Certificateless Hybrid Signcryption Scheme with Known Session-Specific Temporary Information Security

Ming Luo, Yuwei Wan, and Donghua Huang
(Corresponding author: Ming Luo)

School of Software, Nanchang University, Nanchang, China
999, Xuefu Avenue, Jiangxi Sheng, Nanchang Shi, Xinjian Xian, China (Email: lmhappy21@163.com)
(Received Sept. 23, 2016; revised and accepted Jan 10 & Feb. 1, 2017)

Abstract

The hybrid signcryption scheme based on certificateless public key cryptography avoids the complexity of certificate management existing in the traditional public key cryptography and the inherent key escrow problem existing in identity-based public key cryptography. The certificateless hybrid signcryption scheme combined with certificateless signcryption key encapsulation mechanism and data encapsulation mechanism can dispose the messages with arbitrary length while conventional certificateless signcryption schemes cannot. Meanwhile, almost all the proposed certificateless hybrid signcryption schemes cannot survive against the known session-specific temporary information security (KSSTIS) attack. In this paper we propose an efficient certificateless hybrid signcryption scheme, and formally prove its security in random oracle model under the assumption of Diffie-Hellman mathematical hard problems. Compared with the previous schemes, our scheme has the advantage of lower computational cost by reducing the amount of bilinear pairing computation. Moreover, our scheme achieves KSSTIS attribute.

Keywords: Certificateless; Hybrid Signcryption; Random Oracle Model; KSSTIS

1 Introduction

Signcryption is a cryptographic primitive which performs both the functions of signature and encryption in one logical step. With lower computational and communication cost, signcryption promotes the development of public key cryptography. Traditional public key cryptography, identity-based public key cryptography (IBC) and certificateless public key cryptography are three important stages of public key cryptography. For a long period of time, many signcryption schemes using conventional public key infrastructure (PKI) have been proposed, which binds user's identity and public key with a certificate. But the certificate management is a particularly prominent issue. In order to solve this problem and reduce the burden on traditional PKI, Identity-based public key cryptography was proposed, and a number of related signcryption schemes [7, 8] have been proposed in recent years. For IBC, the public key is computed with the binary string of users identity, thus IBC does not need the certificate used in PKI. However, the private key of IBC is generated by a private key generator (PKG). In this situation, private key escrow becomes an inherent problem in IBC. The PKG can forge or decrypt any ciphertext.

The notion of certificateless public key cryptography (CLC) was presented by Al-Ryiami and Paterson [2], which solves the certificate management problem of the traditional PKI and the inherent key escrow problem of IBC. For CLC, the private key is divided into two parts, one part is selected by users themselves and the other is generated by a key generation center (KGC). In 2008, Barbosa and Farshim [3] firstly proposed a certificateless signcryption scheme and its security notions. Recently, many signcryption schemes [6,13] using certificateless cryptography have been proposed.

The notion of hybrid encryption was presented by Abe et al. [1], and then Dent proposed the notion of hybrid signcryption [4]. Hybrid signcryption includes two parts. One part is a key encapsulation mechanism (CLSC-KEM) and the other part is a data encapsulation mechanism (DEM).In recent years, some hybrid signcryption schemes have been proposed for various network applications [9, 11]. Li et al. [5] proposed the first certificateless hybrid signcryption(CLHSC) scheme. The scheme consists of a tag key encapsulation mechanism (tag-KEM) and a data encapsulation mechanism (DEM), and their scheme makes up for the lack of authentication security in Dent's scheme [4]. At the signcryption stage, a symmetric key is generated by the key encapsulation mechanism, and then outputs the signcryption data. At the decryption stage, after obtaining the symmetric key by decapsulating the signcryption data, the ciphertext will be decrypted. Later, Selvi et al. [10] pointed out that Li's scheme may be existentially forgeable and proposed an improved scheme. Recently, Yin and Liang [12] pointed out almost all certificateless signcryption schemes that have been proposed in the literature cannot effectively against the public-key-replacement attacks, and they proposed an enhanced scheme to fill this security gaps.

However, we find these certificateless hybrid signcryption schemes above cannot survive against known session-specific temporary information security (KSSTIS) attack. To compensate for this security flaw, this paper proposes a new hybrid signcryption scheme based on certificateless cryptography and proves that the scheme meets the confidentiality and unforgeability in random oracle model, also our scheme can against the public-key-replacement attacks. Compared with the schemes above, our scheme achieves KSSTIS security attributes and has less bilinear pairing computation.

2 Preliminaries

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be a cyclic additive and multiplicative group respectively, whose prime order is a large prime number q. P is a generator of the group $\mathbb{G}_1$. If a map $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfies the following properties, we call it bilinear pairing.

