Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Low-computation certificateless hybrid signcryption scheme*

Hui-fang YU†1, Bo YANG2,3

(1School of Computer, Qinghai Normal University, Xining 810008, China)

(2School of Computer Science, Shaanxi Normal University, Xi'an 710062, China)

(3State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China)

†E-mail: yuhuifang@qhnu.edu.cn

Received Feb. 25, 2016; Revision accepted July 10, 2016; Crosschecked July 14, 2017

Abstract: Hybrid signcryption is an important technique signcrypting bulk data using symmetric encryption. In this paper, we apply the technique of certificateless hybrid signcryption to an elliptic-curve cryptosystem, and construct a low-computation certificateless hybrid signcryption scheme. In the random oracle model, this scheme is proven to have indistinguishability against adaptive chosen-ciphertext attacks (IND-CCA2) under the elliptic-curve computation Diffie-Hellman assumption. Also, it has a strong existential unforgeability against adaptive chosen-message attacks (sUF-CMA) under the elliptic-curve discrete logarithm assumption. Analysis shows that the cryptographic algorithm does not rely on pairing operations and is much more efficient than other algorithms. In addition, it suits well to applications in environments where resources are constrained, such as wireless sensor networks and ad hoc networks.

Key words: Hybrid signcryption; Scalar multiplication; Certificateless cryptosystem; Provable security
http://dx.doi.org/10.1631/FITEE.1601054 **CLC number:** TP309

1 Introduction

Public key signcryption (Zhang and Xu, 2010; Li *et al*., 2012; Pang *et al*., 2012; Youn and Hong, 2012; Wang and Teng, 2015) can simultaneously fulfill encryption and signature in a logic step with lower computational cost than conventional signature-then-encryption methods. In other words, a signcryption scheme in public key setting is more computationally efficient than a direct composition of encryption and signature. However, public key signcryption often constrains the message space when someone wants to signcrypt a message of arbitrary length. Luckily, this problem has been solved by hybrid signcryption (Dent, 2005). A hybrid signcryption scheme comprises two components: (1) a signcryption key encapsulation mechanism (KEM) that uses a public key technique to generate a symmetric key and encapsulates it, and (2) a data encapsulation mechanism (DEM) that employs the symmetric key from signcryption KEM to encrypt the message of arbitrary length. Each of the two parts has its own security criteria, independent of the operation of the other. This implies that we can study the security of asymmetric signcryption KEM and symmetric DEM separately. Hybrid signcryption schemes are more efficient than signcryption schemes not using DEM, especially in the case of encrypting data in bulk.

Later, Tan (2008) proposed full insider secure signcryption KEM and KEM with a tag (tag-KEM) without random oracles. Wang *et al*. (2012) devised a post-quantum hybrid signcryption scheme under the lattice assumption. To alleviate the problem of low computational efficiency in the existing schemes, Yu and Yang (2015a) devised an efficient identity-based hybrid signcryption scheme using random oracles. A certificateless cryptosystem is regarded as a good

* Project supported by the National Natural Science Foundation of China (Nos. 61572303, 61363080, and 61272436), the Foundation of State Key Laboratory of Information Security (No. 2015-MS-10), and the Foundation of Basic Research of Qinghai Province, China (No. 2016-ZJ-776)

ORCID: Hui-fang YU, http://orcid.org/0000-0003-4711-3218

substitute for traditional public key cryptosystem (TPKC) and identity-based cryptosystem (IBC), since it features no certificate and no key escrow with only a little more computation. Certificateless hybrid signcryption (CLHS) schemes from pairings (Li *et al*., 2013; Yu and Yang, 2015b) eliminate the key escrow issue since the key generation center (KGC) is unable to access the secret value of any user. Only a valid user with both partial private key and secret value can perform the related cryptographic operation.

The security of the elliptic-curve cryptosystem (ECC) (Koblitz, 1987) is based on the computational intractability of the elliptic-curve discrete logarithm (ECDL) problem. Any cryptosystem using ECC can provide a high security with a small key size. For example, an elliptic-curve key defined on an additive group with a 160-bit length is considered to be as secure as an RSA key with a 1024-bit length (Li *et al*., 2016). ECC is particularly useful in settings where storage space, power consumption, bandwidth, or processing power is constrained. In general, the computational cost of one pairing operation is higher than that of ECC scalar multiplication. For instance, NanoECC, which uses the MIRACL library, takes around 17.93 s to compute one pairing operation and around 1.27 s to compute one ECC scalar multiplication on the MICA2 (8 MHz) mote (Szczechowiak *et al*., 2008). One defect of the existing hybrid signcryption schemes is that they lack computational efficiency due to the use of pairing operations; thus, they are not applicable to security mechanisms for resource-limited environments, such as wireless sensor networks and ad hoc networks. Hence, it is important to devise a secure and efficient CLHS scheme suitable for such environments.

Up to now, most of the existing CLHS schemes are based on bilinear pairings. In this study, we construct a low-computation CLHS (LC-CLHS) scheme, which has the advantages of both ECC and CLHS. In the random oracle model, the new scheme is provably IND-CCA2 and sUF-CMA secure under the elliptic-curve computation Diffie-Hellman (ECCDH) assumption and elliptic curve discrete logarithm (ECDL) assumption. In comparison with the existing schemes, our scheme has a lower computational complexity. Its low-computation property makes it very attractive for applications in resource-limited environments.

2 Preliminaries

In this section, we briefly introduce elliptic-curve cryptography and several computational assumptions relevant to the cryptographic algorithm we devise.

2.1 Elliptic-curve cryptography

Let F_p denote the finite field of modulo of a large prime p. A non-singular elliptic curve E over the finite field F_p is defined via

$$y^2 = x^3 + ax + b \bmod p, \tag{1}$$

where a and b are two integers that are smaller than p and satisfy $4a^3+27b^2 \bmod p \neq 0$. Point $S(x, y)$ is an elliptic curve point if it satisfies the above equation, and point $Q(x, -y)$ is called the negative of S, i.e., $S=-Q$. Let $S(x_1, y_1)$ and $Q(x_2, y_2)$ denote two points in Eq. (1). Addition of two points in elliptic curve is defined as a line between the two points, and the intersection of the line in the elliptic curve. The negative of the intersection point is used as the result of the addition. The operation is denoted by $S+Q=R$, or

$$(x_1, y_1)+(x_2, y_2)=(x_3, y_3). \tag{2}$$

A base point P of elliptic curve E with order n should satisfy $nP=O$, where n is a prime and O is a point at infinity of E. The points of elliptic curve $E_p(a, b)$ together with a point O at infinity form an addition cyclic group G_p with order p, i.e.,

$$G_p = (x, y) : x, y \in F_p,\ (x, y) \in E_p(a, b) \cup O. \tag{3}$$

2.2 Computational assumptions

The ECDL problem is described as follows: Given two points B and C on an elliptic curve E with prime order n, where $B=bC$ ($b<n$), the ECDL problem is that it is difficult to determine the value of b.

Definition 1 The ECDL assumption holds if the advantage of any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ as defined below is negligible:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{ECDL}} = \Pr[A(C, B = bC) = b \mid b < n]. \tag{4}$$

ECCDH problem is described as follows: Let P be a base point of an elliptic curve E with prime order n. Given two points $S=aP$ and $Q=bP$ of the elliptic curve E, where a and b are unknown, the ECCDH problem is that it is computationally infeasible to determine another point $R=abP$.

Definition 2 The ECCDH assumption holds if the advantage of any PPT adversary $\mathcal{A}$ defined below is negligible:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{ECCDH}} = \Pr[\mathcal{A}(P, S=aP, Q=bP) = R \mid a, b < n]. \tag{5}$$

The elliptic curve decision Diffie-Hellman (ECDDH) problem is described as follows: Let P be a base point on an elliptic curve E with prime order n. Given (P, aP, bP, cP) for unknown a, b, $c<n$, the ECDDH problem is to decide whether $c=ab \bmod n$.

