The ISC Int'l Journal of Information Security

July 2022, Volume 14, Number 2 (pp. 193–213)

http://www.isecure-journal.org

Attacks to Some Recently Proposed CL-SC Schemes and Presenting a Secure Scheme with KSSTIS **

Parvin Rastegari [1,*]

[1] *Electrical and Computer Engineering Group, Golpayegan College of Engineering, Isfahan University of Technology, Golpayegan, 87717-67498, Iran*

ARTICLE INFO.

Article history:
Received: January 7, 2021
Revised: July 3, 2021
Accepted: February 12, 2022
Published Online: February 20, 2022

Keywords:
Certificateless Signcryption, KSSTIS, Standard Model, Random Oracle Model

Type: Research Article

doi: 10.22042/ISECURE.2022.266258.602

dor: 20.1001.1.20082045.2022.14.2.7.9

ABSTRACT

The certificateless public key cryptography (CL-PKC) setting, makes it possible to overcome the problems of the conventional public key infrastructure and the ID-Based public key cryptography, concurrently. A certificateless signcryption (CL-SC) scheme is an important cryptographic primitive which provides the goals of a signature scheme and an encryption scheme both at once, in a certificateless setting. In addition to the basic security requirements of a CL-SC scheme (i.e. the unforgeability and the confidentiality), a new security notion called as the known session specific temporary information security (KSSTIS) has been proposed in the literature, recently. This security notion guarantees the confidentiality of the message even if the temporary information, used for creating the signcryption on the message, reveals. However, as discussed in the literature, there are not any secure CL-SC schemes in the standard model (i.e. without the assumption of random oracles) which guarantees the KSSTIS. In this paper, three recently proposed CL-SC schemes (Caixue, Shan and Ullah *et al.*'s schemes) are analyzed and it is shown that these schemes not only do not satisfy the KSSTIS, but also they do not even provide the basic security requirements of a CL-SC scheme. Furthermore, an enhanced secure CL-SC scheme is proposed in the standard model which satisfies the KSSTIS.

1 Introduction

The conventional public key infrastructure (PKI) makes difficulties for the certificate authority (CA), as it must manage the creation, the storage and the propagation of the certificates of all users' public keys. In order to overcome this problem, the concept of the ID-based public key cryptography (ID-PKC) was proposed in which a private key generation center (PKG) creates the private keys for the users from the unique identifier information of the users without requiring the corresponding public keys to be certified [1]. However ID-PKC suffers from a major problem called as the key escrow problem, as PKG knows the private keys of all users. The well-known certificateless public key cryptography (CL-PKC) setting provides solutions for overcoming the problems of the conventional PKI and ID-PKC, simultaneously [2]. In a CL-PKC, a part of the private keys is generated by a key generation center (KGC) and the other part is created by the user himself/herself.

In 1997, Zhang introduced the notion of the signcryption scheme which is an important primitive in

* Corresponding author.
**This article is an extended/revised version of an ISCISC'17 paper.
Email address: p.rastegari@iut.ac.ir
ISSN: 2008-2045

public key cryptography to satisfy the confidentiality and the unforgeability both at once, in an approach much more efficient than encrypting and signing messages seperately [3].

In 2008, Barbosa and Farshim proposed the notion of the signcryption scheme in the ceritificateless setting, called as the certificateless signcryption (CL-SC) scheme [4]. A CL-SC scheme should satisfy two basic security requirements of a signcryption scheme (i.e. the confidentiality and the unforgeability) against two types of attackers named as the public key replacement attacker ($\mathcal{A}_I$) and the malicious KGC attacker ($\mathcal{A}_{II}$) in the literature. The researchers had not proposed any CL-SC schemes without the assumption of the random oracles till 2010. Note that according to the Rogaway's discussions in [5], the schemes with the security proofs in the random oracle model may be insecure when the real-world primitives (such as hash functions) are used instead of the random oracles. So, it is desirable to provide the security proofs in the standard model, i.e. without the assumption of the random oracles. In 2010, Liu *et al.* proposed the first CL-SC scheme in the standard model [6] which was later attacked in several papers [7–9]. These attacks break the confidentiality and the unforgeability of the proposal in [6] against both malicious KGC and public key replacement attackers. Consequently, researchers worked more on this topic and tried to propose some CL-SC schemes without random oracles [10–20]. However, the proposed scheme in [10] is still vulnerable against the malicious KGC attacks proposed in [8]. Furthermore, the proposal in [13] is vulnerable in the sense of both the confidentiality and the unforgeability against the key replacement and the malicious KGC attacks proposed in [21].

In 2018, the authors in [17] introduced a new security notion called as the known session specific temporary information security (KSSTIS) for a CL-SC scheme. This security notion guarantees the confidentiality of the message even if the temporary information, used for creating the signcryption on it, reveals. Furthermore, the authors in [17] proposed a CL-SC scheme with the claim of satisfying both the basic security requirements (i.e. the unforgeability and the confidentiality against $\mathcal{A}_I$ and $\mathcal{A}_{II}$) and the KSSTIS in the standard model. However, their proposal has some errors in its construction and is insecure according to the discussions in [20] and [22]. The authors in [20] tried to propose another CL-SC scheme with KSSTIS in the standard model. The authors in [23] analyzed the scheme in [20] and designed an attack against its confidentiality, however their attack is too weak, as the attacker needs to know the private key of the sender to break the confidentiality. So, to the best of the author's knowledge, there are not any secure CL-SC scheme in the standard model which satisfies the KSSTIS, in the literature.

As a result of the previous discussions, the CL-SC schemes in [11, 12, 14, 16, 18, 19] are the only CL-SC schemes without random oracles in the literature which any attacks have not been proposed against them till now. However, as discussed in [20] and will discuss in this paper, none of these schemes do not satisfy the KSSTIS. In this paper, the author will enhance the scheme in [16] to provide the first CL-SC scheme with KSSTIS in the standard model. Note that the scheme in [16] is more efficient than the previously proposed schemes, i.e. the schemes in [11, 12, 14]. Furthermore, although the new proposed CL-SC schemes in [18], [19] and [24] are claimed to be more efficient than the proposal in [16], however none of them are secure according to the attacks will be presented in this paper (Note that the proposed scheme in [24] is not proved in the standard model). As a result, the proposal in [16] seems to be the best candidate among all the CL-SC schemes in the standard model to enhance for providing the KSSTIS.

1.1 Contributions:

The contributions of this work can be categorized as follows:

- The recently proposed CL-SC schemes [18, 19, 24] are analyzed and some attacks are designed which show the vulnerabilities of these schemes. Moreover, it is shown that none of these schemes guarantee the KSSTIS. So, these new proposed CL-SC schemes not only do not satisfy the KSSTIS, but also they do not even provide the basic security requirements of a CL-SC scheme. As a result none of them seem to be a good candidate for improving to provide the enhanced security notion KSSTIS.
- The CL-SC scheme in [16] (which seems to be the best candidate among all the CL-SC schemes according to the previous descriptions) is enhanced to provide the first CL-SC scheme with KSSTIS in the standard model. The new proposal not only inherits the basic security requirements (i.e. the unforgeability and the confidentiality against $\mathcal{A}_I$ and $\mathcal{A}_{II}$) from [16], but also guarantees the enhanced security notion KSSTIS.

The rest of the paper is prepared as follows. Some required preliminaries are provided in Section 2. In Section 3, the concept of the CL-SC scheme and its security requirements are described. In Section 4, the author analyzes Caixue's CL-SC Scheme [18] and designs an attack against its unforgeability. In Section 5, the author analyzes Shan's CL-SC Scheme [19] and

designs malicious KGC attacks against its unforgeability and confidentiality. In Section 6, the author analyzes Ullah *et al.*'s CL-SC Scheme [24] and shows the errors and weaknesses of this scheme. In Section 7, the author proposes her CL-SC scheme with the KSSTIS security in the standard model. In Section 8, a comparison of CL-SC schemes is provided. Finally, the paper is concluded in Section 9.

2 Preliminaries

2.1 Bilinear Pairings

Let G_1 and G_2 be multiplicative cyclic groups of prime order p and g be a generator of G_1. An admissible bilinear pairing is a mapping $e : G_1 \times G_1 \longrightarrow G_2$ which satisfies the following properties:

(1) Bilinearity: For all $a, b \in \mathbb{Z}_p^*$, $e(g^a, g^b) = e(g, g)^{ab}$.
(2) Non-degeneracy: $e(g, g) \neq 1_{G_2}$.
(3) Computability: There exists an efficient algorithm to calculate $e(g, g)$.

2.2 Related Complexity Assumptions

$k+1$-Computational-Diffie-Hellman-Exponent ($k+1$-CDHE) Problem [25]: Given $g, g^a, \ldots, g^{a^k} \in G_1$ (for unknown $a \in \mathbb{Z}_p^*$), compute $g^{a^{k+1}} \in G_1$.

Definition 1. It is said that (t, ε)-$k+1$-CDHE assumption holds in G_1 if there are not any algorithms which can solve the $k+1$-CDHE problem in G_1 in time at most t with probability at least ε.

$(\mathcal{S}, k)$-Computational-Bilinear-Diffie-Hellman-Exponent-Set ($(\mathcal{S}, k)$-CBDHE-Set) Problem [25]: Let $\mathcal{S}$ be a set of integers and $\mathcal{S} +_p \mathcal{S} = \{i + j \bmod \lambda(p) : i, j \in \mathcal{S}\}$, in which $\lambda(p)$ is the order of elements modulo p. Furthermore, consider another integer $k \notin \mathcal{S} +_p \mathcal{S}$. The $(\mathcal{S}, k)$-CBDHE-Set problem is that given $\{g^{a^i} \in G_1 : i \in \mathcal{S}\}$ (for unknown $a \in \mathbb{Z}_p^*$) compute $X = e(g, g)^{a^k}$. Note that if $\mathcal{S} = \mathcal{S}_j = \{0, 1, \ldots, j\}$ and $k = 2j+1$, the $(\mathcal{S}_j, 2j+1)$-CBDHE-Set problem is that given $g, g^a, g^{a^2}, \ldots g^{a^j} \in G_1$ (for unknown $a \in \mathbb{Z}_p^*$) compute $X = e(g, g)^{a^{2j+1}}$.

Definition 2. It is said that (t, ε)-$(\mathcal{S}, k)$-CBDHE-Set assumption holds in (G_1, G_2) if there are not any algorithms which can solve the $(\mathcal{S}, k)$-CBDHE-Set problem in (G_1, G_2) in time at most t with probability at least ε.

$(\mathcal{S}, k)$-Decisional-Bilinear-Diffie-Hellman-Exponent-Set ($(\mathcal{S}, k)$-DBDHE-Set) Problem [25]: Let $\mathcal{S}$ be a set of integers and $\mathcal{S} +_p \mathcal{S} = \{i + j \bmod \lambda(p) : i, j \in \mathcal{S}\}$, in which $\lambda(p)$ is the order of elements modulo p. Furthermore, consider another integer $k \notin \mathcal{S} +_p \mathcal{S}$. The $(\mathcal{S}, k)$-DBDHE-Set problem is that given $\{g^{a^i} \in G_1 : i \in \mathcal{S}\}$ (for unknown $a \in \mathbb{Z}_p^*$) and $X \in G_2$, decide whether $X = e(g, g)^{a^k}$ or not. Note that if $\mathcal{S} = \mathcal{S}_j = \{0, 1, \ldots, j\}$ and $k = 2j+1$, the $(\mathcal{S}_j, 2j+1)$-DBDHE-Set problem is that given $g, g^a, g^{a^2}, \ldots g^{a^j} \in G_1$ (for unknown $a \in \mathbb{Z}_p^*$) and $X \in G_2$, decide whether $X = e(g, g)^{a^{2j+1}}$ or not.

Definition 3. It is said that (t, ε)-$(\mathcal{S}, k)$-DBDHE-Set assumption holds in (G_1, G_2) if there are not any algorithms which can solve the $(\mathcal{S}, k)$-DBDHE-Set problem in (G_1, G_2) in time at most t with probability at least ε.

Remark 1. The security of our proposal in Section 7 is based on the $(\mathcal{S}_1, 3)$-DBDHE-Set, $(\mathcal{S}_2, 5)$-DBDHE-Set, 2-CDHE, 3-CDHE and $(\mathcal{S}_2, 5)$-CBDHE-Set assumptions.

3 CL-SC Scheme

3.1 Syntax

A key generation center (KGC), a sender (A) and a receiver (B) are three entities in a CL-SC scheme which has seven algorithms as follows [6] (note that in the following algorithms the user u may be the sender A or the receiver B):

- **Setup.** The KGC runs this phase, as it takes a security parameter λ as input and returns system parameters $Params$ which will be propagated and a master secret key msk which is a secret for the KGC.

$$(Params, msk) \longleftarrow Setup(\lambda).$$

- **Extract Partial Private Key (ExtPPK).** When a user u asks KGC for a partial private key, the KGC generates a a partial private key d_u, on inputs $Params$, msk and the identity of the user ID_u and sends it to u via a secure channel.

$$d_u \longleftarrow ExtPPK(Params, msk, ID_u).$$

- **Set Secret Value (SetSV).** The user u chooses a secret value x_u for himself/herself, on inputs $Params$ and ID_u.

$$x_u \longleftarrow SetSV(Params, ID_u).$$

- **Set Private Key (SetPrK).** The user u calculates his/her full private key SK_u, on inputs $Params$, ID_u, x_u and d_u and keeps it secret.

$$SK_u \longleftarrow SetPrK(Params, ID_u, x_u, d_u).$$

- **Set Public Key (SetPuK).** The user u calculates his/her public key PK_u, on inputs $Params$, ID_u, x_u and (sometimes) d_u and publishes it.

$$PK_u \longleftarrow SetPuK(Params, ID_u, x_u, (d_u)).$$

- **Signcryption (SC).** The sender A creates a signcryption δ on a message m, on inputs $Params$, m, ID_A, SK_A, ID_B and PK_B. Then A sends δ to B.

$$\delta \longleftarrow SC(Params, m, ID_A, SK_A, ID_B, PK_B).$$

- **Unsigncryption (USC).** Upon receiving δ from A, the receiver B decrypts and verifies δ, on inputs $Params$, δ, ID_A, PK_A, ID_B and SK_B. Then B returns m for a valid and $\perp$ for an invalid signcryption.

$$USC(Params, \delta, ID_A, PK_A, ID_B, SK_B) = m/\perp.$$

It is clear that if a message m is correctly signcrypted by the SC algorithm, the USC algorithm must return m (correctness).

3.2 Security Requirements

In a certificateless setting in public key cryptography, there are two types of adversaries [16]:

- The type I adversary $\mathcal{A}_I$ who can replace public keys of the users, but does not have access to the master secret key which is called as the key replacement attacker. In adversarial models in the literature, $\mathcal{A}_I$ is assumed to have access to the Public-Key, Partial-Private-Key, Replace-Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles.
- The type II adversary $\mathcal{A}_{II}$ who has access to the master secret key, but is not able to replace public keys which is called as the malicious KGC attacker. In adversarial models in the literature, $\mathcal{A}_{II}$ is assumed to have access to the Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles.