1) Bilinearity: for all $a, b \in \mathbb{Z}_q^*$, there is $\hat{e}(aP, bP) = \hat{e}(P, P)^{ab}$.

2) Computability: for all $P, N \in \mathbb{G}_1$, there is an efficient algorithm to compute $\hat{e}(P, N)$.

3) Non-degeneracy: there exists $P \in \mathbb{G}_1$, such that $\hat{e}(P, P) \neq 1_{G2}$.

We can construct bilinear pairing $\hat{e}$ using the modified Tate pairing and Weil pairing of elliptic curve over a finite field. The security of our scheme relies on the following hard problems.

Definition 1. *Computational Diffie-Hellman(CDH) problem: For two integers $a, b \in \mathbb{Z}_q^*$ and a generator P of $\mathbb{G}_1$, given the tuple (P, aP, bP) to compute abP is hard.*

Definition 2. *Computational Bilinear Diffie-Hellman (CBDH) problem: For three integers $a, b, c \in \mathbb{Z}_q^*$ and a generator P of $\mathbb{G}_1$, given the tuple (P, aP, bP, cP) to compute $\hat{e}(P, P)^{abc}$ is hard.*

3 Certificateless Hybrid Signcryption Scheme

In this section, the certificateless hybrid signcryption scheme is described in details. Our scheme includes the following algorithms:

Setup: On input of a security parameter k, KGC picks a bilinear pairing $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and three security cryptographic hash functions $H_1: \{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{G}_1$, $H_2: \{0,1\}^* \times (\mathbb{G}_1)^4 \times \mathbb{G}_2 \rightarrow \{0,1\}^n$ and $H_3: \{0,1\}^* \times (\mathbb{G}_1)^4 \rightarrow \{0,1\}^n$. Then the KGC randomly chooses a master key $s \in \mathbb{Z}_q^*$ and computes the master public key $P_{pub} = sP$. The KGC keeps the master key s and publishes the system parameters $params = < G_1, G_2, \hat{e}, q, P, P_{pub}, H_1, H_2, H_3 >$.

GUK (Generate user key): On input of an identity ID and the system parameters $params$, a user randomly choose $x_{ID} \in \mathbb{Z}_q^*$ as his secret key, and then computes his public key $PK_{ID} = x_{ID}P$.

EPPK (Extract partial private key): On input of an identity ID and the system parameters $params$, KGC computes $Q_{ID} = H_1(ID||PK_{ID})$, and then computes the partial private key $D_{ID} = sQ_{ID}$.

GSK (Generate symmetric key): On input of sender's identity ID_s, public key PK_s, and private key (x_s, D_s), receiver's identity ID_r and public key PK_r. Randomly choose $x, y \in \mathbb{Z}_q^*$, the sender does the following steps.

1) Compute $U = xP$, $T = \hat{e}(D_s, Q_r)$.
2) Compute session key $K_{AB} = H_2(ID_r,\ T,\ U,\ xPK_r,\ x_sPK_r,\ PK_r)$.
3) Obtain internal state information $\bar{W} = (x,\ y,\ U, x_s, D_s, ID_s, PK_s, ID_r, PK_r)$.
   Output $(K_{AB}, \bar{W})$.

Encapsulation: On input of a tag τ and internal state information $\bar{W}$. The algorithm works as the following steps.

1) Compute $w = y(D_s + x_sPK_r)$.
2) Compute $h = H_3(\tau, U, w, PK_s, Pk_r)$.
3) Compute $v = 1/(y(x + h))$.
   Output $\delta = (U, w, v)$.

Decapsulation: On input of signcryption δ, a tag τ, the sender's identity ID_s, public key PK_s, and the receiver's identity ID_r, public key PK_r, private key (x_r, D_r). The receiver does the following steps.

1) Compute $h = H_3(\tau, U, w, PK_s, PK_r)$.
2) Check if $\hat{e}(vw,\ U + hP) \stackrel{?}{=} \hat{e}(Q_s,\ P_{pub})\hat{e}(PK_s,\ PK_r)$. If it is correct, go on and do the following computations. Otherwise stop and return $\perp$.
3) Compute $T = \hat{e}(D_r, Q_s)$.
4) Compute session key $K_{AB} = H_2(ID_r,\ T,\ U,\ x_rU,\ x_rPK_s,\ PK_r)$.

4 Security Analysis

In this section, we use some mathematical hard problems to analyze the confidentiality and unforgeability security of the scheme in the random oracle model, then we show that our scheme can survive against known session-specific temporary information security (KSSTIS) attacks.