Definition 3 The ECDDH assumption holds if the advantage of any PPT adversary $\mathcal{A}$ defined below is negligible:

$$\begin{aligned}\mathrm{Adv}_{\mathcal{A}}^{\mathrm{ECDDH}} =& \mid \Pr[A(P, aP, bP, abP) = 1 \mid a, b < n] \\ & - \Pr[A(P, aP, bP, cP = 1) \mid a, b, c < n] \mid .\end{aligned} \tag{6}$$

3 Formal definition

3.1 Framework for the generic scheme

A generic LC-CLHS scheme is depicted by the following five PPT algorithms:

Setup is an initial algorithm run by KGC that takes a security parameter k as input, and outputs a master private key x along with a set of public system parameters ρ.

KeyGen is a key generation algorithm run by a user that takes ρ and an identity id_i as input, and outputs this identity's secret value x_i and its public key y_i.

PartialKeyGen is a partial key generation algorithm run by KGC and a user. PartialKeyGen takes ρ, a public key y_i on identity id_i, outputs this identity's partial public key u_i, and partial private key s_i as input.

It is apparent that this identity's full public key is $p_i=(u_i, y_i)$ and its full private key is $e_i=(s_i, x_i)$.

Signcrypt is a signcryption algorithm run by a sender that takes ρ, a message m, a sender's identity id_a together with a pair of public and private keys (e_a, p_a), and a recipient's identity id_b and public key p_b as input, and outputs a signcrypted text σ to the receiver. We write this as

$$\sigma = \mathrm{Signcrypt}(\rho, m, \mathrm{id}_a, \mathrm{id}_b, e_a, p_a, p_b). \tag{7}$$

Unsigncrypt is an unsigncryption algorithm run by a recipient that takes as input ρ, a signcrypted text σ, a sender's identity id_a together with public key p_a, and a recipient's identity id_b and a pair of public and private keys (e_b, p_b), and outputs a plaintext m or a symbol $\perp$ denoting failure. We write this as

$$m \text{ or } \perp = \mathrm{Unsigncrypt}(\rho, \sigma, \mathrm{id}_a, \mathrm{id}_b, e_b, p_a, p_b). \tag{8}$$

3.2 Security models

A generic LC-CLHS scheme should satisfy IND-CCA2 and sUF-CMA security. Moreover, an LC-CLHS scheme should resist the attacks of type I adversary $\mathcal{A}_{\mathrm{I}}$ and type II adversary $\mathcal{A}_{\mathrm{II}}$. Type I adversary is a common system user who has the ability to replace any user's public key in an adaptive fashion; however, it does not own the master private key. In contrast, type II adversary can access the master private key but cannot request public key replacement.

In the following security models, queries by a sender and receiver with the same identity are not allowed.

Let us first illustrate the IND-CCA2-I security model in terms of an interaction game named IND-LC-CLHS-CCA2-I between a challenger Γ and a type I adversary $\mathcal{A}_{\mathrm{I}}$.

IND-LC-CLHS-CCA2-I: At the beginning of the game, challenger Γ runs the initial algorithm to generate a set of system parameters ρ together with a master private key x. Γ keeps x to itself but delivers ρ to adversary $\mathcal{A}_{\mathrm{I}}$.

Phase 1: $\mathcal{A}_{\mathrm{I}}$ requests a polynomially bounded number of queries to Γ, and these queries are adaptive.

Request public key: $\mathcal{A}_{\mathrm{I}}$ supplies an identity id_i and asks for a public key for this identity. Γ returns this identity's public key p_i.

Secret value queries: $\mathcal{A}_{\mathrm{I}}$ supplies an identity id_i and asks for a secret value for this identity. Γ returns the secret value x_i if the public key on this identity has

not been replaced.

Partial private key queries: $\mathcal{A}_{\mathrm{I}}$ supplies an identity id_i and asks for a partial private key for this identity. Γ returns this identity's partial private key s_i to the adversary.

Replace public key: $\mathcal{A}_{\mathrm{I}}$ supplies an identity id_i and a new valid full public key to Γ, and replaces the original public key on this identity.

Signcryption queries: $\mathcal{A}_{\mathrm{I}}$ issues a signcryption query on a message m, a sender's identity id_a, and a receiver's id_b. Γ runs the signcryption algorithm to generate signcrypted text σ and delivers it to the adversary.

Unsigncryption queries: For an unsigncryption query on a signcrypted text σ, a sender's identity id_a, and a receiver's id_b, Γ delivers the result of unsigncryption to the adversary.

Challenge: At the end of Phase 1, the adversary outputs a pair of messages $\{m_0, m_1\}\in\{0, 1\}^l$ together with a pair of identities $\{\mathrm{id}_a^*, \mathrm{id}_b^*\}\in\{0, 1\}^*$ on which it wishes to be challenged. In Phase 1, the adversary cannot query the full private key on identity id_b^*. Γ selects a random bit θ from $\{0, 1\}$ to calculate

$$\sigma^* = \mathrm{Signcrypt}(\rho, m_\theta, \mathrm{id}_a^*, \mathrm{id}_b^*, e_a^*, p_a^*, p_b^*), \quad (9)$$

and delivers it to the adversary, where (e_a^*, p_a^*, p_b^*) are obtained from the query-answer list.

Phase 2: $\mathcal{A}_{\mathrm{I}}$ issues a series of queries again in an adaptive manner as in Phase 1, and the challenger responds in the same way as in Phase 1. In this phase, the adversary cannot make the full private key query on identity id_b^*; besides, it cannot make an unsigncryption query on challenge ciphertext σ^*.

Finally, adversary $\mathcal{A}_{\mathrm{I}}$ terminates the game by outputting a guess θ', and wins the above IND-LC-CLHS-CCA2-I game if $\theta'=\theta$. The advantage of the adversary is defined as follows:

$$\mathrm{Adv}_{\mathcal{A}_{\mathrm{I}}}^{\text{IND-ECC-CLHS-CCA2-I}} = |\Pr[\theta'=\theta]-1/2|. \quad (10)$$

Let us introduce the IND-CCA2-II security model in terms of an interactive game named IND-LC-CLHS-CCA2-II between a challenger Γ and a type II adversary $\mathcal{A}_{\mathrm{II}}$.

IND-LC-CLHS-CCA2-II: At the start of the game, challenger Γ runs the initial algorithm to generate a master private key x and a set of system parameters ρ. Γ sends (ρ, x) to adversary $\mathcal{A}_{\mathrm{II}}$. Bear in mind that the master private key x is known to the adversary.

Phase 1: $\mathcal{A}_{\mathrm{II}}$ requests a polynomially bounded number of queries to Γ. The current query relies on the answers to the previous queries.

Request public key: $\mathcal{A}_{\mathrm{II}}$ issues a public key query on identity id_i. Γ delivers id_i's public key p_i to the adversary.

Full private key queries: $\mathcal{A}_{\mathrm{II}}$ issues a full private key query on identity id_i. Γ returns this identity's full private key $e_i=(x_i, s_i)$.

Signcryption queries: $\mathcal{A}_{\mathrm{II}}$ makes a signcryption query on a message m, a sender's identity id_a, and a receiver's id_b. Γ delivers the result of signcryption to the adversary.

Unsigncryption queries: For an unsigncryption query on a ciphertext σ, a sender's identity id_a, and a receiver's id_b, Γ delivers the result of unsigncryption to the adversary.

Challenge. At the end of Phase 1, the adversary outputs two equal-length messages, m_0 and m_1, together with two identities id_a^* and id_b^* on which it wishes to be challenged. In Phase 1, the adversary cannot query the secret value on identity id_b^*. Γ chooses a bit θ from $\{0, 1\}$ and returns

$$\sigma^* = \mathrm{Signcrypt}(\rho, m_\theta, \mathrm{id}_a^*, \mathrm{id}_b^*, e_a^*, p_a^*, p_b^*) \quad (11)$$

to the adversary, where (e_a^*, p_a^*, p_b^*) are obtained from the query-answer list.

Phase 2: $\mathcal{A}_{\mathrm{II}}$ issues a sequence of queries again in an adaptive fashion as in Phase 1, and the challenger answers in the same way as in Phase 1. In Phase 2, the adversary cannot query the secret value on identity id_b^*; besides, it cannot make an unsigncryption query on σ^*.