A CL-SC scheme must satisfy two basic security requirements, i.e. the confidentiality and the unforgeability against both $\mathcal{A}_I$ and $\mathcal{A}_{II}$. These security requirements are defined by the corresponding games against $\mathcal{A}_I$ and $\mathcal{A}_{II}$, as follows.

3.2.1 Confidentiality

The two following games are useful to define the confidentiality of a CL-SC scheme in the sense of IND-CCA against $\mathcal{A}_I$ and $\mathcal{A}_{II}$, respectively [16, 20].

Game 1. This game is executed between a challenger $\mathcal{C}$ and $\mathcal{A}_I$, as follows:

- **Initialization.** On input a security parameter λ, $\mathcal{C}$ produces $Params$ and msk. Then $\mathcal{C}$ gives $Params$ to $\mathcal{A}_I$ and keeps msk secret.
- **Phase 1 Queries.** In this phase, $\mathcal{C}$ must respond to all polynomially bounded number of $\mathcal{A}_I$'s queries from Public-Key, Partial-Private-Key, Replace-Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles.
- **Challenge.** In this step, $\mathcal{A}_I$ chooses two equal lengths messages m_0 and m_1 and two identities ID_{A^*} and ID_{B^*} and sends them to $\mathcal{C}$. $\mathcal{C}$ flips a coin to pick a random $\gamma \in_R \{0,1\}$ and creates a signcryption δ^* of m_γ from A^* to B^* and sends it to $\mathcal{A}_I$.
- **Phase 2 Queries.** $\mathcal{A}_I$ issues polynomially bounded number of queries to the oracles similar to that in Phase 1 Queries and $\mathcal{C}$ must respond to them.
- **Guess.** $\mathcal{A}_I$ returns γ^*.

$\mathcal{A}_I$ is the winner of Game 1 if:

(1) $\gamma^* = \gamma$.
(2) $\mathcal{A}_I$ cannot obtain SK_{B^*}.
(3) $\mathcal{A}_I$ cannot obtain SK_u of a user u, if the corresponding public key has already been replaced.
(4) $\mathcal{A}_I$ cannot obtain d_{B^*} if PK_{B^*} has been replaced before the challenge step.
(5) In phase 2 queries, $\mathcal{A}_I$ is not allowed to send an Unsigncrypt query on δ^* of ID_{A^*} to ID_{B^*}, unless PK_{A^*} or PK_{B^*}, used to signcrypt m_γ, has been replaced after sending the challenge.

Game 2. This game is executed between a challenger $\mathcal{C}$ and $\mathcal{A}_{II}$, as follows:

- **Initialization.** On input a security parameter λ, $\mathcal{C}$ produces $Params$ and msk and gives them to $\mathcal{A}_{II}$.
- **Phase 1 Queries.** In this phase, $\mathcal{C}$ must respond to all polynomially bounded number of $\mathcal{A}_{II}$'s queries from Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles.
- **Challenge.** In this step, $\mathcal{A}_{II}$ chooses two equal lengths messages m_0 and m_1 and two identities ID_{A^*} and ID_{B^*} and sends them to $\mathcal{C}$. $\mathcal{C}$ flips a coin to pick a random $\gamma \in_R \{0,1\}$ and creates a signcryption δ^* of m_γ from A^* to B^* and sends it to $\mathcal{A}_{II}$.
- **Phase 2 Queries.** $\mathcal{A}_{II}$ issues polynomially bounded number of queries to the oracles similar to that in Phase 1 Queries and $\mathcal{C}$ must respond to them.
- **Guess.** $\mathcal{A}_{II}$ returns γ^*.

$\mathcal{A}_{II}$ is the winner of Game 2 if:

(1) $\gamma^* = \gamma$.
(2) $\mathcal{A}_{II}$ cannot obtain SK_{B^*}.
(3) In phase 2 queries, $\mathcal{A}_{II}$ is not allowed to send an Unsigncrypt query on δ^* of ID_{A^*} to ID_{B^*}.

Definition 4. A CL-SC scheme is $(t, \varepsilon, q_{PK}, q_d, q_{RPK}, q_{SK}, q_{SC}, q_{USC})$-confidential in the sense of indistinguishability of encryptions against adap-

tive chosen ciphertext attack (IND-CCA), if there are not any adversaries ($\mathcal{A}_I$ and $\mathcal{A}_{II}$) which can win Game 1 and Game 2 in time at most t, with probability at least $\frac{1}{2}+\varepsilon$, by issuing at most q_{PK} Public-Key queries, q_d Partial-Private-Key queries ($q_d = 0$ for $\mathcal{A}_{II}$), q_{RPK} Replace-Public-Key queries ($q_{RPK} = 0$ for $\mathcal{A}_{II}$), q_{SK} Private-Key queries, q_{SC} Signcrypt queries and q_{USC} Unsigncrypt queries.

3.2.2 Unforgeability

The two following games are useful to define the unforgeability of a CL-SC scheme in the sense of EUF-CMA against $\mathcal{A}_I$ and $\mathcal{A}_{II}$, respectively [16].

Game 3. This game is executed between a challenger $\mathcal{C}$ and $\mathcal{A}_I$, as follows:

- **Initialization.** On input a security parameter λ, $\mathcal{C}$ produces $Params$ and msk. Then $\mathcal{C}$ gives $Params$ to $\mathcal{A}_I$ and keeps msk secret.
- **Queries.** In this phase, $\mathcal{C}$ must respond to all polynomially bounded number of $\mathcal{A}_I$'s queries from Public-Key, Partial-Private-Key, Replace-Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles.
- **Forgery.** $\mathcal{A}_I$ forges a signcryption δ^* on a message m^* from ID_{A^*} to ID_{B^*}.

$\mathcal{A}_I$ is the winner of Game 3 if:

(1) δ^* is a valid signcryption on m^* from ID_{A^*} to ID_{B^*}.
(2) $\mathcal{A}_I$ cannot obtain SK_{A^*}.
(3) $\mathcal{A}_I$ cannot obtain SK_u of a user u, if the corresponding public key has already been replaced.
(4) $\mathcal{A}_I$ cannot obtain d_{A^*}.
(5) $\mathcal{A}_I$ is not allowed to send a Signcrypt query on m^* from ID_{A^*} to ID_{B^*}.

Game 4. This game is executed between a challenger $\mathcal{C}$ and $\mathcal{A}_{II}$, as follows:

- **Initialization.** On input a security parameter λ, $\mathcal{C}$ produces $Params$ and msk and gives them to $\mathcal{A}_{II}$.
- **Queries.** In this phase, $\mathcal{C}$ must respond to all polynomially bounded number of $\mathcal{A}_{II}$'s queries from Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles.
- **Forgery.** $\mathcal{A}_{II}$ forges a signcryption δ^* on a message m^* from ID_{A^*} to ID_{B^*}.

$\mathcal{A}_{II}$ is the winner of Game 4 if:

(1) δ^* is a valid signcryption on m^* from ID_{A^*} to ID_{B^*}.
(2) $\mathcal{A}_{II}$ cannot obtain SK_{A^*}.
(3) $\mathcal{A}_{II}$ is not allowed to send a Signcrypt query on m^* from ID_{A^*} to ID_{B^*}.

Definition 5. A CL-SC scheme is $(t, \varepsilon, q_{PK}, q_d, q_{RPK}, q_{SK}, q_{SC}, q_{USC})$-unforgeable in the sense of unforgeability against adaptive chosen message attack (EUF-CMA), if there are not any adversaries ($\mathcal{A}_I$ and $\mathcal{A}_{II}$) which can win Game 3 and Game 4 in time at most t, with probability at least ε, by issuing at most q_{PK} Public-Key queries, q_d Partial-Private-Key queries ($q_d = 0$ for $\mathcal{A}_{II}$), q_{RPK} Replace-Public-Key queries ($q_{RPK} = 0$ for $\mathcal{A}_{II}$), q_{SK} Private-Key queries, q_{SC} Signcrypt queries and q_{USC} Unsigncrypt queries.

3.2.3 Known Session Specific Temporary Information Security (KSSTIS)

In [17], the authors introduced the notion of KSSTIS, which guarantees that the message will be kept confidential, even if the temporary information used for creating the signcryption reveals. In this paper, the following game is considered for defining the KSSTIS of a CL-SC scheme against an adversary $\mathcal{A}$ who has a signcryption δ^* on a message m^* from A^* to B^* and the corresponding temporary information used for creating δ^*.

Game 5. This game is executed between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, as follows:

- **Initialization.** On input a security parameter λ, $\mathcal{C}$ produces $Params$ and msk. Then $\mathcal{C}$ gives $Params$ to $\mathcal{A}$ and keeps msk secret.
- **Queries.** In this phase, $\mathcal{C}$ must respond to all polynomially bounded number of $\mathcal{A}$'s queries from Public-Key, Partial-Private-Key, Private-Key, Signcrypt and Unsigncrypt oracles.
- **Output.** $\mathcal{A}$ returns the message m^{**}. (Note that $\mathcal{A}$ has a signcryption δ^* on a message m^* from A^* to B^*. Moreover, $\mathcal{A}$ knows the corresponding temporary information used for creating δ^*.)

$\mathcal{A}$ is the winner of Game 5 if:

(1) $m^{**} = m^*$.
(2) $\mathcal{A}$ cannot obtain SK_{B^*}.
(3) $\mathcal{A}$ is not allowed to send an Unsigncrypt query on δ^* of ID_{A^*} to ID_{B^*}.

Definition 6. A CL-SC scheme is $(t, \varepsilon, q_{PK}, q_d, q_{SK}, q_{SC}, q_{USC})$-KSSTIS, if there are not any adversaries $\mathcal{A}$ which can win Game 5 in time at most t, with probability at least ε, by issuing at most q_{PK} Public-Key queries, q_d Partial-Private-Key queries, q_{SK} Private-Key queries, q_{SC} Signcrypt queries and q_{USC} Unsigncrypt queries.

4 On the Security of Caixue's CL-SC Scheme

4.1 Algorithms of Caixue's Scheme

Caixue has designed the algorithms of his CL-SC scheme as follows [18]:

- **Setup.** The KGC gets a security parameter 1^k as input. Then it chooses two cyclic groups G_1 and G_2 of a prime order p, a random generator g of G_1, a bilinear pairing $e : G_1 \times G_1 \longrightarrow G_2$, a hash function $H_1 : \{0,1\}^* \longrightarrow \mathbb{Z}_p^*$, two random values $s \in_R \mathbb{Z}_p^*$ and $h_1 \in_R G_1$. Then it sets $g_1 = g^s$ and also selects a bijection $\phi : \{0,1\}^{2l} \longrightarrow G_2$, where l is the bit length of the message. The public parameters are $Params = \{G_1, G_2, e, g, g_1, h_1, H_1, \phi\}$ and the master secret key is $msk = s$.
- **ExtPPK.** When a user u with identity $ID_u \in \mathbb{Z}_p^*$ requests the KGC for the partial private key, the KGC picks a random $r_u \in_R \mathbb{Z}_p^*$, calculates:

  $$d_u = (d_{1,u}, d_{2,u}) = ((h_1.g^{-r_u})^{\frac{1}{s-ID_u}}, r_u),$$

  and sends it to u. (If $ID_u = s$, the KGC aborts.)
- **SetSV.** u picks $x_u \in_R \mathbb{Z}_p^*$ as his/her secret value.
- **SetPrK.** u sets

  $$SK_u = (SK_{1,u}, SK_{2,u}, SK_{3,u}) = (d_{1,u}, d_{2,u}, x_u)$$

  as his/her full private key.
- **SetPuK.** u sets

  $$PK_u = (PK_{1,u}, PK_{2,u}) = (g^{x_u}, (g_1.g^{-ID_u})^{x_u})$$

  as his/her public key.
- **SC.** In order to produce a signcryption δ on a message $m \in \{0,1\}^l$ for B, A chooses $t_1, t_2 \in_R \mathbb{Z}_p^*$, $T \in_R \{0,1\}^l$ and computes:

  $$\delta_1 = \phi(m||T).e(PK_{1,B}, PK_{1,B})^{-t_1}.e(g,h_1)^{-t_2},$$
  $$\delta_2 = g_1^{t_2}.g^{-t_2 ID_B}, \quad \delta_3 = e(g,g)^{t_1},$$
  $$\delta_4 = e(g,g)^{t_2}, \quad \delta_5 = SK_{2,A},$$
  $$\delta_6 = SK_{1,A}.PK_{1,A}^{SK_{3,A}.w}, \qquad (1)$$

  where in Eq. (1), $w = H_1(T, ID_A, ID_B, PK_A, PK_B, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$. Then A assigns $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6)$ and sends it to B.
- **USC.** In order to decrypt and verify the signcryption $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6)$, B computes:

  $$m||T = \phi^{-1}(\delta_1.e(\delta_2, SK_{1,B}).\delta_4^{SK_{2,B}}.\delta_3^{SK_{3,B}^2}).$$

  Afterwards B, who has the value of $m||T$ now, is able to extract m (the first l bits of $m||T$) and T (the second l bits of $m||T$) and obtain $w = H_1(T, ID_A, ID_B, PK_A, PK_B, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$. Finally, B accepts δ as a valid signcryption on m if the equation:

  $$e(g_1 g^{-ID_A}, \delta_6) = e(g, h_1 g^{-\delta_5}).e(PK_{2,A}, PK_{1,A}^w)$$

  holds, otherwise B rejects δ.

4.2 Cryptanalysis of Caixue's Scheme

In his paper, Caixue has claimed that his proposal is confidential and unforgeable against both $\mathcal{A}_I$ and $\mathcal{A}_{II}$ without the assumption of random oracles. However, the author has designed an attack which shows that Caixue's CL-SC scheme is easily forgeable. Suppose that B receives two signcryptions $\delta' = (\delta'_1, \delta'_2, \delta'_3, \delta'_4, \delta'_5, \delta'_6)$ on a message m' and $\delta'' = (\delta''_1, \delta''_2, \delta''_3, \delta''_4, \delta''_5, \delta''_6)$ on a message m'' from A. We have:

$$\delta'_1 = \phi(m'||T').e(PK_{1,B}, PK_{1,B})^{-t'_1}.e(g,h_1)^{-t'_2},$$
$$\delta'_2 = g_1^{t'_2}.g^{-t'_2 ID_B}, \quad \delta'_3 = e(g,g)^{t'_1},$$
$$\delta'_4 = e(g,g)^{t'_2}, \quad \delta'_5 = SK_{2,A},$$
$$\delta'_6 = SK_{1,A}.PK_{1,A}^{SK_{3,A}.w'},$$

where $w' = H_1(T', ID_A, ID_B, PK_A, PK_B, \delta'_1, \delta'_2, \delta'_3, \delta'_4, \delta'_5)$, and:

$$\delta''_1 = \phi(m''||T'').e(PK_{1,B}, PK_{1,B})^{-t''_1}.e(g,h_1)^{-t''_2},$$
$$\delta''_2 = g_1^{t''_2}.g^{-t''_2 ID_B}, \quad \delta''_3 = e(g,g)^{t''_1},$$
$$\delta''_4 = e(g,g)^{t''_2}, \quad \delta''_5 = SK_{2,A},$$
$$\delta''_6 = SK_{1,A}.PK_{1,A}^{SK_{3,A}.w''},$$

where $w'' = H_1(T'', ID_A, ID_B, PK_A, PK_B, \delta''_1, \delta''_2, \delta''_3, \delta''_4, \delta''_5)$.