4.1 Consistency

Our scheme satisfies the consistency.

$$\begin{aligned}&\hat{e}(vw, U+hP)\\ =&\hat{e}(y(D_s+x_sPK_r)/y(x+h), xP+hP)\\ =&\hat{e}((D_s+x_sPK_r), P)\\ =&\hat{e}(D_s, P)\hat{e}(x_sPK_r, P)\\ =&\hat{e}(Q_s, P_{pub})\hat{e}(PK_s, PK_r).\end{aligned}$$

4.2 Confidentiality

Theorem 1. *Assuming that CBDH is hard to solve in random oracle model, the scheme is secure against any IND-CLHSC-CCA2-I adversary A_I attack.*

Proof. Assuming that the challenger C receives an CBDH challenge tuple (P, aP, bP, cP), where P is a generator of cyclic additive $\mathbb{G}_1$. And the goal for C is to compute the answer of $\hat{e}(P,P)^{abc}$. The challenger C sends the system parameters *params* to A_I, and sets $P_{pub} = aP$. C maintains several lists $L_1, L_2, L_3, L_u, L_e, L_d$ and answers the following queries. Among these lists, L_1, L_2, L_3 simulate H_1, H_2, H_3 oracle respectively,L_u is used to track GUK query, L_e is used to track Encapsulation query, L_d is used to track Decapsulation query.

H_1 **query:** C selects two random numbers $i, j \in \{1, 2, \cdots, q_1\}$, where q_1 is the number of H_1 queries. At the n-th query:

1) if $ID_n = ID_i$, C answers $Q_i = bp$, and adds the tuple $(ID_i, \perp, bP)$ into list L_1.
2) if $ID_n = ID_j$, C answers $Q_j = cP$, and adds the tuple $(ID_j, \perp, cP)$ into list L_1.
3) if $ID_n \notin \{ID_i, ID_j\}$, C randomly chooses $w \in \mathbb{Z}_q^*$, answers $Q_n = wP$, and then returns it and adds the tuple (ID_n, w, Q_n) into list L_1.

H_2 **query:** C checks if there exists a tuple $(ID_r, T, U, xPK_r, x_sPK_r, PK_r, h_2)$ in the list L_2. If the tuple is found, C returns h_2. Otherwise, C randomly chooses $h_2 \in \{0,1\}^n$, and then returns it and adds the tuple $(ID_r, T, U, xPK_r, x_sPK_r, PK_r, h_2)$ into list L_2.

H_3 **query:** C checks if there exists a tuple $(\tau, U, w, PK_s, PK_r, h_3)$ in the list L_3. If the tuple is found, C returns h_3. Otherwise, C randomly chooses $h_3 \in \mathbb{Z}_q^*$, and then returns it and adds the tuple$(\tau, U, w, PK_s, PK_r, h_3)$ into list L_3.

GUK **query:** A_I picks an identity ID_n, C randomly chooses $x_n \in \mathbb{Z}_q^*$, and then answers $PK_n = x_nP$, adds the tuple(ID_n, x_n, PK_n) into list L_u.

$EPPK$ **query:** A_I picks an identity ID_n. Assuming that the identity ID_n has made H_1 query before, if $ID_n \in \{ID_i, ID_j\}$, stops the challenge. Otherwise, C searches the corresponding tuple (ID_n, w, Q_n) in the list L_1, returns $D_n = wP_{pub}$ and answers D_n.

Corruption query: A_I picks an identity ID_n. Assuming that the identity ID_n has made GUK query before, C searches the corresponding tuple in the list L_1, and answers x_n.

RPK query: A_I picks a new tuple (ID_n, PK_n), C updates the list L_u and replaces with $(ID_n, \perp, PK_n)$.

GSK query: A_I picks a tuple (ID_s, PK_s, ID_r, PK_r).

1) If $ID_s \notin \{ID_i, ID_j\}$, C randomly chooses $x, y \in \mathbb{Z}_q^*$, computes U and T. And then C runs the symmetric key generation algorithm and answers K_{AB}, updates and stores the internal state information.
2) If $ID_s \in \{ID_i, ID_j\}$, C stops simulation.

Encapsulation query: A_I produces a tag τ, at the same time, C checks if there exists an internal state information $\bar{W}$. If it is found, C performs the following steps. Otherwise, C stops the simulation and returns a $\perp$.

1) If $ID_s \notin \{ID_i, ID_j\}$, C computes $w = y(D_s + x_sPK_r)$ with the internal state information, and then computes $h = H_3(\tau, U, w, PK_s, PK_r)$and $v = 1/y(x+h)$. Finally, C answers the signcryption $\delta = (U, w, v)$ to A_I.
2) If $ID_s \in \{ID_i, ID_j\}$, C stops simulation.