At the end, the adversary outputs a guess θ' and wins the above IND-LC-CLHS-CCA2-II game if $\theta'=\theta$. We define the advantage of the adversary as

$$\mathrm{Adv}_{\mathcal{A}_{\mathrm{II}}}^{\text{IND-ECC-CLHS-CCA2-II}} = |\Pr[\theta'=\theta]-1/2|. \quad (12)$$

Definition 4 (Confidentiality) An LC-CLHS scheme is ND-CCA2-I secure if there is no PPT adversary $\mathcal{A}_{\mathrm{I}}$ that wins the above IND-LC-CLHS-CCA2-I game with non-negligible advantage. Likewise, an LC-CLHS scheme is IND-CCA2-II secure if no PPT adversary $\mathcal{A}_{\mathrm{II}}$ wins the above IND-LC-CLHS-CCA2-II game with non-negligible advantage. Thus, an LC-CLHS scheme is IND-CCA2 secure if it is both IND-CCA2-I and IND-CCA2-II secure.

Let us describe the sUF-CMA-I security model in terms of the sUF-LC-CLHS-CMA-I game between a challenger Γ and a type I adversary $\mathcal{A}_{\mathrm{I}}$.

sUF-LC-CLHS-CMA-I: At the beginning of the game, challenger Γ runs the initial algorithm to obtain a set of system parameters ρ and a master private key x. Γ keeps x secret but delivers ρ to adversary $\mathcal{A}_{\mathrm{I}}$.

Train: $\mathcal{A}_{\mathrm{I}}$ issues a sequence of queries in an adaptive way just like Phase 1 in the IND-LC-CLHS-CCA2-I game for IND-CCA2-I security. Γ responds in the same way as in Phase 1 in the IND-LC-CLHS-CCA2-I game for IND-CCA2-I security.

Forgery: At the end of the training phase, $\mathcal{A}_{\mathrm{I}}$ outputs a forgery ciphertext σ^* on the identities (id_a^*, id_b^*) to Γ. In the training phase, $\mathcal{A}_{\mathrm{I}}$ cannot query the secret value and partial private key on identity id_a^*; besides, σ^* should not be the response for any signcryption query made by $\mathcal{A}_{\mathrm{I}}$. If the result of unsigncryption is valid, adversary $\mathcal{A}_{\mathrm{I}}$ wins the sUF-LC-CLHS-CMA-I game.

We define the advantage of the adversary as the probability of it winning the sUF-LC-CLHS-CMA-I game.

Let us show the sUF-CMA-II security model in terms of the sUF-LC-CLHS-CMA-II game between a challenger Γ and a type II adversary $\mathcal{A}_{\mathrm{II}}$.

sUF-LC-CLHS-CMA-II: At the start of the game, Γ runs the initial algorithm to obtain a master private key x and a set of system parameters ρ. At the end, Γ delivers (ρ, x) to $\mathcal{A}_{\mathrm{II}}$.

Train: $\mathcal{A}_{\mathrm{II}}$ issues a series of queries in an adaptive way as in Phase 1 in the IND-LC-CLHS-CCA2-II game for IND-CCA2-II security. The challenger answers in the same way as in Phase 1 in the IND-LC-CLHS-CCA2-II game for IND-CCA2-II security.

Forgery: At the end of the training phase, $\mathcal{A}_{\mathrm{II}}$ outputs a forgery (id_a^*, id_b^*, σ^*) to the challenger. In the training phase, the adversary cannot query the secret value on identity id_a^*; besides, σ^* should not be the response for any signcryption query made by the adversary. If the result of unsigncryption is not symbol $\perp$, the adversary wins the sUF-LC-CLHS-CMA-II game.

The advantage of the adversary is defined as the probability of it winning the sUF-LC-CLHS-CMA-II game.

Definition 5 (Unforgeability) An LC-CLHS scheme is sUF-CMA-I secure if no PPT adversary $\mathcal{A}_{\mathrm{I}}$ wins the above sUF-LC-CLHS-CMA-I game with non-negligible advantage. Likewise, an LC-CLHS scheme is sUF-CMA-II secure if no PPT adversary $\mathcal{A}_{\mathrm{II}}$ wins the above sUF-LC-CLHS-CMA-II game with non-negligible advantage. Thus, an LC-CLHS scheme is sUF-CMA secure if it is both sUF-CMA-I and sUF-CMA-II secure.

4 An example of the LC-CLHS scheme

In this section, we construct a concrete example of the LC-CLHS scheme, which is suitable for wireless sensor networks and ad hoc networks. It allows a sender to deliver a signed and encrypted message of arbitrary length to a receiver so that the receiver can decrypt and verify them.

4.1 Setup

On input security parameter k, KGC carries out Algorithm 1 to generate a master private key x and a set of system parameters ρ.

4.2 KeyGen

On input ρ and some user's identity id_i, this key generation algorithm generates this user's secret value and public key.

A sender with identity id_a selects a secret value $x_a \in [1, n)$ as his/her private key and calculates his/her public key $y_a = x_a P$.

In the same way, a receiver with identity id_b chooses a secret value $x_b \in [1, n)$ as his/her private key and calculates his/her public key $y_b = x_b P$.

Algorithm 1 Setup

Input: security parameter k.

Output: master private key x and a set of system parameters ρ.

1 Pick a large prime p and an elliptic curve E defined over the finite field F_p;
2 Pick a base point P of elliptic curve E with prime order n, where P is also the generator of additive cyclic group G_p with prime order p (Hwang *et al*., 2005);
3 Pick the master private key $x\in[1, n)$ uniformly at random and calculate the system public key $y=xP$;
4 Pick four cryptographic hash functions: H_1: $\{0, 1\}^*\times G_p\rightarrow Z_p^*$, H_2: $G_p\times G_p\times G_p\rightarrow\{0, 1\}^l$, H_3: $\{0, 1\}^{*2}\times\{0, 1\}^l\times G_p{}^6\rightarrow Z_p^*$, and H_4: $G_p{}^3\rightarrow G_p$, where l is the symmetric key length of a DEM (Yu and Yang, 2015b);
5 Publish $\rho=(F_p, E, p, G_p, P, y, l, H_1-H_4)$ but keep the master private key x secret.

4.3 PartialKeyGen

In this phase, KGC runs this partial key generation to generate some user's partial private key and partial public key.

KGC randomly picks $\upsilon_a\in[1, n)$ to calculate $s_a=\upsilon_a+x\cdot H_1(\mathrm{id}_a, y_a) \bmod n$ and $u_a=\upsilon_a P$, where s_a is the sender's partial private key and u_a is the sender's partial public key. It is clear that the sender's full public key and the sender's full private key are $p_a=(u_a, y_a)$ and $e_a=(s_a, x_a)$, respectively. After that, KGC calculates $Y_a=s_aP+\upsilon_a y_a$ and sends (s_a, u_a, Y_a) to the sender, who can both verify the legitimacy of the partial private key s_a and check the validity of the partial public key u_a using the following two verification equalities:

$$s_aP=u_a+H_1(\mathrm{id}_a, y_a)y, \tag{13}$$

$$s_aP=Y_a-x_au_a. \tag{14}$$

Likewise, KGC chooses a random $\upsilon_b\in[1, n)$ to compute $s_b=\upsilon_b+x\cdot H_1(\mathrm{id}_b, y_b) \bmod n$ and $u_b=\upsilon_b P$, where s_b is the receiver's partial private key and u_b is the receiver's partial public key. It is clear that the receiver's full public key and full private key are $p_b=(u_b, y_b)$ and $e_b=(s_b, x_b)$, respectively. KGC then sets $Y_b=s_bP+\upsilon_b y_b$ and delivers (s_b, u_b, Y_b) to the receiver, who can both verify the legitimacy of the partial private key s_b and check the validity of the partial public key u_b using the verification equalities as follows:

$$s_bP=u_b+H_1(\mathrm{id}_b, y_b)y, \tag{15}$$

$$s_bP=Y_b-x_bu_b. \tag{16}$$

4.4 Signcrypt

On input ρ, message m, a sender's identity id_a together with a pair of public and private keys (e_a, p_a), and a receiver's identity id_b and public key p_b, the sender runs Algorithm 2 to generate a signcrypted text σ, and sends it to the receiver.