B can easily find $m'||T'$ and $m''||T''$ by decrypting δ' and δ'', compute w', w'' and obtain:

$$PK_{1,A}^{SK_{3,A}} = (\frac{\delta'_6}{\delta''_6})^{(w'-w'')^{-1}},$$
$$SK_{1,A} = \frac{\delta'_6}{(PK_{1,A}^{SK_{3,A}})^{w'}} = \frac{\delta''_6}{(PK_{1,A}^{SK_{3,A}})^{w''}},$$
$$SK_{2,A} = \delta'_5 = \delta''_5.$$

By obtaining $PK_{1,A}^{SK_{3,A}}$, $SK_{1,A}$ and $SK_{2,A}$ from the above equations, B can forge a signcryption δ^* on any message m^* from A to any other receiver B^* by picking random values $t_1^*, t_2^* \in_R \mathbb{Z}_p^*$, $T^* \in_R \{0,1\}^l$ and computing:

$$\delta_1^* = \phi(m^*||T^*).e(PK_{1,B^*}, PK_{1,B^*})^{-t_1^*}.e(g,h_1)^{-t_2^*},$$
$$\delta_2^* = g_1^{t_2^*}.g^{-t_2^* ID_{B^*}}, \quad \delta_3^* = e(g,g)^{t_1^*},$$
$$\delta_4^* = e(g,g)^{t_2^*}, \quad \delta_5^* = SK_{2,A},$$
$$\delta_6^* = SK_{1,A}.(PK_{1,A}^{SK_{3,A}})^{w^*},$$

where $w^* = H_1(T^*, ID_A, ID_{B^*}, PK_A, PK_{B^*}, \delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*, \delta_5^*)$.

Note that the designed attack is done by a receiver B, as he/she can decrypt δ' and δ'' to obtain $m'||T'$ and $m''||T''$ and the corresponding w' and w'', respectively. Moreover, Caixue has claimed that his

scheme is secure even if the adversary has access to the Private-Key oracle, which returns the private key of an entity (except the private key of the target receiver in the proof of the confidentiality and the private key of the target sender in the proof of the unforgeability) to the adversary (see Definition 4 and Definition 5). As a result, the proposed attack shows that a receiver B (or every entity who has obtained SK_B from the Private-Key oracle) can easily forge a signature from A (a sender who produces δ' and δ'' and sends them to B) to any other receiver B^*.

4.3 On KSSTIS of Caixue's Scheme

It is easy to see that the Caixue's CL-SC Scheme does not satisfy the KSSTIS, as if the temporary information t_1, t_2 which is used to create a signcryption $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6)$ from A to B reveals, one can easily obtain $m||T$ as follows:

$$m||T = \phi^{-1}(\delta_1.e(PK_{1,B}, PK_{1,B})^{t_1}.e(g, h_1)^{t_2}),$$

and obtains the message m (the first l bits of $m||T$).

5 On the Security of Shan's CL-SC Scheme

5.1 Algorithms of Shan's Scheme

Shan has designed the algorithms of her CL-SC scheme as follows [19]:

- **Setup.** The KGC gets a security parameter $l \in \mathbb{Z}^+$ as input. Then it selects a prime p of the length l-bits, two multiplicative cyclic groups G and G_t of order p, a generator g of G, a random $\tilde{g} \in_R G$, a bilinear pairing $e : G \times G \longrightarrow G_t$, a hash function $H : G_t \longrightarrow \mathbb{Z}_p$ and $s, x, y \in_R \mathbb{Z}_p^*$. The public parameters are $Params = \{p, G, G_t, e, g, \tilde{g}, \tilde{S} = \tilde{g}^s, \tilde{X} = \tilde{g}^x, \tilde{Y} = \tilde{g}^y, X = g^x, Y = g^y\}$ and the master secret key is $msk = (s, x, y)$.
- **ExtPPK.** When a user u with identity $ID_u \in \mathbb{Z}_p^*$ requests the KGC for the partial private key, the KGC calculates:

$$d_u = (d_{1,u}, d_{2,u}, d_{3,u}) = (g^{\frac{x}{s+ID_u}}, g^{\frac{y}{s+ID_u}}, g^{\frac{1}{s+ID_u}})$$

  and sends it to u.
- **SetSV.** u picks $x_u \in_R \mathbb{Z}_p^*$ as his/her secret value.
- **SetPrK.** u sets $SK_u = (x_u, d_u)$ as his/her private key.
- **SetPuK.** u sets $PK_u = \tilde{g}^{x_u}$ as his/her public key.
- **SC.** In order to produce a signcryption δ on a message $m \in G_t$ for B, A chooses $r_1, r_2 \in_R \mathbb{Z}_p^*$ and computes:

$$U = (d_{1,A})^{r_1}.(d_{2,A})^{H(m)r_1}.(d_{3,A})^{x_A r_1}, \quad (2)$$
$$N = g^{r_1}, \quad V = (d_{3,A})^{r_1},$$
$$L = \tilde{Y}^{\frac{x_A}{r_1}}, \quad (3)$$
$$c = m.e(g, PK_B)^{-r_2}, \quad Z = \tilde{S}^{r_2}.\tilde{g}^{r_2 ID_B}.$$

  Then A assigns $\delta = (U, N, V, L, c, Z)$ and sends (ID_A, ID_B, δ) to B.
- **USC.** In order to decrypt and verify the signcryption $(ID_A, ID_B, \delta = (U, N, V, L, c, Z))$, B computes:

$$m = c.e(d_{3,B}, Z)^{x_B}, \quad (4)$$
$$T = \tilde{S}.\tilde{g}^{ID_A}, \quad W = \tilde{X}.PK_A.\tilde{Y}^{H(m)}.\tilde{g}.$$

  Then B is convinced to accept δ as a valid signcryption on m if the equation:

$$e(U.V, T).e(N, L) = e(N, W).e(Y, PK_A),$$

  holds, otherwise B rejects δ.

5.2 Cryptanalysis of Shan's Scheme

In her paper, Shan has claimed that her proposal is confidential and unforgeable against both $\mathcal{A}_I$ and $\mathcal{A}_{II}$ without the assumption of the random oracles. However, the author has designed malicious KGC attacks which break the confidentiality and the unforgeability of her scheme against $\mathcal{A}_{II}$. In the proposed attack, the malicious KGC ($\mathcal{A}_{II}$) generates all system parameters correctly as explained in the Setup algorithm of the scheme, except $\tilde{g}$. For generating $\tilde{g}$, $\mathcal{A}_{II}$ picks a random $\gamma \in_R \mathbb{Z}_p^*$ and assigns $\tilde{g} = g^\gamma$. By this malice, $\mathcal{A}_{II}$ can break the confidentiality and unforgeability of Shan's scheme, as described in the following two subsections. Note that $\mathcal{A}_{II}$ is a malicious KGC and it knows $msk = (s, x, y)$ and $d_u = (d_{1,u}, d_{2,u}, d_{3,u})$ (but not x_u) of all users.

5.2.1 Malicious KGC Attack Against the Confidentiality

By selecting $\tilde{g} = g^\gamma$, given a signcryption $(ID_A, ID_B, \delta = (U, N, V, L, c, Z))$, $\mathcal{A}_{II}$ can easily decrypt and verify it. For this purpose, $\mathcal{A}_{II}$ computes:

$$m = c.e(PK_B^{\gamma^{-1}}, Z)^{\frac{1}{s+ID_B}} \quad (5)$$
$$T = \tilde{S}.\tilde{g}^{ID_A}, \quad W = \tilde{X}.PK_A.\tilde{Y}^{H(m)}.\tilde{g},$$

and accepts δ if:

$$e(U.V, T).e(N, L) = e(N, W).e(Y, PK_A).$$

It is easy to check the correctness of Eq. (5), as:

$$m = c.e(PK_B^{\gamma^{-1}}, Z)^{\frac{1}{s+ID_B}} = c.e(\tilde{g}^{x_B\gamma^{-1}}, Z)^{\frac{1}{s+ID_B}}$$
$$= c.e(g^{\gamma x_B\gamma^{-1}}, Z)^{\frac{1}{s+ID_B}} = c.e(g^{x_B}, Z)^{\frac{1}{s+ID_B}}$$
$$= c.e(g^{\frac{1}{s+ID_B}}, Z)^{x_B} = c.e(d_{3,B}, Z)^{x_B},$$

which indicates Eq. (4).

As a result, $\mathcal{A}_{II}$ can verify every signcryption δ from A to B and obtain the message m without the knowledge of B's full private key SK_B. Therefore, Shan's scheme is not confidential against $\mathcal{A}_{II}$.

5.2.2 Malicious KGC Attack Against the Unforgeability

By selecting $\tilde{g} = g^{\gamma}$, $\mathcal{A}_{II}$ can easily forge a signcryption δ on m from A to B, by picking $r_1, r_2 \in_R \mathbb{Z}_p^*$ and computing:

$$U = (d_{1,A})^{r_1}.(d_{2,A})^{H(m)r_1}.(PK_A^{\gamma^{-1}})^{\frac{r_1}{s+ID_A}}, \quad (6)$$
$$N = g^{r_1}, \quad V = (d_{3,A})^{r_1},$$
$$L = PK_A{}^{\frac{y}{r_1}}, \quad (7)$$
$$c = m.e(g, PK_B)^{-r_2}, \quad Z = \tilde{S}^{r_2}.\tilde{g}^{r_2 ID_B}.$$

It is easy to check the correctness of Eq. (6) and Eq. (7) as:

$$\begin{aligned} U &= (d_{1,A})^{r_1}.(d_{2,A})^{H(m)r_1}.(PK_A^{\gamma^{-1}})^{\frac{r_1}{s+ID_A}} \\ &= (d_{1,A})^{r_1}.(d_{2,A})^{H(m)r_1}.(\tilde{g}^{x_A\gamma^{-1}})^{\frac{r_1}{s+ID_A}} \\ &= (d_{1,A})^{r_1}.(d_{2,A})^{H(m)r_1}.(g^{\gamma x_A\gamma^{-1}})^{\frac{r_1}{s+ID_A}} \\ &= (d_{1,A})^{r_1}.(d_{2,A})^{H(m)r_1}.(g^{\frac{1}{s+ID_A}})^{x_A r_1} \\ &= (d_{1,A})^{r_1}.(d_{2,A})^{H(m)r_1}.(d_{3,A})^{x_A r_1}, \end{aligned}$$

which indicates Eq. (2), and:

$$L = PK_A{}^{\frac{y}{r_1}} = (\tilde{g}^{x_A})^{\frac{y}{r_1}} = (\tilde{g}^{y})^{\frac{x_A}{r_1}} = \tilde{Y}^{\frac{x_A}{r_1}},$$

which indicates Eq. (3).

As a result, $\mathcal{A}_{II}$ can forge a signcryption δ on a desired message m from A to B without knowing the A's full private key SK_A. Therefore, Shan's scheme is not unforgeable against $\mathcal{A}_{II}$.

5.3 On KSSTIS of Shan's Scheme

It is easy to see that the Shan's CL-SC Scheme does not satisfy the KSSTIS, as if the temporary information r_2 which is used to create a signcryption $\delta = (U, N, V, L, c, Z)$ from A to B reveals, one can easily obtain m as follows:

$$m = c.e(g, PK_B)^{r_2}.$$

6 On the Security of Ullah *et al.*'s CL-SC Scheme

In [24], Ullah *et al.* have proposed a CL-SC scheme and claimed that their scheme is more efficient than the previous schemes including our proposal in [16], but there are two important comments on their paper, as follows:

(1) It should be noted that Ullah *et al.*'s comparison is not fair at all, since our proposal in [16] is provable secure in the standard model and as discussed in [16], it is more efficient than all previously proposed schemes in the standard model. Additionally, according to the vulnerability of the two recently proposed CL-SC schemes in the standard model, i.e. the proposed schemes in [18] and [19], which are described in Section 4 and Section 5, our proposal in [16] is still the most efficient CL-SC scheme in the standard model. However, Ullah *et al.* have not presented any proofs for the security of their scheme, even in the random oracle model, and they discussed on the security of their scheme heuristically, which is not acceptable nowadays.

(2) There are errors in the algorithms of their scheme which will be described as follows.

6.1 Algorithms of Ullah *et al.*'s Scheme

The algorithms of the Ullah *et al.*'s CL-SC scheme are briefly described as follows [24]:

- **Setup.** The KGC picks a random $s \in_R \mathbb{Z}_n^*$ as the master secret key and produces and publishes $PK_{pub} = s\mathcal{D}$, where $\mathcal{D}$ is the divisor of a hyper elliptic curve. It also picks two hash functions $H, H_1 : \{0,1\}^* \longrightarrow \mathbb{Z}_n^*$ and publishes them.
- **ExtPPK.** When a user u with identity ID_u requests the KGC for the partial private key, the KGC picks a random $r_u \in_R \mathbb{Z}_n^*$ and calculates:
  $$d_u = (\mathcal{L}_u, \alpha_u) = (r_u\mathcal{D}, r_u + sH_1(ID_u, \mathcal{L}_u, \mathcal{Y}_u)),$$
  where $\mathcal{Y}_u$ is a part of the user's public key.
- **SetSV.** u picks $x_u \in_R \mathbb{Z}_n^*$ as his/her secret value.
- **SetPrK.** u sets $SK_u = (x_u, \alpha_u)$ as his/her private key.
- **SetPuK.** u computes $\mathcal{Y}_u = x_u\mathcal{D}$ and sets $PK_u = (\mathcal{Y}_u, \mathcal{L}_u)$ as his/her public key.
- **SC.** In order to produce a signcryption δ on m for B, the sender A chooses $t, N_c \in_R \mathbb{Z}_n^*$ and computes:
  $$\gamma = \mathcal{L}_B + H_1(ID_B, \mathcal{L}_B, \mathcal{Y}_B)PK_{pub},$$
  $$Z = t\mathcal{D},$$
  $$K = (t(\mathcal{Y}_B + \gamma), Z, ID_B, \mathcal{L}_B, \mathcal{Y}_B),$$
  $$C = Enc_K(m, ID_A, N_c),$$
  where Enc_K is a symmetric encryption algorithm with key K. A also computes:
  $$h = H(m, ID_A, N_c),$$
  $$S = x_A + h(t + \alpha_A)$$
  Then A assigns $\delta = (C, S, h, Z)$ and sends it to B.
- **USC.** In order to decrypt and verify the signcryption $\delta = (C, S, h, Z)$, B computes:

$$K = ((x_B + \alpha_B)Z, Z, ID_B, \mathcal{L}_B, \mathcal{Y}_B)$$
$$(m, ID_A, N_c) = Dec_K(C),$$

where Dec_K is the symmetric decryption algorithm with key K. Then B calculates:

$$\beta = \mathcal{L}_A + H_1(ID_A, \mathcal{L}_A, \mathcal{Y}_A)PK_{pub},$$

and accepts the signature if the following equation holds:

$$S.\mathcal{D} = \beta + h(Z + \mathcal{Y}_A), \tag{8}$$

where $h = H(m, ID_A, N_c)$.