Decapsulation query: A_I picks the tag τ, signcryption $\delta = (U, w, v)$, the sender's identity ID_s and the receiver's identity ID_r. C does the following processing:

1) If $ID_r \notin \{ID_i, ID_j\}$, firstly C computes $h = H_3(\tau, U, w, PK_s, PK_r)$, and then checks if $\hat{e}(vw, U+hP) \stackrel{?}{=} \hat{e}(Q_s, P_{pub})\hat{e}(PK_s, PK_r)$. If it is failure, C stops simulation and returns $\perp$. Otherwise, C computes $T = \hat{e}(D_r, Q_s)$, and then computes the session key $K_{AB} = H_2(ID_r, T, U, x_rU, x_rPK_s, PK_r)$.
2) If $ID_r \in \{ID_i, ID_j\}$, C stops simulation.

Challenge: A_I can stop the phase 1 queries whenever he wants, and then produces two challenge identities $\{ID_A, ID_B\}$, which $ID_A \neq ID_B$. if $\{ID_A, ID_B\} \notin \{ID_i, ID_j\}$, C stops simulation. Otherwise C randomly chooses $x, y \in \mathbb{Z}_q^*$, and sets $T^* = \eta$ (η as a candidate answer for CBDH problem), and then computes $U^* = xP$, $K_1 = H_2(ID_B, T^*, U^*, xPK_B, x_APK_B, PK_B)$. C randomly chooses a number $K_0 \in \{0,1\}^n$ and a bit $d \in \{0,1\}$, sends K_d to A_I. A_I chooses a tagτ^*

and sends it to C, C picks $w^* \in \mathbb{G}_1$, computes $h^* = H_3(\tau^*, U^*, w^*, PK_A, PK_B)$, $v^* = 1/y(x + h^*)$. Finally, C sends the signcryption $\delta^* = (U^*, w^*, v^*)$ to A_I.

A_I makes the queries of Phase 2 just like he made in the first phase. At last A_I produces a bit $d' \in \{0, 1\}$ as a guess to d. Only when A_I uses the tuple $(ID_B, T^*, U^*, xPK_B, x_A PK_B, PK_B)$to make H_2 query, he can check the correctness of the signcryption $\delta^* = (U^*, w^*, v^*)$, and if $d' = d$, C outputs T as a solution of the CBDH since the candidate answer $K_{AB} = H_2(ID_B, T^*, U^*, x_B U^*, x_B PK_A, PK_B)$ for CBDH problem is in the list L_2, where $T^* = \eta = \hat{e}(D_B, Q_A) = \hat{e}(acP, bP) = \hat{e}(P, P)^{abc}$. If $d' \neq d$, C fails and outputs F.

Thus, if the adversary A_I wants to break the signcryption algorithm, he must solve the CBDH with non-negligible advantage first. What he can do is to extract information from the signcryption messages, then uses some polynomial-time algorithm to solve the CBDH problem. But we all know that this algorithm does not exist so far. Therefore, when attacked by an IND-CLHSC-CCA2 adversary A_I, the proposed CLHSC scheme can maintain a safe state. □

Theorem 2. *Assuming that CDH is hard to solve in random oracle model, the scheme is secure against any IND-CLHSC-CCA2-II adversary A_{II} attack.*

Proof. Assuming that the challenger C receives an CDH challenge tuple (P, aP, bP), where P is a generator of cyclic additive $\mathbb{G}_1$. And the goal for C is to compute the answer of abP. C randomly chooses a number $s \in \mathbb{Z}_q^*$ as the master secret key, sets $P_{pub} = sP$, and sends the system parameters *params* and s to A_{II}. C maintains several lists $L_1, L_2, L_3, L_u, L_e, L_d$ and answers the following queries. Among these lists, L_1, L_2, L_3 simulate H_1, H_2, H_3 oracle respectively, L_u is used to track GUK query, L_e is used to track Encapsulation query, L_d is used to track Decapsulation query.

H_1 **query:** A_{II} randomly picks an identity ID_i, and sends it to C. C randomly chooses $w \in \mathbb{Z}_q^*$, computes $Q_n = wP$, and then returns it and adds the tuple (ID_n, w, Q_n) into list L_1.

H_2 **query:** The same as Theorem 1.

H_3 **query:** The same as Theorem 1.

GUK **query:** C selects a random number $i \in \{0, 1, \cdots, q_u\}$, where q_u is the number of GUK queries. At the n-th query:

1) If $ID_n \neq ID_i$, C randomly chooses $x_n \in \mathbb{Z}_q^*$ as the secret value, computes the public key $PK_n = x_n P$, and then adds the tuple (ID_n, x_n, PK_n) into list L_u and answers PK_n.
2) If $ID_n = ID_i$, C sets $PK_i = bP$, adds the tuple $(ID_i, \perp, bP)$ into list L_u.