Algorithm 2 Signcrypt

Input: system parameters ρ, message m, a sender's identity id_a together with a pair of public and private keys (e_a, p_a), and a receiver's identity id_b and public key p_b.

Output: master private key x and a set of system parameters ρ.

1 Calculate $\sigma_1=rP$, where r is chosen randomly from $[1, n)$;
2 Calculate $t=r(u_b+H_1(\mathrm{id}_b, y_b)y)$;
3 Calculate $\kappa=H_2(t, ry_b, \sigma_1)$;
4 Make use of the symmetric encryption algorithm to calculate $\sigma_2=\mathrm{DEM.Enc}(\kappa, m)$, where κ is obtained from Step 3;
5 Calculate $\gamma=H_4(t, ry_b, \sigma_1)$;
6 Calculate $u=r\gamma$ and $\sigma_3=(s_a+x_a)\gamma$;
7 Calculate $\sigma_4=H_3(\mathrm{id}_a, \mathrm{id}_b, m, u, p_a, p_b, \sigma_1)$;
8 Calculate $\sigma_5=r+(s_a+x_a)\sigma_4 \bmod n$;
9 Send the ciphertext $\sigma\leftarrow(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ to the receiver.

4.5 Unsigncrypt

After the recipient obtains the signcrypted text $\sigma\leftarrow(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ from the sender, he/she recovers message m by running the symmetric decryption algorithm and simultaneously verifies the signature $(\sigma_1, \sigma_3, \sigma_4, \sigma_5)$ by Algorithm 3.

Algorithm 3 Unsigncrypt

1 Calculate $t=s_b\sigma_1$;
2 Calculate $\kappa=H_2(t, x_b\sigma_1, \sigma_1)$;
3 Make use of the symmetric decryption algorithm to recover $m=\mathrm{DEM.Dec}(\kappa, \sigma_2)$, where κ is obtained from Step 2;
4 Calculate $\gamma=H_4(t, x_b\sigma_1, \sigma_1)$;
5 Calculate $u=\sigma_5\gamma-\sigma_3\sigma_4$;
6 Calculate $\sigma_4'=H_3(\mathrm{id}_a, \mathrm{id}_b, m, u, p_a, p_b, \sigma_1)$;
7 **if** $\sigma_4'=\sigma_4$
8 The recipient accepts the valid signcrypted text received from the sender;
9 **else**
10 The recipient outputs an error symbol $\perp$;
11 **end if**

It is easy for us to verify that Algorithm 2 is consistent with Algorithm 3 using two equalities as follows:

$$\begin{aligned}\kappa &= H_2(s_b\sigma_1, x_b\sigma_1, \sigma_1)\\ &= H_2((\upsilon_b + x\cdot H_1(\mathrm{id}_b, y_b))rP, x_b\sigma_1, \sigma_1)\\ &= H_2(ru_b + x\cdot H_1(\mathrm{id}_b, y_b)\cdot rP, x_b rP, \sigma_1)\\ &= H_2(r(u_b + H_1(\mathrm{id}_b, y_b)y), ry_b, \sigma_1)\\ &= H_2(t, ry_b, \sigma_1),\end{aligned} \tag{17}$$

$$\begin{aligned}u &= \sigma_5\gamma - \sigma_3\sigma_4\\ &= (r + (s_a + x_a)\sigma_4)\gamma - (s_a + x_a)\gamma\sigma_4\\ &= r\gamma + (s_a + x_a)\sigma_4\gamma - (s_a + x_a)\gamma\sigma_4\\ &= r\gamma.\end{aligned} \tag{18}$$

5 Security analysis

Theorem 1 If a PPT adversary $\mathcal{A}_\mathrm{I}$ can break the IND-CCA2-I security of our LC-CLHS scheme with an advantage ε, issuing q_i (i=1, 2, 3, 4) queries to oracle H_i, q_p partial private key queries, q_s secret value queries, and q_r public key replacement, then there exists a challenger Γ that can make use of adversary $\mathcal{A}_\mathrm{I}$ to solve the ECCDH problem with advantage ε', where

$$\varepsilon' \geq \varepsilon\cdot\frac{1}{e}\cdot\frac{1}{q_2}\cdot\frac{1}{q_\mathrm{p}+q_\mathrm{s}+q_\mathrm{r}}. \tag{19}$$

Proof Assume that challenger Γ receives a random instance (P, C_1=aP, C_2=bP) of the ECCDH problem and its aim is to determine abP. Bear in mind that a and b are unknown to Γ. To determine abP, Γ runs adversary $\mathcal{A}_\mathrm{I}$ as a subroutine and acts as its challenger in the game as follows.

At the start of the game, challenger Γ runs the initial algorithm to generate the system parameters ρ with y=C_1 and sends ρ to the adversary $\mathcal{A}_\mathrm{I}$.

Phase 1: $\mathcal{A}_\mathrm{I}$ issues a polynomially bounded number of queries to Γ as described below, and these queries are adaptive.

H_1 queries: $\mathcal{A}_\mathrm{I}$ makes a series of H_1 queries on identities of his/her choice. $\mathcal{A}_\mathrm{I}$ issues an H_1 query on identity id_i, Γ checks whether this query on this identity already exists in list L_1 which is initially empty. If there is a matching tuple, Γ returns l_i; otherwise, it returns an l_i chosen randomly from Z_p^* and stores tuple (id_i, l_i) in list L_1.

Assume that in the whole game, the adversary first makes the H_1 query with identity id_i before querying any other oracles with identity id_i as input.

H_2 queries: $\mathcal{A}_\mathrm{I}$ makes a new H_2 query, Γ checks whether this query already exists in list L_2 which is initially empty. If there is a matching tuple, Γ returns the symmetric key κ; otherwise, it returns a random symmetric key $\kappa\in\{0, 1\}^l$ and stores tuple (t, ry_b, σ_1, κ) in list L_2.

H_3 queries: $\mathcal{A}_\mathrm{I}$ makes a new H_3 query, Γ checks whether this query already exists in list L_3 which is initially empty. If there is such a tuple, Γ delivers σ_3 to $\mathcal{A}_\mathrm{I}$; otherwise, it returns a σ_3 chosen randomly from Z_p^* and stores the relevant tuple in list L_3.

H_4 queries: $\mathcal{A}_\mathrm{I}$ makes a new H_4 query, Γ checks whether there is a matching tuple in list L_4 which is initially empty. If there exists a relevant one, Γ delivers γ to $\mathcal{A}_\mathrm{I}$; otherwise, it returns $\gamma\in_R G_p$ and stores tuple (t, ry_b, σ_1, γ) in list L_4.

Request public key: Assume that $\mathcal{A}_\mathrm{I}$ has made the H_1 query before asking for a public key using identity id_i. Γ picks a random index τ from {1, 2, …, q_1}, where q_1 is the number of queries to H_1. It then fixes id_τ as the target identity for the challenge phase. Bear in mind that τ and id_τ are unknown to the adversary. Let δ denote the probability that id_i=id_τ, and it will be determined later. $\mathcal{A}_\mathrm{I}$ issues a public key query on identity id_i, and Γ responds to this query in the following way:

1. If this query is the τth query, the challenger calculates y_i=x_iP and u_i=(1–l_1)C_1, where x_i is chosen randomly from [1, n). After that, it returns the full public key p_i=(u_i, y_i) and stores (id_i, –, –, x_i, –, p_i) in list L_k which is initially empty. Bear in mind that list L_k is used to save the query-answer values relevant to the full public key and full private key.

2. If this query is not the τth query, Γ calculates y_i=x_iP and u_i=υ_iP–l_iC_1, where x_i and υ_i are two random values chosen from [1, n). It then returns the full public key p_i=(u_i, y_i) and stores (id_i, υ_i, –, x_i, –, p_i) in list L_k.