6.2 Cryptanalysis of Ullah *et al.*'s Scheme

As said, there are errors in Ullah *et al.*'s CL-SC scheme. for example:

(1) The verification algorithm in Eq. (8) is not correct, as:

$$\begin{aligned} S.\mathcal{D} &= (x_A + h(t + \alpha_A)).\mathcal{D} \\ &= x_A\mathcal{D} + h(t\mathcal{D} + \alpha_A\mathcal{D}) \\ &= \mathcal{Y}_A + h(Z + (r_A + sH_1(ID_A, \mathcal{L}_A, \mathcal{Y}_A))\mathcal{D}) \\ &= \mathcal{Y}_A + h(Z + r_A\mathcal{D} + H_1(ID_A, \mathcal{L}_A, \mathcal{Y}_A)s\mathcal{D}) \\ &= \mathcal{Y}_A + h(Z + \mathcal{L}_A + H_1(ID_A, \mathcal{L}_A, \mathcal{Y}_A)PK_{pub}) \\ &= \mathcal{Y}_A + h(Z + \beta), \end{aligned}$$

which is not satisfied by Eq. (8).

(2) Another concern about this scheme is that the KGC uses $\mathcal{Y}_u$ (a part of the user's public key) to generate d_u. However, in a CL-PKC, the KGC must generate the partial private keys of the users independently, before the users set their keys, except when we require the security against the level-3 KGC according to Girault's categorization [26], which is not discussed in Ullah *et al.*'s paper. Readers can refer to [27, 28] for more details on this topic.

(3) At last, this scheme is not a signcryption at all! Since in this scheme, A computes C which is in fact an encryption of m with a random key K and then creates a signature S on m. Therefore, this scheme actually applies the algorithms of a symmetric encryption scheme and then the algorithms of a signature scheme on m. However in a signcryption scheme the signature and encryption on a message m should be produced both at once by the public key cryptographic methods [3].

Totally, in contrast to Ullah *et al.*'s claims, their scheme is not comparable with the signcryption schemes in the standard model. Moreover, the security and even the correctness of the algorithms of their scheme is questionable as discussed above.

7 Proposed CL-SC Scheme

In this section, the author enhances the proposal in [16] to guarantee the KSSTIS security.

7.1 Algorithms of the Proposal

The algorithms of the proposal are as follows:

- **Setup.** Given a security parameter λ, the KGC creates and publishes $Params = \{G_1, G_2, p, e, g, g_1, T, u', v', U, V, H_1, H_2, H_u\}$, where G_1 and G_2 are multiplicative cyclic groups of a large prime order p, e is a bilinear pairing as $e : G_1 \times G_1 \longrightarrow G_2$, g is a generator of G_1, $g_1 = g^\alpha$ where $\alpha \in_R \mathbb{Z}_p^*$, $T = e(g_1, g_1)$, $u', v' \in_R G_1$, $U = (u_i)_{i=1}^{n_u}$ and $V = (v_j)_{j=1}^{n_m}$ are two vectors of lengths n_u and n_m that their elements are selected from G_1 randomly and $H_u : \{0,1\}^* \longrightarrow \{0,1\}^{n_u}$, $H_1 : G_2 \times G_1^4 \times \{0,1\}^* \longrightarrow \{0,1\}^{n_m}$, $H_2 : G_1^3 \times \{0,1\}^* \longrightarrow \mathbb{Z}_p^*$ are three collision resistant hash functions. Furthermore, the KGC computes $msk = g_1^\alpha$ and keeps it secret.
- **ExtPPK.** Suppose that $U = H_u(ID_u)$ and $U[i]$ denotes the i-th bit of U. Define the set $\mathcal{U}_u = \{i | U[i] = 1, i = 1, 2, \ldots, n_u\}$. The KGC picks a random $r_u \in_R \mathbb{Z}_p^*$ and computes the partial private key of the user u, d_u as follows:

$$d_u = (d_{u,1}, d_{u,2}) = (g_1^\alpha (u' \prod_{i \in \mathcal{U}_u} u_i)^{r_u}, g^{r_u}).$$

- **SetSV.** u picks $x_u, r'_u \in_R \mathbb{Z}_p^*$ as his/her secret value.
- **SetPrK.** u computes his full private key as:

$$\begin{aligned} SK_u &= (SK_{u,1}, SK_{u,2}, SK_{u,3}) \\ &= (d_{u,1}^{x_u^2} (u' \prod_{i \in \mathcal{U}_u} u_i)^{r'_u}, d_{u,2}^{x_u^2} g^{r'_u}, x_u). \end{aligned}$$

- **SetPuK.** u computes his/her public key as:

$$PK_u = (PK_{u,1}, PK_{u,2}) = (g_1^{x_u}, g_1^{\frac{1}{x_u}}).$$

- **SC.** Suppose that A wants to create a signcryption δ on a message $m \in G_2$ for B. A Checks the equality $e(PK_{B,1}, PK_{B,2}) = T$. If it does not hold, A returns $\perp$, otherwise A picks $r_1, r_2 \in_R \mathbb{Z}_p^*$ and computes:

$$\delta_1 = m.e(PK_{B,1}^{r_1 + SK_{A,3}}, PK_{B,1}),$$
$$\delta_2 = g^{r_1}, \quad \delta_3 = (u' \prod_{i \in \mathcal{U}_B} u_i)^{r_1}, \quad \delta_4 = SK_{A,2}.g^{r_2},$$
$$\delta_5 = SK_{A,1}.(u' \prod_{i \in \mathcal{U}_A} u_i)^{r_2}.(PK_{A,1}^h (v' \prod_{j \in \mathcal{M}_m} v_j))^{r_1},$$

where $\mathcal{M}_m = \{j | M[j] = 1, j = 1, 2, \ldots, n_m\}$, in which $M = H_1(\delta_1, \delta_2, \delta_3, \delta_4, PK_{B,1}, ID_B) \in \{0,1\}^{n_m}$, and $M[j]$ is the j-th bit of M. More-

over, $h = H_2(PK_{A,1}, \delta_2, \delta_4, ID_A, M)$. Then A sends $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ to B.

- **USC.** Upon receiving δ, B checks the equality $e(PK_{A,1}, PK_{A,2}) = T$. If it does not hold, B returns $\perp$, otherwise B computes $M = H_1(\delta_1, \delta_2, \delta_3, \delta_4, PK_{B,1}, ID_B)$, $\mathcal{M}_m = \{j | M[j] = 1, j = 1, 2, \ldots, n_m\}$ and $h = H_2(PK_{A,1}, \delta_2, \delta_4, ID_A, M)$ and verifies the equality:

$$e(\delta_5, g) = e(PK_{A,1}, PK_{A,1}).e(u' \prod_{i \in \mathcal{U}_A} u_i, \delta_4) .e(PK_{A,1}^h (v' \prod_{j \in \mathcal{M}_m} v_j), \delta_2). \quad (9)$$

If the equally does not hold, B returns $\perp$, otherwise, B computes m as follows:

$$m = \delta_1 . \frac{e(\delta_3, SK_{B,2})}{e(\delta_2, SK_{B,1}).e(PK_{A,1}^{SK_{B,3}}, PK_{B,1})}, \quad (10)$$

else returns $\perp$.

Remark 2. The proposed scheme is similar to the proposal in [16] in all algorithms, except in computing δ_1, which includes $SK_{A,3}$ in order to satisfy KSSTIS and consequently the USC algorithm is modified, too.

7.2 Security Analysis of the Proposal

7.2.1 Correctness

The correctness of Eq. (9) is easily satisfied as:

$$\begin{aligned}
e(\delta_5, g) &= e(SK_{A,1}, g).e((u' \prod_{i \in \mathcal{U}_A} u_i)^{r_2}, g) \\
&\quad .e((PK_{A,1}^h (v' \prod_{j \in \mathcal{M}_m} v_j))^{r_1}, g) \\
&= e(g^{\alpha^2 x_A^2} (u' \prod_{i \in \mathcal{U}_A} u_i)^{r_A x_A^2 + r'_A}, g) \\
&\quad .e((u' \prod_{i \in \mathcal{U}_A} u_i)^{r_2}, g) \\
&\quad .e((PK_{A,1}^h (v' \prod_{j \in \mathcal{M}_m} v_j))^{r_1}, g) \\
&= e(g^{\alpha^2 x_A^2}, g) \\
&\quad .e((u' \prod_{i \in \mathcal{U}_A} u_i)^{r_A x_A^2 + r'_A + r_2}, g) \\
&\quad .e((PK_{A,1}^h (v' \prod_{j \in \mathcal{M}_m} v_j))^{r_1}, g) \\
&= e(g_1^{x_A}, g_1^{x_A}).e(u' \prod_{i \in \mathcal{U}_A} u_i, SK_{A,2} g^{r_2}) \\
&\quad .e(PK_{A,1}^h (v' \prod_{j \in \mathcal{M}_m} v_j), g^{r_1}) \\
&= e(PK_{A,1}, PK_{A,1}).e(u' \prod_{i \in \mathcal{U}_A} u_i, \delta_4) \\
&\quad .e(PK_{A,1}^h (v' \prod_{j \in \mathcal{M}_m} v_j), \delta_2),
\end{aligned}$$

which shows the correctness of Eq. (9). Furthermore, the correctness of Eq. (10) is easily satisfied as:

$$\begin{aligned}
&\delta_1 . \frac{e(\delta_3, SK_{B,2})}{e(\delta_2, SK_{B,1}).e(PK_{A,1}^{SK_{B,3}}, PK_{B,1})} \\
&= \frac{m.e(PK_{B,1}^{r_1 + SK_{A,3}}, PK_{B,1}).e((u' \prod_{i \in \mathcal{U}_B} u_i)^{r_1}, g^{r_B x_B^2 + r'_B})}{e(g^{r_1}, g_1^{\alpha x_B^2} (u' \prod_{i \in \mathcal{U}_B} u_i)^{r_B x_B^2 + r'_B}).e(g_1^{x_A x_B}, g_1^{x_B})} \\
&= \frac{m.e(g_1^{x_B r_1}, g_1^{x_B}).e(g_1^{x_B x_A}, g_1^{x_B}).e((u' \prod_{i \in \mathcal{U}_B} u_i)^{r_1}, g^{r_B x_B^2 + r'_B})}{e(g^{r_1}, g_1^{\alpha x_B^2} (u' \prod_{i \in \mathcal{U}_B} u_i)^{r_B x_B^2 + r'_B}).e(g_1^{x_A x_B}, g_1^{x_B})} \\
&= \frac{m.e(g^{\alpha x_B r_1}, g^{\alpha x_B}).e(g^{r_1}, (u' \prod_{i \in \mathcal{U}_B} u_i)^{r_B x_B^2 + r'_B})}{e(g^{r_1}, g^{\alpha^2 x_B^2}) e(g^{r_1}, (u' \prod_{i \in \mathcal{U}_B} u_i)^{r_B x_B^2 + r'_B})} \\
&= \frac{m.e(g^{r_1}, g^{\alpha^2 x_B^2})}{e(g^{r_1}, g^{\alpha^2 x_B^2})} = m,
\end{aligned}$$

which shows the correctness of Eq. (10).

7.2.2 Confidentiality

The confidentiality of the proposed scheme, according to Definition 4, implies from two following lemmas:

Lemma 1. *If (t, ε)-$(\mathcal{S}_1, 3)$-DBDHE-Set assumption (according to Definition 3) holds in (G_1, G_2), the proposal is $(t_I, \varepsilon_I, q_{PK}, q_d, q_{RPK}, q_{SK}, q_{SC}, q_{USC})$-confidential against $\mathcal{A}_I$, where:*

$$\begin{aligned}
&\varepsilon \geq \frac{\varepsilon_I}{8 q_{USC}(q_d + q_{SK} + q_{SC} + q_{USC} + 1)(n_m + 1)(n_u + 1)}, \\
&t \leq t_I + order(((q_d + q_{SK} + q_{SC} + q_{USC}) n_u \\
&\quad + (q_{SC} + q_{USC}) n_m) T_M \\
&\quad + (q_{PK} + q_d + q_{SK} + q_{SC} + q_{USC}) T_E \\
&\quad + (q_{SC} + q_{USC}) T_P),
\end{aligned}$$

in which T_M, T_E and T_P are the times required for a multiplication and exponentiation in G_1 and a pairing computation, respectively.

Proof. See Appendix A. □

Lemma 2. *If (t, ε)-$(\mathcal{S}_2, 5)$-DBDHE-Set assumption (according to Definition 3) holds in (G_1, G_2), the proposal is $(t_{II}, \varepsilon_{II}, q_{PK}, q_{SK}, q_{SC}, q_{USC})$-confidential against $\mathcal{A}_{II}$, where:*

$$\begin{aligned}
&\varepsilon \geq \frac{\varepsilon_{II}}{8 q_{USC}(q_{SK} + q_{SC} + q_{USC} + 1)(n_m + 1)(n_u + 1)}, \\
&t \leq t_{II} + order(((q_{SK} + q_{SC} + q_{USC}) n_u \\
&\quad + (q_{SC} + q_{USC}) n_m) T_M \\
&\quad + (q_{PK} + q_{SK} + q_{SC} + q_{USC}) T_E \\
&\quad + (q_{SC} + q_{USC}) T_P),
\end{aligned}$$

Proof. See Appendix B. □

Theorem 1. *The proposed CL-SC scheme is confidential (IND-CCA) according to Definition 4.*

Proof. The proof implies from Lemma 1 and Lemma 2, directly. □

7.2.3 Unforgeability

The unforgeability of the proposed scheme, according to Definition 5, implies from two following lemmas:

Lemma 3. *If (t,ε)-2-CDHE assumption (according to Definition 1) holds in G_1, the proposal is $(t_I,\varepsilon_I,q_{PK},q_d,q_{RPK},q_{SK},q_{SC},q_{USC})$-unforgeable against $\mathcal{A}_I$, where:*

$$\varepsilon \geq \frac{\varepsilon_I}{8q_{USC}(q_d+q_{SK}+q_{SC}+q_{USC}+1)(n_m+1)(n_u+1)},$$
$$\begin{aligned} t \leq t_I + order(((q_d+q_{SK}+q_{SC}+q_{USC})n_u \\ +(q_{SC}+q_{USC})n_m)T_M \\ +(q_{PK}+q_d+q_{SK}+q_{SC}+q_{USC})T_E \\ +(q_{SC}+q_{USC})T_P), \end{aligned}$$

Proof. See Appendix C. □

Lemma 4. *If (t,ε)-3-CDHE assumption (according to Definition 1) holds in G_1, the proposal is $(t_{II},\varepsilon_{II},q_{PK},q_{SK},q_{SC},q_{USC})$-unforgeable against $\mathcal{A}_{II}$, where:*

$$\varepsilon \geq \frac{\varepsilon_{II}}{8q_{USC}(q_{SK}+q_{SC}+q_{USC}+1)(n_m+1)(n_u+1)},$$
$$\begin{aligned} t \leq t_{II} + order(((q_{SK}+q_{SC}+q_{USC})n_u \\ +(q_{SC}+q_{USC})n_m)T_M \\ +(q_{PK}+q_{SK}+q_{SC}+q_{USC})T_E \\ +(q_{SC}+q_{USC})T_P), \end{aligned}$$

Proof. See Appendix D. □

Theorem 2. *The proposed CL-SC scheme is unforgeable (EUF-CMA) according to Definition 5.*

Proof. The proof implies from Lemma 3 and Lemma 4, directly. □

7.2.4 KSSTIS

In the proposed CL-SC scheme, the author considers $SK_{A,3}$ in the power of $PK_{B,1}$ in computing δ_1, which guarantees KSSTIS. Note that as:

$$\delta_1 = m.e(PK_{B,1}^{r_1+SK_{A,3}}, PK_{B,1}),$$

even if the temporary information r_1 reveals, m keeps confidential. As a result, in contrast to the proposal in [16] in which we have:

$$\delta_1 = m.e(PK_{B,1}^{r_1}, PK_{B,1}),$$

and one can obtain m easily if r_1 reveals, the new scheme does not suffer from this weakness and satisfies the KSSTIS. The following theorem guarantees this claim, more precisely.