Corruption query: A_{II} picks an identity ID_n. Assuming that the identity ID_n has made GUK query before, if $ID_n = ID_i$, C stops simulation. Otherwise, C searches the corresponding tuple in the list L_u and answers x_n.

GSK query: A_{II} picks a tuple (ID_s, PK_s, ID_r, PK_r).

1) If $ID_s \neq ID_i$, C randomly chooses $x, y \in \mathbb{Z}_q^*$, computes U and T. And then C runs the symmetric key generation algorithm and answers K_{AB}, updates and stores the internal state information.
2) If $ID_s = ID_i$, C stops simulation.

Encapsulation query: A_{II} produces a tag τ, and at the same time, C checks if there exists an internal state information $\bar{W}$. If it is found, perform the following steps. Otherwise, C stops the simulation and returns $\perp$.

1) If $ID_s \neq ID_i$, C computes $w = y(D_s + x_s PK_r)$ with the internal state information, and then computes $h = H_3(\tau, U, w, PK_s, PK_r)$ and $v = 1/y(x + h)$. Finally, C answers the signcryption $\delta = (U, w, v)$ to A_{II}.
2) If $ID_s = ID_i$, C stops simulation.

Decapsulation query: A_{II} picks the tag τ, signcryption $\delta = (U, w, v)$, the sender's identity ID_s and the receiver's identity ID_r. C does the following processing:

1) If $ID_r \neq ID_i$, firstly C computes $h = H_3(\tau, U, w, PK_s, PK_r)$, and then checks if $\hat{e}(vw, U + hP) \stackrel{?}{=} \hat{e}(Q_s, P_{pub})\hat{e}(PK_s, PK_r)$. If it is failure, C stops simulation and returns $\perp$. Otherwise, C computes $T = \hat{e}(D_r, Q_s)$, and then computes the session key $K_{AB} = H_2(ID_r, T, U, x_r U, x_r PK_s, PK_r)$.
2) If $ID_r = ID_i$, C stops simulation.

Challenge: A_{II} can stop the phase 1 queries whenever he wants, and produces two challenge identities $\{ID_A, ID_B\}$, which $ID_A \neq ID_B$. If $ID_B \neq ID_i$, C stops simulation. Otherwise C sets $U^* = aP$, randomly chooses $y \in \mathbb{Z}_q^*$, and then computes $T^* = \hat{e}(D_A, Q_B)$, $K_1 = H_2(ID_B, T^*, U^*, \eta, x_A PK_B, PK_B)$ (η as a candidate answer for CDH problem). C randomly chooses a number $K_0 \in \{0, 1\}^n$ and a bit $d \in \{0, 1\}$, sends K_d to A_{II}. A_{II} chooses a tag τ^* and sends it to C, C picks $v^* \in \mathbb{Z}_q^*$, computes $w^* = y(D_A + x_A PK_B)$, $h^* = H_3(\tau^*, U^*, w^*, PK_A, PK_B)$. Finally, C sends the signcryption $\delta^* = (U^*, w^*, v^*)$ to A_{II}.

A_{II} makes the queries of Phase 2 just like he made in the first phase. At last A_{II} produces a bit $d' \in \{0, 1\}$ as a guess to d. Only when A_{II} uses the tuple $(ID_B, T^*, U^*, \eta, x_A PK_B, PK_B)$ to make H_2 query, he can check the correctness of the signcryption $\delta^* = (U^*, w^*, v^*)$, and if $d' = d$, C outputs T as a solution of the CDH since the candidate answer $K_{AB} = H_2(ID_B, T^*, U^*, \eta, x_A PK_B, PK_B)$ for CDH problem is in the list L_2, where $\eta = x_B U^* = baP = abP$. If $d' \neq d$, C fails and outputs F.

Thus, if the adversary A_{II} wants to break the signcryption algorithm, he must solve the CDH with non-negligible advantage first. What he can do is to extract information from the signcryption messages, then use some polynomial-time algorithm to solve the CDH problem. This algorithm does not exist yet. Therefore, when attacked by an IND-CLHSC-CCA2 adversary A_{II}, the proposed CLHSC scheme can maintain a safe state. □

4.3 Unforgeability

Theorem 3. *Assuming that CDH is hard to solve in random oracle model, our scheme is secure against any sUF-CLHSC-CMA-I adversary A_I attack.*

Proof. Assuming that the challenger C receives an CDH challenge tuple (P, aP, bP), where P is a generator of cyclic additive $\mathbb{G}_1$. And the goal for C is to compute the answer of abP. The challenger C sends the system parameters *params* to A_I, and set $P_{pub} = aP$. C maintains several lists $L_1, L_2, L_3, L_u, L_e, L_d$ and answers the following queries. Among these lists, L_1, L_2, L_3 simulate H_1, H_2, H_3 oracle respectively, L_u is used to track GUK query,L_e is used to track Encapsulation query, L_d is used to track Decapsulation query.