Secret value queries: $\mathcal{A}_{\mathrm{I}}$ issues a secret value query on identity id_i, and Γ fails and stops this query if $\mathrm{id}_i=\mathrm{id}_\tau$; otherwise, Γ returns the secret value x_i obtained from list L_k.

Partial private key queries: Assume that $\mathcal{A}_{\mathrm{I}}$ has made the H_1 query before Γ receives a partial private key query for identity id_i. $\mathcal{A}_{\mathrm{I}}$ submits a partial private key query on identity id_i, and Γ fails and stops this query if $\mathrm{id}_i=\mathrm{id}_\tau$; otherwise, Γ obtains υ_i from list L_k, to set $s_i=\upsilon_i$ as the partial private key and calculates $Y_i=s_iP+\upsilon_i y_i$. It is clear that $e_i=(s_i, x_i)$ is the full private key. It then returns (s_i, Y_i) to the adversary and updates list L_k with a new entry $(\mathrm{id}_i, \upsilon_i, s_i, x_i, e_i, p_i)$.

$\mathcal{A}_{\mathrm{I}}$ can verify the validity of the partial public key u_i and partial private key s_i by

$$s_iP=u_i+l_iC_1, \tag{20}$$

$$s_iP=Y_i-x_iu_i. \tag{21}$$

Replace public key: $\mathcal{A}_{\mathrm{I}}$ chooses an identity id_i and a new full public key $p_i'=(u_i', y_i')$, and its aim is to use p_i' to replace the public key p_i. Γ fails and stops this query if $\mathrm{id}_i=\mathrm{id}_\tau$; otherwise, it updates list L_k with $(\mathrm{id}_i, –, –, –, –, p_i')$.

Signcryption queries: Assume that the adversary has made the H_1 query and checked list L_k before Γ receives a signcryption query. $\mathcal{A}_{\mathrm{I}}$ issues a signcryption query on a message m, a sender's identity id_a, and a receiver's id_b. Γ returns a signcrypted text σ by running Algorithm 3 if $\mathrm{id}_a\neq\mathrm{id}_\tau$; otherwise, it calculates $\sigma_1=rP$ and $t=r_b\sigma_1$, and stores $(t, x_b\sigma_1, \sigma_1, \kappa)$ in list L_2, where r is chosen randomly from $[1, n)$. Γ continues to calculate

$$\kappa=H_2(t, x_b\sigma_1, \sigma_1), \tag{22}$$

$$\sigma_2=\mathrm{DEM.Enc}(\kappa, m), \tag{23}$$

$$\gamma=w(u_a+l_aC_1+y_a), \tag{24}$$

and stores $(t, ry_b, \sigma_1, \gamma)$ in list L_4, where w is chosen randomly from $[1, n)$. Γ then sets $u=r\gamma$ and calculates

$$\sigma_3=wd(u_a+l_aC_1+y_a), \tag{25}$$

$$\sigma_4=rd, \tag{26}$$

and stores $(\mathrm{id}_a, \mathrm{id}_b, m, u, p_a, p_b, \sigma_1, \sigma_4)$ in list L_4, where d is selected randomly from $[1, n)$. In the end, Γ calculates $\sigma_5=r+\sigma_4 d \bmod n$ and returns the signcrypted text $\sigma\leftarrow(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ to the adversary.

Unsigncryption queries: Assume that the adversary has made the H_3 and H_4 queries before Γ obtains an unsigncryption query. $\mathcal{A}_{\mathrm{I}}$ issues an unsigncryption query on a signcrypted text σ, a sender's identity id_a, and a receiver's id_b. Γ returns the result of Algorithm 2 if $\mathrm{id}_b\neq\mathrm{id}_\tau$; otherwise, it searches list $\mathcal{A}_{\mathrm{I}}$ to achieve (p_a, p_b), and goes through list $\mathcal{A}_{\mathrm{II}}$ to seek a tuple $(t, x_b\sigma_1, \sigma_1, \kappa)$ for different values of t, such that the ECDDH oracle returns 1 when queried on (y, σ_1, t), where x_b is from either the adversary or list $\mathcal{A}_{\mathrm{I}}$. In this case, Γ recovers $m=\mathrm{DEM.Dec}(\kappa, \sigma_2)$ and calculates

$$u=\sigma_5\gamma-\sigma_4\sigma_3, \tag{27}$$

$$\sigma_4'=H_3(\mathrm{id}_a, \mathrm{id}_b, m, u, p_a, p_b, \sigma_1). \tag{28}$$

Finally, Γ returns m if and only if $\sigma_4'=\sigma_4$; otherwise, it returns $\perp$.

Challenge: Assume that $\mathcal{A}_{\mathrm{I}}$ has made the H_1 query and retrieved list L_k before it issues a challenge query. At the end of the interaction of Phase 1, the adversary selects two equal-length messages, m_0 and m_1, together with a sender's identity id_a^* and a receiver's identity id_b^* on which it wishes to be challenged. In Phase 1, the adversary cannot query the full private key on identity id_b^*. Γ fails and stops this query if $\mathrm{id}_b^*\neq\mathrm{id}_\tau$; otherwise, it sets $\sigma_1^*=C_2$ and $\kappa_1=H_2(t^*, x_b^*\sigma_1^*, \sigma_1^*)$ and stores $(t^*, x_b^*\sigma_1^*, \sigma_1^*, \kappa_1)$ in list L_2, where t^* is chosen from G_p. After that, it selects a symmetric key κ_0 from the key space of our LC-CLHS scheme and a bit θ from $\{0, 1\}$, calculates $\sigma_2^*=\mathrm{DEM.Enc}(\kappa_\theta, m_\theta)$ and $\gamma^*=w^*C_2$, and stores $(t^*, x_b^*\sigma_1^*, \sigma_1^*, \gamma^*)$ in list L_4, where w^* is chosen randomly from $[1, n)$. It sets $u^*=\gamma^*$ and continues to calculate

$$\sigma_3^*=w^*(s_a^*+x_a^*)C_2, \tag{29}$$

$$\sigma_4^*=H_3(\mathrm{id}_a^*, \mathrm{id}_b^*, m_\theta, u^*, p_a^*, p_b^*, \sigma_1^*), \tag{30}$$

$$\sigma_5^*=1+\sigma_4^*(s_a^*+x_a^*) \bmod n, \tag{31}$$

and stores $(\mathrm{id}_a^*, \mathrm{id}_b^*, m_\theta, u^*, p_a^*, p_b^*, \sigma_1^*, \sigma_4^*)$ in list L_3. In the end, it returns the challenge ciphertext $\sigma^*\leftarrow(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*)$ to $\mathcal{A}_{\mathrm{I}}$.

Phase 2: $\mathcal{A}_{\mathrm{I}}$ issues a series of oracle queries in an adaptive manner again as in Phase 1, and the challenger responds as in Phase 1. In Phase 2, the adversary cannot query the full private key on identity id_b^*; besides, it cannot make an unsigncryption query on the challenge ciphertext σ^*.

Guess: In accordance with the assumption at the beginning of the proof, we know that the adversary has the ability to break the IND-CCA2-I security of our LC-CLHS scheme. Hence, the adversary should have made the H_2 racle query with $(t^*, x_b^*\sigma_1^*, \sigma_1^*)$ as inputs. If the adversary has made q_2 queries to the H_2 oracle, then there must be q_2 tuples stored in list L_2, and one t^* among q_2 values in list L_2 should be the solution for the ECCDH problem instance.

Now, the challenger picks one t^* uniformly at random from q_2 values stored in list L_2, and outputs it as the solution of the ECCDH problem instance:

$$\begin{aligned} t^* &= r_b^*\sigma_1^* = b(u_b^* + l_b^* C_1) \\ &= b[(1-l_b^*)C_1 + l_b^* C_1] = abP. \end{aligned} \tag{32}$$

Let us assess the probability of the challenger determining the solution of the ECCDH problem instance.