Theorem 3. *If (t,ε)-$(\mathcal{S}_2,5)$-CBDHE-Set assumption (according to Definition 2) holds in (G_1,G_2), the proposal is $(t',\varepsilon',q_{PK},q_d,q_{SK},q_{SC},q_{USC})$-confidential against $\mathcal{A}$, where:*

$$\varepsilon \geq \frac{\varepsilon'}{8(q_d+q_{SK}+q_{SC}+q_{USC})^2(n_u+1)^2},$$
$$\begin{aligned} t \leq t' + order(((q_d+q_{SK}+q_{SC}+q_{USC})n_u \\ +(q_{SC}+q_{USC})n_m)T_M \\ +(q_{PK}+q_d+q_{SK}+q_{SC}+q_{USC})T_E \\ +(q_{SC}+q_{USC})T_P), \end{aligned}$$

Proof. See Appendix E. □

In fact, the new scheme not only inherits the main security requirements (i.e. the confidentiality and the unforgeability against $\mathcal{A}_I$ and $\mathcal{A}_{II}$) from the proposal in [16], but also satisfies the KSSTIS property.

Remark 3. Note that in the security proofs, a simulator $\mathcal{B}$ is built to solve an instance of a hard problem by the use of the adversary as a sub-routine. To this goal, $\mathcal{B}$ must play the role of the challenger $\mathcal{C}$ (In Games 1, 2, 3, 4 and 5). If the simulator answers to all adversaries' queries without any ideal assumptions (such as existing random oracles), the proof will be in the standard model, but if some ideal assumptions (such as existing random oracles) are considered in the proof approach, the proof will be for example in the ROM. The proofs in Appendixes are provided in the standard model, which guarantee that the proposal is secure against key replacement and malicious KGC attackers.

8 Comparison

As mentioned in Section 1, after proposing the first CL-SC scheme in the standard model by Liu *et al.* in 2010 [6], the researchers have started to work on this topic and proposed several CL-SC schemes in the standard model [10–14, 16–20]. In Table 1, a comparison of these schemes is provided. In this table, E_{G_1}, E_{G_2} and P denote an exponentiation in G_1, an exponentiation in G_2 and a pairing computations, respectively. Furthermore, $|aG|$ shows the binary length of a elements in G and n_m is the binary length of the message m. As shown in Table 1, the proposals in [6, 10, 13, 17, 20] have some faults in their securities according to the discussions provided in [7–9, 20–23], as described in Section 1. Furtheremore, the proposals in [18] and [19] are not secure according to the proposed attacks in Section 4 and Section 5, respectively. As a result, the proposals in [11, 12, 14, 16] are the only CL-SC schemes in the literature which have not been attacked till now. Among these schemes, the proposal in [16] is the most efficient one. Note that although Shan in [19] and Ullah *et al.* in [24] have

Table 1. Comparison of CL-SC schemes in the standard model

Scheme	Signcrypt	Unsigncrypt		Ciphertext	Proposed	KSSTIS
		valid signcryption	invalid signcryption	length	attacks	
[6]	$1E_{G_2}+3E_{G_1}$	$5P$	$3P$	$\|4G_1\|+\|1G_2\|$	[7–9]	×
[10]	$3E_{G_2}+3E_{G_1}$	$5P+2E_{G_2}$	$3P+2E_{G_2}$	$\|4G_1\|+\|1G_2\|$	[8]	×
[13]	$1P+1E_{G_2}+5E_{G_1}$	$4P+2E_{G_1}$	$6P+2E_{G_1}$	$\|4G_1\|+n_m$	[21]	×
[17]	$1P+4E_{G_1}$	$6P+2E_{G_1}$	$4P$	$\|2G_1\|+\|1G_2\|$	[20, 22]	✓
[20]	$2P+4E_{G_1}$	$8P+2E_{G_1}$	$5P+1E_{G_1}$	$\|4G_1\|+\|1G_2\|$	[23]	✓
[18]	$1P+4E_{G_2}+3E_{G_1}$	$4P+5E_{G_1}$	$4P+5E_{G_1}$	$\|3G_1\|+\|3G_2\|$	Section 4	×
[19]	$1P+1E_{G_2}+8E_{G_1}$	$5P+1E_{G_2}+2E_{G_1}$	$5P+1E_{G_2}+2E_{G_1}$	$\|5G_1\|+\|1G_2\|$	Section 5	×
[11]	$3P+3E_{G_2}+6E_{G_1}$	$8P+2E_{G_2}+3E_{G_1}$	$6P+2E_{G_2}+1E_{G_1}$	$\|4G_1\|+\|1G_2\|$	-	×
[12]	$5P+1E_{G_2}+3E_{G_1}$	$10P$	$10P$	$\|4G_1\|+\|1G_2\|$	-	×
[14]	$2P+3E_{G_2}+5E_{G_1}$	$7P+2E_{G_1}$	$7P+2E_{G_1}$	$\|4G_1\|+\|2G_2\|$	-	×
[16]	$2P+7E_{G_1}$	$7P+1E_{G_1}$	$5P+1E_{G_1}$	$\|4G_1\|+\|1G_2\|$	-	×
proposed scheme	$2P+7E_{G_1}$	$7P+1E_{G_1}$	$5P+1E_{G_1}$	$\|4G_1\|+\|1G_2\|$	-	✓

claimed that their schemes are more efficient than the proposal in [16], however Shan's scheme [19] is not secure according to the proposed attacks in Section 5 and Ullah *et al.*'s scheme has basic errors as discussed in Section 6, e.g. it does not even satisfy the correctness and there are not any security proofs even in the random oracle model. Therefore, the author selected the scheme in [16] for enhancing to satisfy the KSSTIS and proposed the first secure CL-SC scheme in the standard model which guarantees the KSSTIS. As shown in Table 1, this enhancement do not affect on the computation and communication costs of the original scheme in [16].

9 Conclusion

In this work, the author provided a study on certificateless signcryption (CL-SC) schemes by focusing on an enhanced security notion called as the known session specific temporary information security (KSSTIS). She firstly discussed that none of the proposed CL-SC schemes in the standard model satisfy the KSSTIS. Moreover, she showed that three recently proposed CL-SC schemes in [18, 19, 24] not only do not satisfy the KSSTIS, but also they do not even satisfy the basic security requirements of a CL-SC scheme (according to the designed attacks). Finally, She enhanced the proposal in [16] which seems to be the best candidate (in the sense of both the security and the efficiency) among CL-SC schemes in the standard model to propose the first secure CL-SC scheme in the standard model which satisfies the KSSTIS. Security analysis show that the new scheme not only inherits the basic security requirement from the scheme in [16], but also satisfies the KSSTIS. The new proposal is the first CL-SC scheme in the standard model which guarantees the KSSTIS, too. Appendix A

References

[1] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.

[2] Sattam S Al-Riyami and Kenneth G Paterson. Certificateless public key cryptography. In *International conference on the theory and application of cryptology and information security*, pages 452–473. Springer, 2003.

[3] Yuliang Zheng. Digital signcryption or how to achieve cost (signature & encryption) ≪ cost (signature) + cost (encryption). In *Annual international cryptology conference*, pages 165–179. Springer, 1997.

[4] Manuel Barbosa and Pooya Farshim. Certificateless signcryption. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 369–372, 2008.

[5] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[6] Zhenhua Liu, Yupu Hu, Xiangsong Zhang, and Hua Ma. Certificateless signcryption scheme in the standard model. *Information Sciences*, 180(3):452–464, 2010.

[7] S Sharmila Deva Selvi, S Sree Vivek, and C Pandu Rangan. Security weaknesses in two certificateless signcryption schemes. *IACR Cryptol. ePrint Arch.*, 2010:92, 2010.

[8] Jian Weng, Guoxiang Yao, Robert H Deng, Min-Rong Chen, and Xiangxue Li. Cryptanalysis of a certificateless signcryption scheme in the standard model. *Information Sciences*, 181(3):661–667, 2011.

[9] Songqin Miao, Futai Zhang, Sujuan Li, and Yi Mu. On security of a certificateless signcryption scheme. *Information Sciences*, 232:475–481, 2013.

[10] Zhengping Jin, Qiaoyan Wen, and Hua Zhang. A supplement to liu et al.'s certificateless signcryption scheme in the standard model. *IACR Cryptol. ePrint Arch.*, 2010:252, 2010.

[11] Hu Xiong. Toward certificateless signcryption scheme without random oracles. *IACR Cryptol. ePrint Arch.*, 2014:162, 2014.

[12] Lin Cheng and Qiaoyan Wen. An improved certificateless signcryption in the standard model. *Int. J. Netw. Secur.*, 17(3):229–237, 2015.

[13] Xiao Zheng and Xudong Li. An efficient certificateless signcryption in the standard model. In *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 199–205. IEEE, 2016.

[14] Caixue Zhou, Guangyong Gao, and Zongmin Cui. Certificateless signcryption in the standard model. *Wireless Personal Communications*, 92(2):495–513, 2017.

[15] Parvin Rastegari and Mehdi Berenjkoub. An improved certificateless signcryption scheme. In *2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, pages 106–111. IEEE, 2016.

[16] Parvin Rastegari and Mehdi Berenjkoub. An efficient certificateless signcryption scheme in the standard model. *ISeCure*, 9(1), 2017.

[17] Ming Luo and Yuwei Wan. An enhanced certificateless signcryption in the standard model. *Wireless Personal Communications*, 98(3):2693–2709, 2018.

[18] ZHOU Caixue. Certificateless signcryption scheme without random oracles. *Chinese Journal of Electronics*, 27(5):1002–1008, 2018.

[19] Shan Shan. An efficient certificateless signcryption scheme without random oracles. *International Journal of Electronics and Information Engineering*, 11(1):9–15, 2019.

[20] Parvin Rastegari, Willy Susilo, and Mohammad Dakhlalian. Efficient certificateless signcryption in the standard model: Revisiting luo and wan's scheme from wireless personal communications (2018). *The Computer Journal*, 62(8):1178–1193, 2019.

[21] Parvin Rastegari and Mohammad Dakhilalian. Cryptanalysis of a certificateless signcryption scheme. In *2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, pages 67–71. IEEE, 2019.

[22] Yumin Yuan. Security analysis of an enhanced certificateless signcryption in the standard model. *Wireless Personal Communications*, pages 1–8, 2020.

[23] Xi-Jun Lin, Lin Sun, Zhen Yan, Xiaoshuai Zhang, and Haipeng Qu. On the security of a certificateless signcryption with known session-specific temporary information security in the standard model. *The Computer Journal*, 63(8):1259–1262, 2020.

[24] Insaf Ullah, Noor Ul Amin, Mahdi Zareei, Asim Zeb, Hizbullah Khattak, Ajab Khan, and Shidrokh Goudarzi. A lightweight and provable secured certificateless signcryption approach for crowdsourced iiot applications. *Symmetry*, 11(11):1386, 2019.

[25] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In *International Workshop on Public Key Cryptography*, pages 277–290. Springer, 2004.

[26] Marc Girault. Self-certified public keys. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 490–497. Springer, 1991.

[27] Yi-Fan Tseng, Chun-I Fan, and Ching-Wen Chen. Top-level secure certificateless signature scheme in the standard model. *IEEE Systems Journal*, 13(3):2763–2774, 2019.

[28] Wenjie Yang, Shangpeng Wang, Wei Wu, and Yi Mu. Top-level secure certificateless signature against malicious-but-passive kgc. *IEEE Access*, 7:112870–112878, 2019.

A Proof of Lemma 1

Let $\mathcal{A}_I$ be a $(t_I\varepsilon_I, q_{PK}, q_d, q_{RPK}, q_{SK}, q_{SC}, q_{USC})$-type I adversary, with the ability of winning Game 1 to break the IND-CCA of the proposed scheme. Then one can build a simulator $\mathcal{B}$ which can apply $\mathcal{A}_I$ as a sub-routine to solve an instance of a $(\mathcal{S}_1, 3)$-DBDHE-Set problem in time at most t with a probability at least ε. It contradicts the (t, ε)-$(\mathcal{S}_1, 3)$-DBDHE-Set assumption in (G_1, G_2). Let G_1 and G_2 be two multiplicative cyclic groups of a large prime order p, g be a random generator of G_1 and $e : G_1 \times G_1 \longrightarrow G_2$ be a bilinear pairing. Suppose that one gives $(g \in G_1, C = g^a \in G_1, X \in G_2)$ to $\mathcal{B}$ as a random $(\mathcal{S}_1, 3)$-DBDHE-Set challenge and $\mathcal{B}$ should return $\beta = 1$, if he decides that $X = e(g, g)^{a^3}$ and $\beta = 0$, otherwise. To this goal, $\mathcal{B}$ applies $\mathcal{A}_I$ as a sub-routine, simulates the challenger $\mathcal{C}$ (in Game 1) and responds

to $\mathcal{A}_I$'s queries from Public-Key, Partial-Private-Key, Replace-Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles. Firstly, $\mathcal{B}$ generates a list $\mathcal{L} = \{(ID_u, d_u, x_u, PK_u, Sk_u, state_u = 0)\}$ which is initially empty. Consequently, $\mathcal{B}$ executes Game 1 with $\mathcal{A}_I$ as follows:

- **Initialization:** Suppose that $l_u = 2(q_d + q_{SK} + q_{SC} + q_{USC} + 1)$ and $l_m = 2q_{USC}$ with assumptions $l_u(n_u+1) < p$ and $l_m(n_m+1) < p$. $\mathcal{B}$ chooses $k_u \in_R \{0, 1, \ldots, n_u\}$ and $k_m \in_R \{0, 1, \ldots, n_m\}$ (Note that the assumptions $l_u(n_u + 1) < p$ and $l_m(n_m + 1) < p$ imply $0 \leq k_u l_u < p$ and $0 \leq k_m l_m < p$, respectively). Moreover, $\mathcal{B}$ selects $x', x_1, \ldots, x_{n_u} \in_R \mathbb{Z}_{l_u}$, $y', y_1, \ldots, y_{n_u} \in_R \mathbb{Z}_p$, $z', z_1, \ldots, z_{n_m} \in_R \mathbb{Z}_{l_m}$ and $w', w_1, \ldots, w_{n_m} \in_R \mathbb{Z}_p$. $\mathcal{B}$ keeps these values secret and sets:

$$
\begin{aligned}
&g_1 = C = g^a,\\
&u' = g_1^{-k_u l_u + x'} g^{y'},\\
&u_i = g_1^{x_i} g^{y_i} \ (i = 1, 2, \ldots, n_u),\\
&v' = g_1^{-k_m l_m + z'} g^{w'},\\
&v_j = g_1^{z_j} g^{w_j} \ (j = 1, 2, \ldots, n_m).
\end{aligned}
$$