H_1 **query:** C selects a random number $i \in \{1, 2, \cdots, q_1\}$,where q_1 is the number of H_1 queries. At the n-th query:

1) If $ID_n = ID_i$, C answers $Q_i = bP$, and adds the tuple $(ID_i, \perp, bP)$ into list L_1.
2) If $ID_n \neq ID_i$, C randomly chooses $w \in Z_q^*$, answers $Q_n = wP$, and then returns it and adds the tuple (ID_n, w, Q_n) into list L_1.

H_2 **query:** The same as Theorem 1.

H_3 **query:** The same as Theorem 1.

GUK **query:** The same as Theorem 1.

$EPPK$ **query:** A_I picks an identity ID_n. Assuming that the identity ID_n has made H_1 query before, if $ID_n = ID_i$, stops the challenge. Otherwise, C searches the corresponding tuple (ID_n, w, Q_n) in the list L_1, returns $D_n = wP_{pub}$ and answers D_n.

Corruption query: The same as Theorem 1.

RPK query: The same as Theorem 1.

GSK query: The same as Theorem 2.

Encapsulation query: The same as Theorem 2.

Decapsulation query: The same to Theorem 2.

Eventually, A_I produces a valid forgery quaternion $(\tau^*, \delta^*, ID_A, ID_B)$. C checks if $ID_A \neq ID_i$. If it is the case, C aborts. Otherwise, with the help of GUK oracle, C can obtain $ID_A's$ public key PK_A and $ID_B's$ public key PK_B, respectively. After that C uses tuple $(\tau^*, U^*, w^*, PK_A, PK_B)$ to make H_3 query and obtains h^* from list L_3. Then C does the following verification:

$$\begin{aligned} \hat{e}(v^*w^*, U^* + h^*P) &= \hat{e}(Q_A, P_{pub})\hat{e}(PK_A, PK_B) \\ \hat{e}(w^*/y, P) &= \hat{e}(bP, aP)\hat{e}(x_AP, PK_B) \\ \hat{e}(abP, P) &= \hat{e}(P, (w^*/y) - x_APK_B). \end{aligned}$$

At last, C can compute $abP = (w^*/y) - x_APK_B$.

If verification is right, C returns 1, otherwise 0.

So, if there exists a special adversary A_I who can forge a valid encapsulation message by learning something about the signcryption, that means there is an algorithm which can solve CDH problem with non-negligible advantage. However, this cannot happen. In other words, there is no adversary who can forge in this way. Thus, the scheme is secure against any sUF-CLHSC-CMA-I adversary A_I attack. □

Theorem 4. *Assuming that CDH is hard to solve in random oracle model, the scheme is secure against any IND-CLHSC-CCA2-II adversary A_{II} attack.*

Proof. Assuming that the challenger C receives an CDH challenge tuple (P, aP, bP), where P is a generator of cyclic additive $\mathbb{G}_1$. And the goal for C is to compute the answer of abP. C randomly chooses a number $s \in \mathbb{Z}_q^*$ as the master secret key, sets $P_{pub} = sP$, and sends the system parameters *params* and s to A_{II}. C maintains several lists, $L_1, L_2, L_3, L_u, L_e, L_d$ and answers the following queries. Among these lists, L_1, L_2, L_3 simulate H_1, H_2, H_3 oracle respectively, L_u is used to track GUK query, L_e is used to track Encapsulation query, L_d is used to track Decapsulation query.

H_1 **query:** The same as Theorem 2.

H_2 **query:** The same as Theorem 1.

H_3 **query:** The same as Theorem 1.

GUK **query:** C selects two random numbers $i, j \in \{1, 2, \cdots, q_u\}$, where q_u is the number of GUK queries. At the n-th query:

1) If $ID_n = ID_i$, C answers $PK_i = aP$, and adds the tuple $(ID_i, \perp, aP)$ into list L_u.
2) If $ID_n = ID_j$, C answers $PK_j = bP$, and adds the tuple $(ID_j, \perp, bP)$ into list L_u.
3) If $ID_n \notin \{ID_i, ID_j\}$, C randomly chooses $x_n \in \mathbb{Z}_q^*$ as the secret key and computes $PK_n = x_nP$, and then answers it and adds the tuple (ID_n, x_n, PK_n) into list L_u.

Corruption query: A_{II} picks an identity ID_n. Assuming that the identity ID_n has been made GUK query before, if $ID_n \in \{ID_i, ID_j\}$, C stops simulation. Otherwise, C searches the corresponding tuple in the list L_u and answers x_n.