According to the above game, we know that the probability of the challenger not failing and stopping the game in the first or second phase is $\delta^{q_{\mathrm{p}}+q_{\mathrm{s}}+q_{\mathrm{r}}}$, and the probability of the challenger not terminating the game during the challenge phase is $(1-\delta)$. Thus, the probability of the challenger not aborting the execution of the game is $\delta^{q_{\mathrm{p}}+q_{\mathrm{s}}+q_{\mathrm{r}}}(1-\delta)$. This value is maximized at

$$\delta = 1 - \frac{1}{1+q_{\mathrm{p}}+q_{\mathrm{s}}+q_{\mathrm{r}}}. \tag{33}$$

Referring to the probability analysis in an identity-based hybrid signcryption scheme (Yu and Yang, 2015a), we obtain that the probability of the challenger not aborting the execution of the game is at least

$$\frac{1}{e}\cdot\frac{1}{q_{\mathrm{p}}+q_{\mathrm{s}}+q_{\mathrm{r}}}. \tag{34}$$

On the other hand, the probability that the challenger chooses t^* uniformly from list L_2 is $1/q_2$; therefore, probability ε' of solving the ECCDH problem is at least

$$\varepsilon' = \varepsilon\cdot\frac{1}{e}\cdot\frac{1}{q_2}\cdot\frac{1}{q_{\mathrm{p}}+q_{\mathrm{s}}+q_{\mathrm{r}}}. \tag{35}$$

Theorem 2 If a PPT adversary $\mathcal{A}_{\mathrm{II}}$ can break the IND-CCA2-II security of our LC-CLHS scheme with an advantage ε, making q_i queries to H_i (i=1, 2, 3, 4) and q_{f} full private key queries, then there exists a challenger Γ that can use adversary $\mathcal{A}_{\mathrm{II}}$ to solve the ECCDH problem with advantage ε', where

$$\varepsilon' \geq \varepsilon\cdot\frac{1}{e}\cdot\frac{1}{q_2}\cdot\frac{1}{q_{\mathrm{f}}}. \tag{36}$$

Proof Let us suppose that challenger Γ receives a random instance (P, C_1=aP, C_2=bP) of the ECCDH problem, and its aim is to calculate abP. Bear in mind that Γ does not know the values of a and b. In the game, Γ runs adversary $\mathcal{A}_{\mathrm{II}}$ as a subroutine and acts as its challenger.

At the start of the game, challenger Γ runs the initial algorithm to generate the system parameter ρ with y=xP and sends (ρ, x) to adversary $\mathcal{A}_{\mathrm{II}}$.

Phase 1: $\mathcal{A}_{\mathrm{II}}$ requests a polynomially bounded number of queries described below, and the current query depends on the answers to the previous queries. Moreover, the H_1–H_4 queries are identical to those in Phase 1 in Theorem 1.

Request public key: Assume that the adversary has made the H_1 query before it asks for a public key using identity id_i. Γ selects a random index τ from {1, 2, …, q_1} and fixes id_τ as the target identity for the challenge phase. Bear in mind that τ and id_τ are not known to the adversary. Let δ be the probability that id_i=id_τ. δ will be determined later. Upon receiving a public key query on identity id_i, Γ responds to this query in the following ways:

1. If this query is the τth query, the challenger calculates u_i=$\upsilon_i P$ and sets y_i=C_1, where υ_i is selected randomly from [1, n]. After that, it returns the full public key p_i=(u_i, y_i) and stores (id_i, υ_i, –, –, p_i) in list L_k which is initially empty. Note that list L_k is

employed to save the query-answer values relevant to the full public key and full private key.

2. If this query is not the τth query, the challenger calculates $u_i=\upsilon_i P$ and $y_i=x_i P$, where υ_i and x_i are two random values chosen from [1, n]. It then returns the full public key $p_i=(u_i, y_i)$ and stores (id_i, υ_i, x_i, –, p_i) in list L_k.

Full private key queries: Assume that the adversary has made the H_1 query and searched list L_k before it issues a partial private key using identity id_i. Upon receiving a partial private key query on identity id_i, Γ responds to this query in the following way:

1. If this query is the τth query, it fails and stops this query.

2. If this query is not the τth query, it calculates $s_i=\upsilon_i+l_i x \bmod n$ and $Y_i=s_i P+\upsilon_i y_i$. It then returns the full private key $e_i=(s_i, x_i)$ to the adversary and updates list L_k with a new entry (id_i, υ_i, s_i, e_i, p_i).

$\mathcal{A}_{\mathrm{II}}$ can verify the validity of the partial public key u_i and partial private key s_i using

$$s_i P = u_i + l_i y, \tag{37}$$

$$s_i P = Y_i - x_i u_i. \tag{38}$$

Signcryption queries: Assume that the adversary has made the H_1 query and retrieved list L_k before it asks for a signcryption query. Upon receiving a signcryption query on a message m, a sender's identity id_a, and a receiver's identity id_b, the challenger responds to this query in the following way:

1. If $\mathrm{id}_a \neq \mathrm{id}_\tau$, it runs Algorithm 2 in a normal way and returns a signcrypted text σ.

2. If $\mathrm{id}_a=\mathrm{id}_\tau$, it calculates $\sigma_1=rP$, $t=s_b\sigma_1$, and $\kappa=H_2(t, x_b\sigma_1, \sigma_1)$, and stores ($t$, $x_b\sigma_1$, σ_1, κ) in list L_2, where r is chosen randomly from [1, n). It continues to calculate

$$\sigma_2 = \mathrm{DEM.Enc}(\kappa, m), \tag{39}$$

$$\gamma = w(u_a + l_a y + C_1), \tag{40}$$

and stores (t, ry_b, σ_1, γ) in list L_2, where w is chosen randomly from [1, n). It sets $u=r\gamma$ and continues to calculate

$$\sigma_3 = wd(u_a + l_a y + C_1), \tag{41}$$

$$\sigma_4 = rd, \tag{42}$$

and stores (id_a, id_b, m, u, p_a, p_b, σ_1, σ_4) in list L_3, where d is chosen randomly from [1, n). Finally, it calculates $\sigma_5=r+\sigma_4 d \bmod n$ and returns the signcrypted text $\sigma \leftarrow (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ to the adversary.

Unsigncryption queries: Suppose that the adversary has queried the H_3 and H_4 oracles before it asks for an unsigncryption query. For an unsigncryption query on a ciphertext σ, a sender's identity id_a, and a receiver's id_b, Γ responds to this query in the following way:

1. If $\mathrm{id}_b \neq \mathrm{id}_\tau$, the challenger runs the result of Algorithm 3.

2. If $\mathrm{id}_b=\mathrm{id}_\tau$, the challenger searches list L_k to obtain (s_b, p_a, p_b) and calculates $t=s_b\sigma_1$. It then looks over list L_2 to search a tuple (t, υ, σ_1, κ) for different values of υ, such that the ECDDH oracle returns 1 when queried on (y_b, σ_1, υ). If this case occurs, it recovers $m=\mathrm{DEM.Dec}(\kappa, \sigma_2)$ and calculates

$$u = \sigma_5 \gamma - \sigma_4 \sigma_3, \tag{43}$$

$$\sigma_4' = H_3(\mathrm{id}_a, \mathrm{id}_b, m, u, p_a, p_b, \sigma_1). \tag{44}$$

Finally, it returns m if and only if $\sigma_4'=\sigma_4$; otherwise, it returns $\perp$.