Furthermore, $\mathcal{B}$ calculates $T = e(g_1, g_1)$ and chooses three collision resistant hash functions $H_u : \{0,1\}^* \longrightarrow \{0,1\}^{n_u}$, $H_1 : G_2 \times G_1^4 \times \{0,1\}^* \longrightarrow \{0,1\}^{n_m}$ and $H_2 : G_1^3 \times \{0,1\}^* \longrightarrow \mathbb{Z}_p^*$. Finally, $\mathcal{B}$ sends $Params = \{G_1, G_2, p, e, g, g_1, T, u', v', U = (u_i)_{i=1}^{n_u}, V = (v_j)_{j=1}^{n_m}, H_1, H_2, H_u\}$ to $\mathcal{A}_I$. From the view of $\mathcal{A}_I$, all distributions are similar to them in the real world. We define the following functions to follow the proof more easily:

$$
\begin{aligned}
F(u) &= x' + \sum_{i \in \mathcal{U}_u} x_i - k_u l_u,\\
J(u) &= y' + \sum_{i \in \mathcal{U}_u} y_i,\\
K(m) &= z' + \sum_{j \in \mathcal{M}_m} z_j - k_m l_m,\\
L(m) &= w' + \sum_{j \in \mathcal{M}_m} w_j,
\end{aligned}
$$

where $\mathcal{U}_u$ and $\mathcal{M}_m$ are defined as described in the proposed scheme. By These settings, we have:

$$
\begin{aligned}
u' \prod_{i \in \mathcal{U}_u} u_i &= g_1^{F(u)} g^{J(u)},\\
v' \prod_{j \in \mathcal{M}_m} v_j &= g_1^{K(m)} g^{L(m)}.
\end{aligned}
$$

Note that $\mathcal{B}$ doesn't know $msk = g^{a^2}$ and must respond to $\mathcal{A}_I$'s queries in Game 1 without the knowledge of msk.

- **Phase 1 Queries:** In this step, $\mathcal{B}$ responds to $\mathcal{A}_I$'s queries from Public-Key, Partial-Private-Key, Replace-Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles as follows:
  - Public-Key-Query: When $\mathcal{A}_I$ requests the public key of a user u, i.e. PK_u, $\mathcal{B}$ firstly checks the list $\mathcal{L}$ to find it. If PK_u exists in $\mathcal{L}$, $\mathcal{B}$ returns it to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ executes the SetPuK algorithm to create PK_u and sends it to $\mathcal{A}_I$. Furthermore, $\mathcal{B}$ inserts PK_u and its corresponding x_u in $\mathcal{L}$.
  - Partial-Private-Key-Query: When $\mathcal{A}_I$ requests the partial private key of a user u, i.e. d_u, $\mathcal{B}$ firstly checks the list $\mathcal{L}$ to find it. If d_u exists in $\mathcal{L}$, $\mathcal{B}$ returns it to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ tries to create d_u without the knowledge of the master secret key as follows:
    - If $F(u) = 0 \bmod p$, $\mathcal{B}$ aborts.
    - If $F(u) \neq 0 \bmod p$, $\mathcal{B}$ randomly selects $r \in \mathbb{Z}_p^*$ and creates d_u as follows:

      $$
      \begin{aligned}
      d_u &= (d_{u,1}, d_{u,2})\\
      &= (g_1^{-\frac{J(u)}{F(u)}} (u' \prod_{i \in \mathcal{U}_u} u_i)^r, g_1^{-\frac{1}{F(u)}} g^r).
      \end{aligned}
      $$

      Afterwards, $\mathcal{B}$ sends d_u to $\mathcal{A}_I$ and inserts it in $\mathcal{L}$.
      We can easily check that:

      $$
      \begin{aligned}
      &g_1^{-J(u)/F(u)} (u' \prod_{i \in \mathcal{U}_u} u_i)^r = g^{\alpha^2} (u' \prod_{i \in \mathcal{U}_u} u_i)^{\tilde{r}},\\
      &g_1^{-1/F(u)} g^r = g^{\tilde{r}},
      \end{aligned}
      $$

      where $\tilde{r} = r - a/F(u)$. So, d_u which is created by $\mathcal{B}$, has the correct construction.
  - Replace-Public-Key-Query: Suppose that $\mathcal{A}_I$ requests to replace the public key of a user u, i.e. $PK_u = (PK_{u,1}, PK_{u,2})$ corresponding to x_u, with a new public key $PK'_u = (PK'_{u,1}, PK'_{u,2})$, corresponding to x'_u. $\mathcal{B}$ firstly checks the equality $e(PK'_{u,1}, PK'_{u,2}) = T$ holds or not. If the equality holds, $\mathcal{B}$ replaces (x_u, PK_u) with (x'_u, PK'_u) in the list $\mathcal{L}$. If there is not any (x_u, PK_u) corresponding to the user u in $\mathcal{L}$, $\mathcal{B}$ directly inserts $(x_u, PK_u) = (x'_u, PK'_u)$ in $\mathcal{L}$. After this replacement, $\mathcal{B}$ assigns $state_u = 1$.
  - Private-Key-Query: When $\mathcal{A}_I$ requests the private key of a user u, i.e. SK_u, $\mathcal{B}$ firstly checks the list $\mathcal{L}$ to find it. If SK_u exists in $\mathcal{L}$, $\mathcal{B}$ returns it to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ searches $\mathcal{L}$ for d_u. If d_u exists in $\mathcal{L}$, $\mathcal{B}$ picks it, otherwise $\mathcal{B}$ acts as follows:

– If $F(u) = 0 \bmod p$, $\mathcal{B}$ aborts.
– If $F(u) \neq 0 \bmod p$, $\mathcal{B}$ generates d_u such that explained in responding to Partial-Private-Key-Query.

Then, $\mathcal{B}$ gets (x_u, PK_u) ($\mathcal{B}$ picks (x_u, PK_u) from $\mathcal{L}$ if exists and the corresponding $state_u = 0$, otherwise $\mathcal{B}$ produces (x_u, PK_u) by executing the SetPuK algorithm). Then $\mathcal{B}$ can produce SK_u by executing the SetPrK algorithm, by the use of x_u and d_u. So $\mathcal{B}$ produces SK_u, sends it to $\mathcal{A}_I$ and also inserts it in $\mathcal{L}$.

- Signcrypt-Query: When $\mathcal{A}_I$ requests for a signcryption of a message m from a sender A to a receiver B, $\mathcal{B}$ obtains the private key of the sender SK_A (as explained in responding to the Private-Key-Query) and produces a signcryption $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ on m by executing the SC algorithm. Then $\mathcal{B}$ sends δ to $\mathcal{A}_I$. If $\mathcal{B}$ cannot simulate SK_A (i.e. $F(A) = 0 \bmod p$), $\mathcal{B}$ aborts the simulation.
- Unigncrypt-Query: When $\mathcal{A}_I$ requests for an unsigncryption of $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ from A to B, $\mathcal{B}$ firstly executes the verification part of the USC algorithm according to Eq. (9). If the verification fails, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ obtains m as follows:
  – If $state_B = 0$ (i.e. PK_B has never been replaced), $\mathcal{B}$ checks whether SK_B exists in the list $\mathcal{L}$ or not. If so, $\mathcal{B}$ picks it. Otherwise, $\mathcal{B}$ obtains SK_B such that explained in responding to the Private-Key-Query (by the assumption of $F(B) \neq 0 \bmod p$). Afterwards, $\mathcal{B}$ executes the unsigncrypt part of the USC algorithm according to Eq. (10) to obtain m and sends it to $\mathcal{A}_I$.
  – If $state_B = 1$ (i.e. PK_B has been replaced), $\mathcal{B}$ acts as follows:
    If $F(B) = 0 \bmod p$, $\mathcal{B}$ aborts.
    If $F(B) \neq 0 \bmod p$, $\mathcal{B}$ firstly obtains $g_1^{r_1}$ as follows:

$$g_1^{r_1} = \left(\frac{\delta_3}{\delta_2^{J(B)}}\right)^{\frac{1}{F(B)}}.$$

Note that:

$$\begin{aligned}
&\left(\frac{\delta_3}{\delta_2^{J(B)}}\right)^{\frac{1}{F(B)}} \\
&= \left(\frac{(u' \prod_{i \in \mathcal{U}_B} u_i)^{r_1}}{g^{r_1 \cdot J(B)}}\right)^{\frac{1}{F(B)}} \\
&= \left(\frac{(g_1^{F(B)} g^{J(B)})^{r_1}}{g^{r_1 \cdot J(B)}}\right)^{\frac{1}{F(B)}} \\
&= g_1^{r_1}.
\end{aligned}$$

Afterwards, $\mathcal{B}$ retrieves x_B corresponding to PK_B from $\mathcal{L}$ (Note that PK_B is a replaced public key). Then $\mathcal{B}$ extracts m as follows:

$$m = \frac{\delta_1}{e(g_1^{r_1} PK_{A,1}, g_1^{x_B^2})},$$

and sends it to $\mathcal{A}_I$.

- **Challenge:** In this step, $\mathcal{A}_I$ chooses two identities ID_{A^*} and ID_{B^*} and two equal length messages $m_0, m_1 \in G_2$ as the challenge ($\mathcal{A}_I$ has never sent a Private-Key-Query for B^*). Then $\mathcal{A}_I$ sends $\{ID_{A^*}, ID_{B^*}\}$ and $\{m_0, m_1\}$ to $\mathcal{C}$ (according to Game 1). $\mathcal{B}$ simulates $\mathcal{C}$ as follows:
  - If $F(B^*) \neq 0 \bmod p$ or $F(A^*) = 0 \bmod p$, $\mathcal{B}$ aborts.
  - If $F(B^*) = 0 \bmod p$ and $F(A^*) \neq 0 \bmod p$, $\mathcal{B}$ picks a bit γ by flipping a fair coin and produces a signcryption on m_γ from A^* to B^* as follows.
    Let $PK_{A^*} = (g_1^{x_{A^*}}, g_1^{1/x_{A^*}})$ and $PK_{B^*} = (g_1^{x_{B^*}}, g_1^{1/x_{B^*}})$ be the current public keys of A^* and B^*, respectively. Remind that $(g, C = g^a, X)$ is the input of the $(\mathcal{S}_1, 3)$-DBDHE-Set problem which $\mathcal{B}$ is trying to solve it. $\mathcal{B}$ retrieves x_{A^*} and x_{B^*} and assigns:

$$\begin{aligned}
&\delta_1^* = m_\gamma . X^{x_{B^*}^2} . e(PK_{B^*,1}, PK_{B^*,1})^{x_{A^*}}, \\
&\delta_2^* = C = g_1, \\
&\delta_3^* = C^{J(B^*)}, \\
&\delta_4^* = (C^{x_{A^*}^2})^{\frac{-1}{F(A^*)}} g^{t^*}, \quad t^* \in_R \mathbb{Z}_p^*.
\end{aligned}$$

Let $\mathcal{M}_{m_\gamma} = \{j | M_\gamma[j] = 1, j = 1, 2, \ldots, n_m\}$, where $M_\gamma = H_1(\delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*, PK_{B^*,1} ID_{B^*})$ and $h^* = H_2(PK_{A^*,1}, \delta_2^*, \delta_4^*, ID_{A^*}, M_\gamma)$.
  – If $K(m_\gamma) + x_{A^*} h^* \neq 0 \bmod p$, $\mathcal{B}$ aborts.
  – If $K(m_\gamma) + x_{A^*} h^* = 0 \bmod p$, $\mathcal{B}$ assigns:

$$\delta_5^* = (A^{x_{A^*}^2})^{\frac{-J(A^*)}{F(A^*)}} (u' \prod_{i \in \mathcal{U}_{A^*}} u_i)^{t^*} A^{L(m_\gamma)},$$

and sends $\delta^* = (\delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*, \delta_5^*)$ to $\mathcal{A}_I$.
It is straightforward to check that if $X = e(g, g)^{a^3}$ (i.e. $(g, A = g^a, X)$ is a valid $(\mathcal{S}_1, 3)$-DBDHE-Set tuple), δ^* is a valid signcryption on m_γ by considering α, r_{A^*}, r'_{A^*}, r_1 and r_2 in the proposed scheme as follows:

$$\begin{aligned}
&r_1 = a, \\
&\alpha = a \text{ (i.e. } msk = g^{a^2} \text{ and } \\
&g_1 = g^a = A), \\
&r'_{A^*} + r_2 = t^*,
\end{aligned}$$

$$r_{A^*} = -a/F(A^*).$$

Otherwise, (if X is a random element of G_2 and not equal to $e(g,g)^{a^3}$), δ^* is a random tuple which is not a valid signcryption neither for m_0 nor for m_1.

- **Phase 2 Queries:** $\mathcal{A}_I$ sends queries to the Public-Key, Partial-Private-Key, Replace-Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles again and $\mathcal{B}$ responds to them similar to that explained in the Phase 1 Queries step. Note that $\mathcal{A}_I$ is not permitted to send an unsigncryption query on δ^* from A^* to B^* unless PK_{A^*} or PK_{B^*} used to signcrypt m_γ, has been replaced after the challenge was issued.
- **Guess:** In thi step, $\mathcal{A}_I$ returns a guess γ^* of γ.

  Finally, when Game 1 between $\mathcal{A}_I$ and $\mathcal{B}$ terminates, $\mathcal{B}$ acts as follows:
  - If the simulation is aborted in any steps, $\mathcal{B}$ randomly selects its guess β' of β.
  - Otherwise, if $\gamma^* = \gamma$, $\mathcal{B}$ outputs a guess $\beta' = 1$, implying that $X = e(g,g)^{a^3}$, else $\mathcal{B}$ outputs $\beta' = 0$ to the $(\mathcal{S}_1, 3)$-DBDHE-Set problem.

Time Analysis: According to the above descriptions, $\mathcal{B}$ requires the time:

$$\begin{aligned} t \leq t_I + order(((q_d + q_{SK} + q_{SC} + q_{USC})n_u \\ + (q_{SC} + q_{USC})n_m)T_M \\ + (q_{PK} + q_d + q_{SK} + q_{SC} + q_{USC})T_E \\ + (q_{SC} + q_{USC})T_P), \end{aligned}$$

for solving the problem.