GSK query: The same as Theorem 1.

Encapsulation query: The same as Theorem 1.

Decapsulation query: The same as Theorem 1.

Eventually, A_{II} produces a valid forgery quaternion $(\tau^*, \delta^*, ID_A, ID_B)$. C checks if $\{ID_A, ID_B\} \notin \{ID_i, ID_j\}$ and $ID_A \neq ID_B$. If it is the case, C aborts. Otherwise, with the help of GUK oracle, C can obtain $ID_A's$ public key PK_A and $ID_B's$ public key PK_B respectively. After that C uses tuple $(\tau^*, U^*, w^*, PK_A, PK_B)$ to make H_3 query and obtains h^* from list L_3. Then do the following verification:

$$\hat{e}(v^*w^*, U^* + h^*P) = \hat{e}(Q_A, P_{pub})\hat{e}(PK_A, PK_B)$$
$$\hat{e}(w^*/y, P) = \hat{e}(D_A, P)\hat{e}(aP, bP)$$
$$\hat{e}(abP, P) = \hat{e}(P, (w^*/y) - D_A).$$

At last, C can compute $abP = (w^*/y) - D_A$.

If verification is right, C returns 1, otherwise 0.

So, if there exists a special adversary A_{II} who can forge a valid encapsulation message by learning something about the signcryption, that means there is an algorithm which can solve CDH problem with non-negligible advantage. This is impossible. In other words, there is no adversary who can forge in this way. Thus, the scheme is secure against any sUF-CLHSC-CMA-II adversary A_{II} attack. □

4.4 Known Session-specific Temporary Information Security

Assuming that at the $j-th$ communication, ephemeral key x_j and signcryption $\delta_j = (U_j, w_j, v_j)$ is leaked. For adversary A_I, he can not obtain the related information about private key (D_s, x_s) or (D_r, x_r). A_I cannot compute $T_j = \hat{e}(D_s, Q_r)$ or $T_j = \hat{e}(D_r, Q_s)$ under the assumption of CBDH problem and cannot compute x_sPK_r or x_rPK_s under the assumption of CDH problem. All above problems will lead to the result that it is hard to obtain the value of session key $K_{AB} = H_2(ID_r, T, U, x_jPK_r, x_sPK_r, PK_r)$ for A_I. For adversary A_{II}, in the scheme, A_{II} can obtain the partial private key D_s or D_r, and then he can compute $T_j = \hat{e}(D_s, Q_r)$ or $T_j = \hat{e}(D_r, Q_s)$. But A_{II} cannot compute x_sPK_r or x_rPK_s without x_s or x_r under the assumption of CDH problem. This leads to the result that it is hard to compute $K_{AB} = H_2(ID_r, T, U, x_jPK_r, x_sPK_r, PK_r)$. Hence, our scheme can survive against Known session-specific temporary information security (KSSTIS) attack. But in Li's scheme [5], when the adversary obtains the ephemeral key r_j of $j-th$ communication, he can obtain $T = \hat{e}(P_{pub}, Q_{ID_r})^{r_j}$ easily. And then it is easy for the adversary to obtain the session key $K_{AB} = H_2(U, T, r_jPK_{ID_r}, ID_r, PK_{ID_r})$. The same situation happens in Yin's scheme [12]. When the adversary obtains the ephemeral key r_{1-j}, r_{2-j} of $j-th$ communication, he can obtain $R_1 = r_{1-j}P, R_2 = r_{2-j}P, U = r_{1-j}PK_R$ and $V = \hat{e}(r_{2-j}Q_R, P_{pub})$ easily. And the session key $K = H_2(ID_S, ID_R, R_1, R_2, U, V)$ can be easily obtained.

5 Performance Analysis

In this section, we will compare the scheme with Li's scheme and Yin's scheme from two aspects: the security and the efficiency of Encapsulation(include GSK phase) and Decapsulation phase in the table 1. We assume that all the three schemes use the same parameters $< G_1, G_2, \hat{e}, q >$. In the column of "Security", "KISSTIS" refers to known session-specific temporary information security. "Y" and "N" denote that whether satisfy this security property. In the column of "Computation Cost", the notations "Encapsulation" and "Decapsulation" refer to the computation of Encapsulation and Decapsulation, respectively. Note that offline computation is not included in "Computation Cost". And here, three operations will be involved. MUL, EXP and PAI refer to the number of point scalar multiplications, exponentiations and bilinear pairing computations, respectively.

