $A_1$’s challenge.

**Lemma 4.** In the random oracle model, the CLSC scheme is secure against any EUF-CLSC-CMA2 adversary $A_2$ if the MCDHIP is hard in $G_1$.

Lemma 4 can be formally proven in the same way as that of Lemma 3. So, I omit it here.

### 4.3. Efficiency Analysis

Here notations $Sm$, $Ex$ and $Pr$ denote the cost of computing a scalar multiplication in $G_1$, an exponentiation in $G_2$, and one pairing computation, respectively. If we also select the base group $G_1$ and the bilinear map from elliptic curves [29], which results in a group of 160 bits size. So, the public/private key size of our scheme is about 160 bits, and message expansion is 320 bits length which is the same as that of Zheng’s scheme [16].

The hash function maps an identity to an element in $G_1$ used by these schemes [17, 18, 27] usually requires a MapToPoint operation [30], which is inefficient and probabilistic. Our scheme avoids this problem since it does not require mapping an identity onto a cyclic group. ("$Pr^\ast\ast$" denotes the pairing can be pre-computed.)

**Table 1. Comparison between My Scheme and Other Schemes**

<table>
<thead>
<tr>
<th>CLSC</th>
<th>Signcryption</th>
<th>Unsigncryption</th>
<th>Public-Key-Size</th>
<th>Private-Key-Size</th>
<th>Message Expansion</th>
</tr>
</thead>
<tbody>
<tr>
<td>[17]</td>
<td>$4Sm+1Pr+1Ex$</td>
<td>$5Pr+1Sm$</td>
<td>160bits</td>
<td>320bits</td>
<td>320bits</td>
</tr>
<tr>
<td>[18]</td>
<td>$4Sm+1Pr+1Ex$</td>
<td>$5Pr+1Sm$</td>
<td>160bits</td>
<td>320bits</td>
<td>320bits</td>
</tr>
<tr>
<td>[21]</td>
<td>$2Sm+2Ex$</td>
<td>$4Sm+2Ex+2Pr$</td>
<td>160bits</td>
<td>320bits</td>
<td>480bits</td>
</tr>
<tr>
<td>[27]</td>
<td>$3Sm+2Ex$</td>
<td>$2Sm+1Pr+2Pr^\ast$</td>
<td>320bits</td>
<td>320bits</td>
<td>480bits</td>
</tr>
<tr>
<td>My Scheme</td>
<td>$3Sm+2Ex$</td>
<td>$1Ex+1Sm+2Pr$</td>
<td>160bits</td>
<td>160bits</td>
<td>320bits</td>
</tr>
</tbody>
</table>

### 5. Conclusions

Signcryption is a useful tool and it can achieve confidentiality, authentication, integration and non-repudiation in an efficient and secure manner. This paper proposes a CLSC scheme that is proved to be secure in the random oracle. The proposed scheme satisfies all the current requirements of the signcryption schemes and it achieves significant efficiency and its ciphertext expansion is short such that it can be used in environments with bandwidth constraints.

**Acknowledgements**

This work was supported by the National Natural Science Foundation of China (Grant No. 61402015), Scientific Research Project of Shaanxi Provincial Education Department.
A New Provably Secure Certificateless Short Signature Scheme

References


Author

Hongzhen Du, she received the Ph.D. degree from Beijing University of Posts and Telecommunications in 2009. From 2011, she is an associate professor at Baoji University of Arts and Sciences. Her research interests include cryptography, digital signature. E-mail:hongzhendu@163.com.