Probability Analysis: Let $\Pr[\mathcal{B}\text{ wins}]$ be the success probability of $\mathcal{B}$ in solving the $(\mathcal{S}_1, 3)$-DBDHE-Set problem and $\Pr[\mathcal{A}_I\text{ wins}]$ be the success probability of $\mathcal{A}_I$ in Game 1. Note that if the simulation is aborted in any steps, $\mathcal{B}$ randomly chooses its guess β' of β and so $\Pr[\mathcal{B}\text{ wins}] = \frac{1}{2}$. By the assumption of $\Pr[\mathcal{A}_I\text{ wins}] \geq \frac{1}{2} + \varepsilon_I$, we have:

$$\begin{aligned} \Pr[\mathcal{B}\text{ wins}] &= \Pr[\mathcal{B}\text{ wins}|\text{abort}]\Pr[\text{abort}] \\ &\quad + \Pr[\mathcal{B}\text{ wins}|\overline{\text{abort}}]\Pr[\overline{\text{abort}}] \\ &= \frac{1}{2}\Pr[\text{abort}] + \Pr[\mathcal{A}_I\text{ wins}]\Pr[\overline{\text{abort}}] \\ &\geq \frac{1}{2}(1 - \Pr[\overline{\text{abort}}]) + (\frac{1}{2} + \varepsilon_I)\Pr[\overline{\text{abort}}] \\ &= \frac{1}{2} + \varepsilon_I\Pr[\overline{\text{abort}}] \end{aligned}$$

$\mathcal{B}$ will not abort if all the following independent events happen:

- E_1: $F(A^*) \neq 0 \bmod p$, and $F(u) \neq 0 \bmod p$ for all Partial-Private-Key, Private-Key, Signcrypt and Unsigncrypt queries.
- E_2: $F(B^*) = 0 \bmod p$.
- E_3: $K(m_\gamma) + x_{A^*}h^* = 0 \bmod p$.

It is easy to see that:

$$\Pr[F(u) = 0 \bmod p] = \frac{1}{l_u(n_u+1)},$$

and:

$$\Pr[K(m_\gamma) + x_{A^*}h^* = 0 \bmod p] = \frac{1}{l_m(n_m+1)}.$$

So:

$$\begin{aligned} &\Pr[\overline{\text{abort}}] \geq \Pr[E_1 \bigcap E_2 \bigcap E_3] = \Pr[E_1].\Pr[E_2].\Pr[E_3] \\ &\geq (1 - \frac{q_d + q_{SK} + q_{SC} + q_{USC} + 1}{l_u(n_u+1)}).\frac{1}{l_u(n_u+1)l_m(n_m+1)} \\ &\geq (1 - \frac{q_d + q_{SK} + q_{SC} + q_{USC} + 1}{l_u}).\frac{1}{l_u(n_u+1)l_m(n_m+1)} \\ &= \frac{1}{8q_{USC}(q_d + q_{SK} + q_{SC} + q_{USC} + 1)(n_u+1)(n_m+1)}, \end{aligned}$$

where the rightmost equality is implied from $l_u = 2(q_d + q_{SK} + q_{SC} + q_{USC} + 1)$ and $l_m = 2q_{USC}$. Finally we have:

$$\begin{aligned} \Pr[\mathcal{B}\text{ wins}] \geq \frac{1}{2} \\ + \frac{\varepsilon_I}{8q_{USC}(q_d + q_{SK} + q_{SC} + q_{USC} + 1)(n_m+1)(n_u+1)}. \end{aligned}$$

In other words, if $\mathcal{A}_I$ wins Game 1 with a non-negligible advantage ε_I (i.e. guesses γ correctly with probability at least $\frac{1}{2} + \varepsilon_I$ for a non-negligible value of ε_I), then $\mathcal{B}$ can solve an instance of the $(\mathcal{S}_1, 3)$-DBDHE-Set problem with a non-negligible advantage ε (i.e. guess β correctly with probability at least $\frac{1}{2}+\varepsilon$), where

$$\varepsilon \geq \frac{\varepsilon_I}{8q_{USC}(q_d + q_{SK} + q_{SC} + q_{USC} + 1)(n_m+1)(n_u+1)},$$

which is a contradiction with the $(\mathcal{S}_1, 3)$-DBDHE-Set assumption in complexity theory.

B Proof of Lemma 2

Let $\mathcal{A}_{II}$ be a $(t_{II}, \varepsilon_{II}, q_{PK}, q_{SK}, q_{SC}, q_{USC})$-type II adversary, with the ability of winning Game 2 to break the IND-CCA of the proposed scheme. Then one can build a simulator $\mathcal{B}$ which can apply $\mathcal{A}_{II}$ as a sub-routine to solve an instance of a $(\mathcal{S}_2, 5)$-DBDHE-Set problem in time at most t with a probability at least ε. It contradicts the (t,ε)-$(\mathcal{S}_2, 5)$-DBDHE-Set assumption in (G_1, G_2). Let G_1 and G_2 be two multiplicative cyclic groups of a large prime order p, g be a random generator of G_1 and $e : G_1 \times G_1 \longrightarrow G_2$ be a bilinear pairing. Suppose that one gives $(C = h \in G_1, D = h^a \in G_1, E = h^{a^2} \in G_1, X \in G_2)$ to $\mathcal{B}$ as a random $(\mathcal{S}_2, 5)$-DBDHE-Set challenge and $\mathcal{B}$ should return $\beta = 1$, if he decides that $X = e(h,h)^{a^5}$ and $\beta = 0$, otherwise (Suppose that $D = h^a$ is a generator of G_1). To this goal, $\mathcal{B}$ applies $\mathcal{A}_{II}$ as a sub-routine, simulates the challenger $\mathcal{C}$ (in Game 2) and responds

to $\mathcal{A}_{II}$'s queries from Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles. Firstly, $\mathcal{B}$ generates a list $\mathcal{L} = \{(ID_u, x_u, PK_u, Sk_u)\}$ which is initially empty. Consequently, $\mathcal{B}$ executes Game 2 with $\mathcal{A}_{II}$ as follows:

- **Initialization:** Suppose that $l_u = 2(q_{SK} + q_{SC} + q_{USC} + 1)$ and $l_m = 2q_{USC}$ with assumptions $l_u(n_u + 1) < p$ and $l_m(n_m + 1) < p$. $\mathcal{B}$ chooses a random $\alpha \in_R \mathbb{Z}_p^*$ as the master secret key. In addition, $\mathcal{B}$ chooses $k_u, k_m, x', x_1, \ldots, x_{n_u}, y', y_1, \ldots, y_{n_u}, z', z_1, \ldots, z_{n_m}, w', w_1, \ldots, w_{n_m}$ such that explained in the proof of Lemma 1 and sets:

$$
\begin{aligned}
&g = D = h^a,\\
&g_1 = g^\alpha = D^\alpha\\
&u' = E^{-k_u l_u + x'} D^{y'},\\
&u_i = E^{x_i} D^{y_i}\ (i = 1, 2, \ldots, n_u),\\
&v' = E^{-k_m l_m + z'} D^{w'},\\
&v_j = E^{z_j} D^{w_j}\ (j = 1, 2, \ldots, n_m).
\end{aligned}
$$

  Furthermore, $\mathcal{B}$ calculates $T = e(g_1, g_1)$ and chooses three collision resistant hash functions $H_u : \{0,1\}^* \longrightarrow \{0,1\}^{n_u}$, $H_1 : G_2 \times G_1^4 \times \{0,1\}^* \longrightarrow \{0,1\}^{n_m}$ and $H_2 : G_1^3 \times \{0,1\}^* \longrightarrow \mathbb{Z}_p^*$. Finally, $\mathcal{B}$ sends $Params = \{G_1, G_2, p, e, g, g_1, T, u', v', U = (u_i)_{i=1}^{n_u}, V = (v_j)_{j=1}^{n_m}, H_1, H_2, H_u\}$ and α to $\mathcal{A}_{II}$. Define four functions $F(u)$, $J(u)$, $K(m)$ and $L(m)$ similar to that explained in the proof of Lemma 1. By these assignments we have:

$$
\begin{aligned}
u' \prod_{i \in \mathcal{U}_u} u_i &= E^{F(u)} D^{J(u)},\\
v' \prod_{j \in \mathcal{M}_m} v_j &= E^{K(m)} D^{L(m)}.
\end{aligned}
$$

  Note that (in contrast to the proof of Lemma 1) $\mathcal{B}$ knows $msk = g^{\alpha^2}$ and must respond to $\mathcal{A}_{II}$'s queries in Game 2 by this fact.
- **Phase 1 Queries:** In this step, $\mathcal{B}$ responds to $\mathcal{A}_{II}$'s queries from Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles as follows:
  - Public-Key-Query: When $\mathcal{A}_{II}$ requests the public key of a user u, i.e. PK_u, $\mathcal{B}$ firstly checks the list $\mathcal{L}$ to find it. If PK_u exists in $\mathcal{L}$, $\mathcal{B}$ returns it to $\mathcal{A}_{II}$. Otherwise, $\mathcal{B}$ picks a random $x_u \in_R \mathbb{Z}_p^*$ and acts as follows:
    - If $F(u) = 0 \bmod p$, $\mathcal{B}$ assigns $PK_u = (PK_{u,1}, PK_{u,2}) = (E^{\alpha x_u}, C^{\alpha / x_u})$. Note that by this setting, the real secret value of the user u is ax_u which is unknown to $\mathcal{B}$ (as $\mathcal{B}$ doesn't know a).
    - If $F(u) \neq 0 \bmod p$, $\mathcal{B}$ assigns $PK_u = (PK_{u,1}, PK_{u,2}) = (g_1^{x_u}, g_1^{1/x_u})$. Note that by this setting, the real secret value of the user u is x_u which is known to $\mathcal{B}$

    Then, $\mathcal{B}$ sends PK_u to $\mathcal{A}_{II}$ and inserts PK_u and its corresponding x_u in $\mathcal{L}$.
  - Private-Key-Query: When $\mathcal{A}_{II}$ requests the private key of a user u, i.e. SK_u, $\mathcal{B}$ firstly checks the list $\mathcal{L}$ to find it. If SK_u exists in $\mathcal{L}$, $\mathcal{B}$ returns it to $\mathcal{A}_{II}$. Otherwise, $\mathcal{B}$ acts as follows:
    - If $F(u) = 0 \bmod p$, $\mathcal{B}$ aborts.
    - If $F(u) \neq 0 \bmod p$, $\mathcal{B}$ checks whether (x_u, PK_u) exists in $\mathcal{L}$ or not. If so, $\mathcal{B}$ picks it. Otherwise, $\mathcal{B}$ creates (x_u, PK_u) such that explained in responding to the Public-Key-Query. Then $\mathcal{B}$ obtains d_u by executing the ExtPPK algorithm (note that $\mathcal{B}$ can run this algorithm since he knows α). Then $\mathcal{B}$ can generate SK_u by executing the SetPrK algorithm by the knowledge of x_u and d_u. So, $\mathcal{B}$ creates SK_u, sends it to $\mathcal{A}_{II}$ and inserts it in $\mathcal{L}$.
  - Signcrypt-Query: When $\mathcal{A}_{II}$ requests for a signcryption of a message m from a sender A to a receiver B, $\mathcal{B}$ obtains the private key of the sender SK_A (as explained in responding to the Private-Key-Query) and produces a signcryption $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ on m by executing the SC algorithm. Then $\mathcal{B}$ sends δ to $\mathcal{A}_{II}$. If $\mathcal{B}$ cannot simulate SK_A (i.e. $F(A) = 0 \bmod p$), $\mathcal{B}$ aborts the simulation.
  - Unigncrypt-Query: When $\mathcal{A}_{II}$ requests for an unsigncryption of $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ from A to B, $\mathcal{B}$ firstly executes the verification part of the USC algorithm according to Eq. (9). If the verification fails, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_{II}$. Otherwise, $\mathcal{B}$ checks whether SK_B exists in the list $\mathcal{L}$ or not. If so, $\mathcal{B}$ picks it. Otherwise, $\mathcal{B}$ obtains SK_B such that explained in responding to the Private-Key-Query (by the assumption of $F(B) \neq 0 \bmod p$). Afterwards, $\mathcal{B}$ executes the unsigncrypt part of the USC algorithm according to Eq. (10) to obtain m and sends it to $\mathcal{A}_{II}$.
- **Challenge:** In this step, $\mathcal{A}_{II}$ chooses two identities ID_{A^*} and ID_{B^*} and two equal length messages $m_0, m_1 \in G_2$ as the challenge ($\mathcal{A}_{II}$ has never sent a Private-Key-Query for B^*). Then $\mathcal{A}_{II}$ sends $\{ID_{A^*}, ID_{B^*}\}$ and $\{m_0, m_1\}$ to $\mathcal{C}$ (according to Game 2). $\mathcal{B}$ simulates $\mathcal{C}$ as follows:
  - If $F(B^*) \neq 0 \bmod p$ or $F(A^*) = 0 \bmod p$,

$\mathcal{B}$ aborts.

- If $F(B^*) = 0 \bmod p$ and $F(A^*) \neq 0 \bmod p$, $\mathcal{B}$ picks a bit γ by flipping a fair coin and produces a signcryption on m_γ from A^* to B^* as follows.
  Note that $PK_{A^*} = (g_1^{x_{A^*}}, g_1^{1/x_{A^*}})$ (since $F(A^*) \neq 0 \bmod p$) and $PK_{B^*} = (E^{\alpha x_{B^*}}, C^{\alpha/x_{B^*}})$ (since $F(B^*) = 0 \bmod p$). Remind that $(C = h \in G_1, D = h^a \in G_1, E = h^{a^2} \in G_1, X \in G_2)$ is the input of the $(\mathcal{S}_2, 5)$-DBDHE-Set problem which $\mathcal{B}$ is trying to solve it. $\mathcal{B}$ retrieves x_{A^*} and x_{B^*} and assigns:

$$\delta_1^* = m_\gamma . X^{(\alpha x_{B^*})^2} . e(PK_{B^*,1}, PK_{B^*,1})^{x_{A^*}},$$
$$\delta_2^* = E = h^{a^2},$$
$$\delta_3^* = E^{J(B^*)},$$
$$\delta_4^* = (E^{(\alpha x_{A^*})^2})^{\frac{-1}{F(A^*)}} g^{t^*}, \quad t^* \in_R \mathbb{Z}_p^*.$$

  Let $\mathcal{M}_{m_\gamma} = \{j | M_\gamma[j] = 1, j = 1, 2, \ldots, n_m\}$, where $M_\gamma = H_1(\delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*, PK_{B^*,1} ID_{B^*})$ and $h^* = H_2(PK_{A^*,1}, \delta_2^*, \delta_4^*, ID_{A^*}, M_\gamma)$.
  - If $K(m_\gamma) + x_{A^*} h^* \neq 0 \bmod p$, $\mathcal{B}$ aborts.
  - If $K(m_\gamma) + x_{A^*} h^* = 0 \bmod p$, $\mathcal{B}$ assigns:

$$\delta_5^* = (E^{(\alpha x_{A^*})^2})^{\frac{-J(A^*)}{F(A^*)}} (u' \prod_{i \in \mathcal{U}_{A^*}} u_i)^{t^*} E^{L(m_\gamma)},$$

    and sends $\delta^* = (\delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*, \delta_5^*)$ to $\mathcal{A}_{II}$.
    It is straightforward to check that if $X = e(h,h)^{a^5}$ (i.e. $(C = h \in G_1, D = h^a \in G_1, E = h^{a^2} \in G_1, X \in G_2)$ is a valid $(\mathcal{S}_2, 5)$-DBDHE-Set tuple), δ^* is a valid signcryption on m_γ by considering α, r_{A^*}, r'_{A^*}, r_1 and r_2 in the proposed scheme as follows:

$$r_1 = a,$$
$$r'_{A^*} + r_2 = t^*,$$
$$r_{A^*} = -\alpha^2 / a F(A^*).$$

    Otherwise, (if X is a random element of G_2 and not equal to $e(h,h)^{a^5}$), δ^* is a random tuple which is not a valid signcryption neither for m_0 nor for m_1.