Table 1: Comparison of efficiency

Scheme	Security	Computation Cost	
	KISSTIS	Encapsulation	Decapsulation
Li [5]	N	4MUL+EXP	MUL+4PAI
Yin [12]	N	5MUL+EXP	4MUL+3PAI
Ours	Y	3MUL	3MUL+PAI

Through Table 1 we can see that our scheme only needs 3 point scalar multiplications at the Encapsulation step, which is more efficient than the other two schemes. And at the Decapsulation stage, our scheme needs three point scalar multiplications and one bilinear pairing computation. The computation cost of bilinear pairing computation is the most expensive in the scheme based on bilinear pairing. Although Li's scheme only needs one point scalar multiplication, the number of bilinear pairing computations is far more than our scheme. Hence, our scheme is the most efficient. And from the security aspect, our scheme achieves the known session-specific temporary information security, which Li's and Yin's schemes can not satisfy.

6 Conclusion

In this paper, a secure CLHSC scheme is proposed from bilinear pairing in random oracle model. In addition, the scheme is highly efficient with only one bilinear pairing operation. In terms of security, we solve the flaw that most of the hybrid signcryption schemes cannot survive against known session-specific temporary information security attack. Considering any length of plaintext can be handled by hybrid signcryption and the efficiency of our scheme, our scheme can be applied to the high security requirements of communication networks and bandwidth-constrained communication environments, such as ad hoc net, 4G communication and so on.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China under grant [no. 61662046 and 61601215] and the Science and Technology project of Jiangxi Province of China under grant no. 20142BBE50019.

References

[1] M. Abe, R. Gennaro, K. Kurosawa, and V. Shoup, "Tag-kem/dem: A new framework for hybrid encryption and a new analysis of kurosawa-desmedt kem," in *Advances in Cryptology (EUROCRYPT'05)*, pp. 128–146, 2005.

[2] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 452–473, 2003.

[3] M. Barbosa and P. Farshim, "Certificateless signcryption," in *Proceedings of the 2008 ACM symposium on Information, Computer and Communications Security*, pp. 369–372, 2008.

[4] A. W. Dent, "Hybrid signcryption schemes with outsider security," in *International Conference on Information Security*, pp. 203–217, 2005.

[5] F. G. Li, M. Shirase, and T. Takagi, "Certificateless hybrid signcryption," *Mathematical and Computer Modelling*, vol. 57, no. 3, pp. 324–343, 2013.

[6] M. Luo, S. Q. Wang, and J. Hu, "A more efficient and secure broadcast signcryption scheme using certificateless public-key cryptography for resource-constrained networks," *Journal of Internet Technology*, vol. 17, no. 1, pp. 81–89, 2016.

[7] M. Mandal, G. Sharma, and A. K. Verma, "A computational review of identity-based signcryption schemes," *International Journal of Network Security*, vol. 18, no. 5, pp. 969–977, 2016.

[8] Y. Ming and Y. M. Wang, "Cryptanalysis of an identity based signcryption scheme in the standard model," *International Journal of Network Security*, vol. 18, no. 1, pp. 165–171, 2016.

[9] M. Ramakrishnan and R. Sujatha, "Cf-huffman code based hybrid signcryption technique for secure data transmission in medical sensor network," *International Journal of Applied Engineering Research*, vol. 10, no. 4, pp. 11455–11474, 2015.

[10] S. S. D. Selvi, S. S. Vivek, and C. Pandu Rangan, "Breaking and re-building a certificateless hybrid signcryption scheme," *ePint IACR org/2009/62*, 2009.

[11] R. Sujatha, M. Ramakrishnan, N. Duraipandian, and B. Ramakrishnan, "Optimal adaptive genetic algorithm based hybrid signcryption algorithm for information security," *CMES: Computer Modeling in Engineering & Sciences*, vol. 105, no. 1, pp. 47–68, 2015.

[12] A. H. Yin and H. C. Liang, "On security of a certificateless hybrid signcryption scheme," *Wireless Personal Communications*, vol. 85, no. 4, pp. 1727–1739, 2015.

[13] Y. W. Zhou, B. Yang, and W. Z. Zhang, "Provably secure and efficient leakage-resilient certificateless signcryption scheme without bilinear pairing," *Discrete Applied Mathematics*, vol. 204, pp. 185–202, 2016.

Biography

Ming Luo received the B.E. and Ph.D degree from Northeastern University, Shenyang, China in 2004 and 2010, respectively. Now he is an associate professor at the School of Software, Nanchang University, Nanchang, China. He has won lots of scholarships in China and was supported by the National High-Tech Research and Development Plan of China, the National Natural Science Foundation of China and the Science and Technology Program of Jiangxi Province. His research interests are networks security, information security and cryptography.

Yuwei Wan is a M.S. candidate at the school of software, Nanchang University. Her current research interests include information security and cryptography.

Donghua Huang is a M.S. candidate at the school of software, Nanchang University. His current research interests include information security and cryptography.