Challenge: At the end of the interaction of Phase 1, $\mathcal{A}_{\mathrm{II}}$ chooses two messages, m_0 and m_1, of equal length together with a sender's identity id_a^* and a recipient's identity id_b^* on which it wishes to be challenged. In Phase 1, the adversary cannot query the full private key on identity id_b^*. Assume that the adversary has made the H_1 query and retrieved list $\mathcal{A}_{\mathrm{I}}$ before it asks for a challenge query. In this challenge phase, Γ fails and stops this query if $\mathrm{id}_b^* \neq \mathrm{id}_\tau$; otherwise, it sets $\sigma_1^*=C_2$ and $t^*=r_b^* C_2$, calculates $\kappa_1=H_2(t^*, \eta^*, \sigma_1^*)$, and then stores ($t^*$, η^*, σ_1^*, κ_1) in list L_2, where η^* is chosen from G_p. It chooses κ_0 from the key space of our LC-CLHS scheme and a random bit θ from {0, 1} to calculate $\sigma_2^*=\mathrm{DEM.Enc}(\kappa_\theta, m_\theta)$ and $\gamma^*=w^* C_2$, and stores (t^*, η^*, σ_1^*, γ^*) in list L_4, where w^* is chosen randomly from [1, n). Γ sets $u^*=\gamma^*$ and continues to calculate

$$\sigma_3^* = w^*(s_a^* + x_a^*) C_2, \tag{45}$$

$$\sigma_4^* = H_3(\mathrm{id}_a^*, \mathrm{id}_b^*, m_\theta, u^*, p_a^*, p_b^*, \sigma_1^*), \tag{46}$$

and stores (id_a^*, id_b^*, m_θ, u^*, p_a^*, p_b^*, σ_1^*, σ_4^*) in list L_3.

Finally, it calculates $\sigma_5^*=1+\sigma_3^*(s_a^*+x_a^*) \bmod n$ and delivers $\sigma^* \leftarrow (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*)$ to $\mathcal{A}_{\mathrm{II}}$.

Phase 2: $\mathcal{A}_{\mathrm{II}}$ issues a series of oracle queries again in an adaptive way as in Phase 1. Γ answers as in Phase 1. In this phase, the adversary cannot make the secret value query on identity id_b^*; besides, it cannot make an unsigncryption query on σ^*.

Guess: As indicated by the above game, we know that the adversary can break the IND-CCA2-II security of our LC-CLHS scheme. Then the adversary should have made the H_2 random oracle query with $(t^*, \eta^*, \sigma_1^*)$ as inputs. If the adversary has made q_2 queries to the H_2 random oracle, list L_2 must have q_2 tuples. One η^* among q_2 values in list L_2 is the solution of the ECCDH problem instance. Γ picks one η^* value uniformly at random from list L_2 and outputs it as the solution of the ECCDH problem instance:

$$\eta^* = x_b^* \sigma_1^* = x_b^* C_2 = abP. \tag{47}$$

Let us assess the probability that the challenger solves the ECCDH problem in the above game.

The probability of the challenger not terminating the game in Phase 1 or 2 is $\delta^{q_{\mathrm{f}}}$, and the probability of the challenger not terminating the game during the challenge phase is $(1-\delta)$. Hence, the probability that the challenger does not abort the execution of the game is $\delta^{q_{\mathrm{f}}}$, and

$$\max\{\delta^{q_{\mathrm{f}}}\} \le \delta = 1 - \frac{1}{1+q_{\mathrm{f}}}. \tag{48}$$

Referring to the probability analysis in an identity-based hybrid signcryption scheme (Yu and Yang, 2015a), we can obtain that the probability of the challenger not aborting the execution of the game is at least

$$\frac{1}{e} \cdot \frac{1}{q_{\mathrm{f}}}. \tag{49}$$

On the other hand, the probability that the challenger chooses η^* uniformly at random from list L_2 as the solution of the ECCDH problem instance is $1/q_2$; therefore, probability ε' of solving the ECCDH problem satisfies

$$\varepsilon' \ge \varepsilon \cdot \frac{1}{e} \cdot \frac{1}{q_2} \cdot \frac{1}{q_{\mathrm{f}}}. \tag{50}$$

Theorem 3 If a PPT adversary $\mathcal{A}_{\mathrm{I}}$ has an advantage ε against the sUF-CMA-I security of the LC-CLHS scheme, issuing q_i queries to H_i (i=1, 2, 3, 4), q_{p} partial private key queries, q_{s} secret value queries, and q_{r} public key replacement, then there exists an algorithm Γ that can solve the ECDL problem with an advantage ε', where

$$\varepsilon' \ge \varepsilon \cdot \frac{1}{e} \cdot \frac{1}{q_{\mathrm{p}} + q_{\mathrm{s}} + q_{\mathrm{r}}}. \tag{51}$$

Proof Assume that challenger Γ receives a random instance $(P, C_1=aP)$ of the ECDL problem and its goal is to determine a. Bear in mind that a is unknown to the challenger. In the game, Γ runs adversary $\mathcal{A}_{\mathrm{I}}$ as a subroutine and attempts to use $\mathcal{A}_{\mathrm{I}}$ to obtain the solution of the ECDL problem instance.

At the beginning of the game, Γ runs the initial algorithm with a security parameter k to generate the system parameters ρ with $y=C_1$ and delivers ρ to adversary $\mathcal{A}_{\mathrm{I}}$.

Train: In this phase, $\mathcal{A}_{\mathrm{I}}$ makes a series of oracle queries as those in an adaptive fashion. Γ answers these queries as in Phase 1 in Theorem 1.

Forgery: At the end of the training phase, $\mathcal{A}_{\mathrm{I}}$ outputs a forgery $(\sigma^*, \mathrm{id}_a^*, \mathrm{id}_b^*)$ to the challenger. In the training phase, the adversary cannot query the secret value and partial private key on identity id_a^*; besides, σ^* should not be the response for any signcryption queries by the adversary. Γ fails and stops the game if $\mathrm{id}_a^* \neq \mathrm{id}_r$; otherwise, it uses the replay technique to generate another valid ciphertext $\sigma^{**} \leftarrow (\sigma_1^*, \sigma_2^{**}, \sigma_3^{**}, \sigma_4^{**}, \sigma_5^{**})$. Γ obtains the solution of the ECDL problem instance by using the forking lemma:

$$\begin{cases} \sigma_1^* = \sigma_5^* - \sigma_4^*(\upsilon_a^* + a + x_a^*), \\ \sigma_1^* = \sigma_5^{**} - \sigma_4^{**}(\upsilon_a^{**} + a + x_a^{**}), \end{cases} \tag{52}$$

$$a = \frac{\sigma_4^* - \sigma_4^{**} + \sigma_3^{**}(\upsilon_a^{**} + x_a^{**}) - \sigma_3^*(\upsilon_a^* + x_a^*)}{\sigma_3^* - \sigma_3^{**}}. \tag{53}$$

In the following, we estimate the probability that

the challenger obtains the solution of the ECDL problem instance.

In accordance with the probability analysis in Theorem 1, we can obtain that the probability of the challenger not aborting the execution of the game is at least

$$\frac{1}{e}\cdot\frac{1}{q_{\mathrm{p}}+q_{\mathrm{s}}+q_{\mathrm{r}}}. \tag{54}$$

On the basis of the above analysis, we can obtain that probability ε' of the challenger solving the ECDL problem satisfies

$$\varepsilon'\geq\varepsilon\cdot\frac{1}{e}\cdot\frac{1}{q_{\mathrm{p}}+q_{\mathrm{s}}+q_{\mathrm{r}}}. \tag{55}$$

Theorem 4 If a PPT adversary $\mathcal{A}_{\mathrm{II}}$ has an advantage ε against the sUF-CMA-II security of the LC-CLHS scheme, making q_i queries to H_i (i=1, 2, 3, 4) and q_{f} full private key queries, then there exists an challenger Γ that can solve the ECDL problem with an advantage ε', where

$$\varepsilon'\geq\varepsilon\cdot\frac{1}{e}\cdot\frac{1}{q_{\mathrm{f}}}. \tag{56}$$

Proof Assume that challenger Γ receives a random instance (P, C_1=aP) of the ECDL problem and its aim is to determine a. Bear in mind that a is unknown to the challenger. In the game, Γ runs adversary $\mathcal{A}_{\mathrm{II}}$ as a subroutine and attempts to make use of $\mathcal{A}_{\mathrm{II}}$ to obtain the solution of the ECDL problem instance.

At the beginning of the game, Γ runs the initial algorithm with security parameter k to obtain system parameters ρ with y=xP and delivers (ρ, x) to $\mathcal{A}_{\mathrm{II}}$.

Train: $\mathcal{A}_{\mathrm{II}}$ issues a series of oracle queries in an adaptive manner as those in Phase 1 in Theorem 2. All answers to these queries are identical to those in Phase 1 in Theorem 2.

Forgery: At the end of the training phase, the adversary outputs a forgery (σ^*, $\mathrm{id}_a{}^*$, $\mathrm{id}_b{}^*$). In the training phase, the adversary cannot query the secret value on identity $\mathrm{id}_a{}^*$; besides, σ^* should not be the response for any signcryption queries by the adversary. Γ fails and stops the game if $\mathrm{id}_a{}^*\neq\mathrm{id}_\tau$; otherwise, it uses the replay technique to generate another valid ciphertext $\sigma^{**}\leftarrow(\sigma_1{}^*, \sigma_2{}^{**}, \sigma_3{}^{**}, \sigma_4{}^{**}, \sigma_5{}^{**})$. Γ obtains the solution for the ECDL problem instance by using the forking lemma:

$$\begin{cases}\sigma_1{}^*=\sigma_5{}^*-\sigma_4{}^*(\upsilon_a{}^*+x+a),\\ \sigma_1{}^*=\sigma_5{}^{**}-\sigma_4{}^{**}(\upsilon_a{}^{**}+x+a),\end{cases} \tag{57}$$

$$a=\frac{\sigma_4{}^*-\sigma_4{}^{**}+\sigma_3{}^{**}(\upsilon_a{}^{**}+x)-\sigma_3{}^*(\upsilon_a{}^*+x)}{\sigma_3{}^*-\sigma_3{}^{**}}. \tag{58}$$

In the following, we estimate the probability that the challenger solves the ECDL problem.

In accordance with the probability analysis in Theorem 2, the probability of the challenger not terminating the game is at least

$$\frac{1}{e}\cdot\frac{1}{q_{\mathrm{f}}}. \tag{59}$$

On the basis of the above analysis, we can determine that probability ε' of the challenger solving the ECDL problem satisfies

$$\varepsilon'\geq\varepsilon\cdot\frac{1}{e}\cdot\frac{1}{q_{\mathrm{f}}}. \tag{60}$$

As described above, if probability ε is not negligible, then the challenger has the ability to solve the ECDL or ECCDH problem with non-negligible probability ε'. Because the challenger runs the adversary as a subroutine, this implies that the challenger can make use of the adversary to solve the ECDL or ECCDH problem with non-negligible probability ε'. It contradicts the initial ECDL or ECCDH assumption. Thus, we conclude that, given the assumption regarding the ECDL or ECCDH problem, no PPT adversary can succeed with probability ε that is not negligible.

6 Comparison

In this section, we compare the new scheme and similar schemes (Sun and Li, 2011; Li *et al*., 2013; Yu and Yang, 2015b) in terms of the computational cost and security properties (Table 1).

Table 1 Comparison of efficiency and security in different schemes

Scheme	Efficiency		Security	
	Mul	Pair	Con	Unf
Li *et al*. (2013)	5	1(+6)	Yes	Yes
Yu and Yang (2015b)	3	1(+7)	Yes	Yes
Sun and Li (2011)	2	2(+0)	Yes	No
New scheme	9	0(+0)	Yes	Yes

Mul: number of scalar multiplication operations in additive groups; Pair: number of bilinear pairing operations in multiplication groups; Con: confidentiality (IND-CCA2 security); Unf: unforgeability (sUF-CMA security). $x(+y)$ indicates x times pairing computations and y times pairing pre-computations

As we know, the computational cost of one pairing operation is higher than that of one ECC scalar multiplication operation (Szczechowiak *et al*., 2008). It is easy to see from Table 1 that the new scheme with no pairing operations is more efficient than the existing schemes (Sun and Li, 2011; Li *et al*., 2013; Yu and Yang, 2015b). In addition, the new scheme satisfies both IND-CCA2 and sUF-CMA security. Therefore, our scheme is superior to the existing schemes in terms of efficiency and security.

7 Conclusions

In this paper, we combined CLHS with an ECC to devise a novel and LC-CLHS scheme under the ECCDH and ECDL assumption. In the random oracle model, this scheme can resist adaptive chosen-ciphertext attacks and adaptive chosen-message attacks. In the light of the analysis of security properties and efficiency, the new scheme can realize secure, authentic, and efficient communication of arbitrary messages in resource-limited environments, such as ad hoc networks and wireless sensor networks.

In the future, we are going to study signcryption KEMs suitable for network coding and the universally composable security of hybrid signcryption algorithms.

References

Dent, A.W., 2005. Hybrid signcryption schemes with insider security. *LNCS*, **3574**:253-266. http://dx.doi.org/10.1007/11506157_22

Hwang, R.J., Lai, C.H., Su, F.F., 2005. An efficient signcryption scheme with forward secrecy based on elliptic curve. *Appl. Math. Comput.*, **167**:870-881. http://dx.doi.org/10.1016/j.amc.2004.06.124

Koblitz, N., 1987. Elliptic curve cryptosystems. *Math. Comput.*, **48**(177):203-209. http://dx.doi.org/10.1090/S0025-5718-1987-0866109-5

Li, F.G., Liao, Y.J., Qin, Z.G., *et al*., 2012. Further improvement of an identity-based signcryption scheme in the standard model. *Comput. Electr. Eng.*, **38**(2):413-421. http://dx.doi.org/10.1016/j.compe-leceng.2011.11.001

Li, F.G., Shirase, M., Takagi, T., 2013. Certificateless hybrid signcryption. *Math. Comput. Model.*, **57**(3-4):324-343. http://dx.doi.org/10.1016/j.mcm.2012.06.011

Li, F.G., Zheng, Z.H., Jin, C.H., 2016. Identity-based deniable authenticated encryption and its application to e-mail system. *Telecommun. Syst.*, **62**(4):625-639. http://dx.doi.org/10.1007/s11235-015-0099-1

Pang, L.J., Cui, J.J., Li, H.X., *et al*., 2011. A new multi-receiver ID-based anonymous signcryption. *Chin. J. Comput.*, **34**(11):2104-2113. http://dx.doi.org/10.3724/SP.J.1016.2011.02104

Sun, Y.X., Li, H., 2011. Efficient certificateless hybrid signcryption. *J. Softw.*, **22**(7):1690-1698. http://dx.doi.org/10.3724/SP.J.1001.2011.03825

Szczechowiak, P., Oliveira, L.B., Scott, M., *et al.*, 2008. Testing the limits of elliptic curve cryptography in sensor networks. *LNCS*, **4913**:305-320. http://dx.doi.org/10.1007/978-3-540-77690-1_19

Tan, C.H., 2008. Insider-secure signcryption KEM/tag-KEM schemes without random oracles. 3rd Int. Conf. on Availability, Reliability and Security, p.1275-1281. http://dx.doi.org/10.1109/ARES.2008.112

Wang, D.X., Teng, J.K., 2015. Provably secure identity-based aggregate signcryption scheme. *J. Comput. Appl.*, **35**(2):412-415. http://dx.doi.org/10.11772/j.issn.1001-9081.2015.02.0412

Wang, F.H., Hu, Y.P., Wang, C.X., 2012. Post-quantum secure hybrid signcryption from lattice assumption. *Appl. Math. Inform. Sci.*, **6**(1):23-28.

Youn, T.Y., Hong, D., 2012. Signcryption with fast online signing and short signcryptext for secure and private communication. *Sci. China Inform. Sci.*, **55**(11):2530-2541. http://dx.doi.org/10.1007/s11432-012-4635-2

Yu, H.F., Yang, B., 2015a. Identity-based hybrid signcryption scheme using ECC. *J. Softw.*, **26**(12):3174-3182. http://dx.doi.org/10.13328/j.cnki.jos.004819

Yu, H.F., Yang, B., 2015b. Provable secure certificateless hybrid signcryption. *Chin. J. Comput.*, **38**(4):804-813. http://dx.doi.org/10.3724/SP.J.1016.2015.00804

Zhang, B., Xu, Q.L., 2010. Identity-based multi-signcryption scheme without random oracles. *Chin. J. Comput.*, **33**(1):103-110. http://dx.doi.org/10.3724/SP.J.1016.2010.00103