- **Phase 2 Queries:** $\mathcal{A}_{II}$ sends queries to the Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles again and $\mathcal{B}$ responds to them similar to that explained in the Phase 1 Queries step. Note that $\mathcal{A}_{II}$ is not permitted to send an unsigncryption query on δ^* from A^* to B^*.
- **Guess:** In this step, $\mathcal{A}_{II}$ returns a guess γ^* of γ.

  Finally, when Game 2 between $\mathcal{A}_{II}$ and $\mathcal{B}$ terminates, $\mathcal{B}$ acts as follows:
  - If the simulation is aborted in any steps, $\mathcal{B}$ randomly selects its guess β' of β.
  - Otherwise, if $\gamma^* = \gamma$, $\mathcal{B}$ outputs a guess $\beta' = 1$, implying that $X = e(h,h)^{a^5}$, else $\mathcal{B}$ outputs $\beta' = 0$ to the $(\mathcal{S}_2, 5)$-DBDHE-Set problem.

Time and probability analysis are such that explained in the proof of Lemma 1, but we must consider t_{II} and ε_{II} instead of t_I and ε_I, respectively and $q_d = 0$, here.

C Proof of Lemma 3

Let $\mathcal{A}_I$ be a $(t_I, \varepsilon_I, q_{PK}, q_d, q_{RPK}, q_{SK}, q_{SC}, q_{USC})$-type I adversary, with the ability of winning Game 3 to break the EUF-CMA of the proposed scheme. Then one can build a simulator $\mathcal{B}$ which can apply $\mathcal{A}_I$ as a sub-routine to solve an instance of a 2-CDHE problem in time at most t with a probability at least ε. It contradicts the (t, ε)-2-CDHE assumption in G_1. Let G_1 be a multiplicative cyclic groups of a large prime order p and g be a random generator of G_1. Suppose that one gives $(g \in G_1, C = g^a \in G_1)$ to $\mathcal{B}$ as a random 2-CDHE challenge and $\mathcal{B}$ should return $g^{a^2} \in G_1$. To this goal, $\mathcal{B}$ applies $\mathcal{A}_I$ as a sub-routine, simulates the challenger $\mathcal{C}$ (in Game 3) and responds to $\mathcal{A}_I$'s queries from Public-Key, Partial-Private-Key, Replace-Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles. Firstly, $\mathcal{B}$ selects a multiplicative group G_2 of order p and a bilinear pairing $e : G_1 \times G_1 \longrightarrow G_2$. Then $\mathcal{B}$ generates a list $\mathcal{L} = \{(ID_u, d_u, x_u, PK_u, Sk_u, state_u = 0)\}$ which is initially empty. Consequently, $\mathcal{B}$ executes Game 3 with $\mathcal{A}_I$ as follows:

- **Initialization:** This step is similar to the Initialization step in the proof of Lemma 1, except that l_u is considered as $l_u = 2(q_d + q_{SK} + q_{SC} + q_{USC})$, here.
- **Queries:** This step is similar to the Phase 1 Queries step in the proof of Lemma 1.
- **Forgery** In this step (if the simulation is not aborted in any steps), $\mathcal{A}_I$ generates a valid signcryption $\delta^* = (\delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*, \delta_5^*)$ on a message m^* from A^* with public key $PK_{A^*} = (g_1^{x_{A^*}}, g_1^{1/x_{A^*}})$ to B^* with public key $PK_{B^*} = (g_1^{x_{B^*}}, g_1^{1/x_{B^*}})$. Let $\mathcal{M}_{m^*} = \{j | M^*[j] = 1, j = 1, 2, \ldots, n_m\}$, where $M^* = H_1(\delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*, PK_{B^*,1} ID_{B^*})$ and $h^* = H_2(PK_{A^*,1}, \delta_2^*, \delta_4^*, ID_{A^*}, M^*)$.
  - If $F(A^*) \neq 0 \bmod p$ or $K(m^*) + x_{A^*} h^* \neq$

$0 \bmod p$, $\mathcal{B}$ aborts.

- If $F(A^*) = 0 \bmod p$ and $K(m^*)+x_{A^*}h^* = 0 \bmod p$, $\mathcal{B}$ retrieves x_{A^*} and calculates:

$$g^{a^2} = \left(\frac{\delta_5^*}{(\delta_4^*)^{J(A^*)}(\delta_2^*)^{L(m^*)}}\right)^{\frac{1}{x_{A^*}^2}}.$$

Time Analysis: It is similar to the time analysis in the proof of Lemma 1.

Probability Analysis: Let $\Pr[\mathcal{B} \text{ wins}]$ be the success probability of $\mathcal{B}$ in solving the 2-CDHE problem and $\Pr[\mathcal{A}_I \text{ wins}]$ be the success probability of $\mathcal{A}_I$ in Game 3. By the assumption of $\Pr[\mathcal{A}_I \text{ wins}] \geq \varepsilon_I$, we have:

$$\begin{aligned}\Pr[\mathcal{B} \text{ wins}] &= \Pr[\overline{\text{abort}} \bigcap \mathcal{A}_I \text{ wins}] \\ &= \Pr[\mathcal{A}_I \text{ wins}].\Pr[\overline{\text{abort}}] \geq \varepsilon_I.\Pr[\overline{\text{abort}}]\end{aligned}$$

$\mathcal{B}$ will not abort if all the following independent events happen:

- E_1: $F(u) \neq 0 \bmod p$ for all Partial-Private-Key, Private-Key, Signcrypt and Unsigncrypt queries.
- E_2: $F(A^*) = 0 \bmod p$.
- E_3: $K(m_\gamma) + x_{A^*}h^* = 0 \bmod p$.

So we have:

$$\begin{aligned}&\Pr[\overline{\text{abort}}] \geq \Pr[E_1 \bigcap E_2 \bigcap E_3] = \Pr[E_1].\Pr[E_2].\Pr[E_3] \\ &\geq (1 - \frac{q_d + q_{SK} + q_{SC} + q_{USC}}{l_u(n_u+1)}).\frac{1}{l_u(n_u+1)l_m(n_m+1)} \\ &\geq (1 - \frac{q_d + q_{SK} + q_{SC} + q_{USC}}{l_u}).\frac{1}{l_u(n_u+1)l_m(n_m+1)} \\ &= \frac{1}{8q_{USC}(q_d + q_{SK} + q_{SC} + q_{USC})(n_u+1)(n_m+1)},\end{aligned}$$

where the rightmost equality is implied from $l_u = 2(q_d + q_{SK} + q_{SC} + q_{USC})$ and $l_m = 2q_{USC}$. Then we have:

$$\Pr[\mathcal{B} \text{ wins}] \geq \frac{\varepsilon_I}{8q_{USC}(q_d + q_{SK} + q_{SC} + q_{USC})(n_m+1)(n_u+1)}.$$

In other words, if $\mathcal{A}_I$ wins Game 3 with a non-negligible advantage ε_I (i.e. forges a valid signcryption with a non-negligible probability ε_I), then $\mathcal{B}$ can solve an instance of the 2-CDHE problem with a non-negligible probability ε, where $\varepsilon \geq \frac{\varepsilon_I}{8q_{USC}(q_d+q_{SK}+q_{SC}+q_{USC})(n_m+1)(n_u+1)}$, which is a contradiction with the 2-CDHE assumption in complexity theory.

D Proof of Lemma 4

Let $\mathcal{A}_{II}$ be a $(t_{II}, \varepsilon_{II}, q_{PK}, q_{SK}, q_{SC}, q_{USC})$-type II adversary, with the ability of winning Game 4 to break the EUF-CMA of the proposed scheme. Then one can build a simulator $\mathcal{B}$ which can apply $\mathcal{A}_{II}$ as a sub-routine to solve an instance of a 3-CDHE problem in time at most t with a probability at least ε. It contradicts the (t, ε)-3-CDHE assumption in G_1. Let G_1 be a multiplicative cyclic groups of a large prime order p and g be a random generator of G_1. Suppose that one gives $(C = h \in G_1, D = h^a \in G_1, E = h^{a^2} \in G_1)$ to $\mathcal{B}$ as a random 3-CDHE challenge and $\mathcal{B}$ should return $h^{a^3} \in G_1$. To this goal, $\mathcal{B}$ applies $\mathcal{A}_{II}$ as a sub-routine, simulates the challenger $\mathcal{C}$ (in Game 4) and responds to $\mathcal{A}_{II}$'s queries from Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles. Firstly, $\mathcal{B}$ selects a multiplicative group G_2 of order p and a bilinear pairing $e : G_1 \times G_1 \longrightarrow G_2$. Then $\mathcal{B}$ generates a list $\mathcal{L} = \{(ID_u, x_u, PK_u, Sk_u)\}$ which is initially empty. Consequently, $\mathcal{B}$ executes Game 4 with $\mathcal{A}_{II}$ as follows:

- **Initialization:** This step is similar to the Initialization step in the proof of Lemma 2, except that l_u is considered as $l_u = 2(q_{SK} + q_{SC} + q_{USC})$, here.
- **Queries:** This step is similar to the Phase 1 Queries step in the proof of Lemma 2.
- **Forgery** In this step (if the simulation is not aborted in any steps), $\mathcal{A}_{II}$ generates a valid signcryption $\delta^* = (\delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*, \delta_5^*)$ on a message m^* from A^* to B^*. Let $\mathcal{M}_{m^*} = \{j | M^*[j] = 1, j = 1, 2, \ldots, n_m\}$, where $M^* = H_1(\delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*, PK_{B^*,1} ID_{B^*})$ and $h^* = H_2(PK_{A^*,1}, \delta_2^*, \delta_4^*, ID_{A^*}, M^*)$.
  - If $F(A^*) \neq 0 \bmod p$ or $K(m^*) + x_{A^*}h^* \neq 0 \bmod p$, $\mathcal{B}$ aborts.
  - If $F(A^*) = 0 \bmod p$ and $K(m^*)+x_{A^*}h^* = 0 \bmod p$, $\mathcal{B}$ retrieves x_{A^*} and calculates:

$$h^{a^3} = \left(\frac{\delta_5^*}{(\delta_4^*)^{J(A^*)}(\delta_2^*)^{L(m^*)}}\right)^{\frac{1}{(\alpha x_{A^*})^2}}.$$

Note that as $F(A^*) = 0 \bmod p$, we have $PK_{A^*} = (E^{\alpha x_{A^*}}, C^{\alpha/x_{A^*}})$ according to that explained in responding to the Public-Key-Query in the proof of Lemma 2.

Time and probability analysis are such that explained in the proof of Lemma 3, but we must consider t_{II} and ε_{II} instead of t_I and ε_I, respectively and $q_d = 0$, here.

E Proof of Theorem 3

Let $\mathcal{A}$ be a $(t, \varepsilon, q_{PK}, q_{SK}, q_{SC}, q_{USC})$-adversary, with the ability of winning Game 5 to break the KSSTIS of the proposed scheme. Then one can build a simulator $\mathcal{B}$ which can apply $\mathcal{A}$ as a sub-routine to solve an instance of a $(\mathcal{S}_2, 5)$-CBDHE-Set problem in time at most t with a probability at least ε. It contradicts the (t, ε)-$(\mathcal{S}_2, 5)$-CBDHE-Set assumption in (G_1, G_2). Let G_1 and G_2 be two multiplicative cyclic groups of a large prime order p, g be a random generator of G_1 and $e : G_1 \times G_1 \longrightarrow G_2$ be a bilinear pairing. Suppose that one gives $(C = h \in$

$G_1, D = h^a \in G_1, E = h^{a^2} \in G_1$) to $\mathcal{B}$ as a random $(\mathcal{S}_2, 5)$-CBDHE-Set challenge and $\mathcal{B}$ should return $X = e(h,h)^{a^5}$ (Suppose that $D = h^a$ is a generator of G_1). To this goal, $\mathcal{B}$ applies $\mathcal{A}$ as a sub-routine, simulates the challenger $\mathcal{C}$ (in Game 5) and responds to $\mathcal{A}$'s queries from Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles. Firstly, $\mathcal{B}$ generates a list $\mathcal{L} = \{(ID_u, d_u, x_u, PK_u, Sk_u)\}$ which is initially empty. Consequently, $\mathcal{B}$ executes Game 5 with $\mathcal{A}$ as follows:

- **Initialization:** Suppose that $l_u = 2(q_d + q_{SK} + q_{SC} + q_{USC})$ with the assumption $l_u(n_u + 1) < p$. $\mathcal{B}$ chooses a random $\alpha \in_R \mathbb{Z}_p^*$ as the master secret key. In addition, $\mathcal{B}$ chooses $k_u, x', x_1, \ldots, x_{n_u}$ such that explained in the proof of Lemma 1 and sets $g = D = h^a$ and $g_1 = g^\alpha = D^\alpha$. Furthermore, $\mathcal{B}$ picks $u', u_1, \ldots, u_{n_u}, v', v_1, \ldots, v_{n_m} \in_R G_1$, calculates $T = e(g_1, g_1)$ and chooses three collision resistant hash functions $H_u : \{0,1\}^* \longrightarrow \{0,1\}^{n_u}$, $H_1 : G_2 \times G_1^4 \times \{0,1\}^* \longrightarrow \{0,1\}^{n_m}$ and $H_2 : G_1^3 \times \{0,1\}^* \longrightarrow \mathbb{Z}_p^*$. Finally, $\mathcal{B}$ sends $Params = \{G_1, G_2, p, e, g, g_1, T, u', v', U = (u_i)_{i=1}^{n_u}, V = (v_j)_{j=1}^{n_m}, H_1, H_2, H_u\}$ to $\mathcal{A}$. Define $F(u)$ such that explained in the proof of Lemma 1, i. e:

$$F(u) = x' + \sum_{i \in \mathcal{U}_u} x_i - k_u l_u.$$

- **Queries:** In this step, $\mathcal{B}$ responds to $\mathcal{A}$'s queries from Public-Key, Private-Key, Signcrypt and Unsigncrypt oracles as follows:
  - Public-Key-Query: When $\mathcal{A}$ requests the public key of a user u, i.e. PK_u, $\mathcal{B}$ firstly checks the list $\mathcal{L}$ to find it. If PK_u exists in $\mathcal{L}$, $\mathcal{B}$ returns it to $\mathcal{A}$. Otherwise, $\mathcal{B}$ picks a random $x_u \in_R \mathbb{Z}_p^*$ and acts as follows:
    - If $F(u) = 0 \bmod p$, $\mathcal{B}$ assigns $PK_u = (PK_{u,1}, PK_{u,2}) = (E^{\alpha x_u}, C^{\alpha/x_u})$. Note that by this setting, the real secret value of the user u is ax_u which is unknown to $\mathcal{B}$ (as $\mathcal{B}$ doesn't know a).
    - If $F(u) \neq 0 \bmod p$, $\mathcal{B}$ assigns $PK_u = (PK_{u,1}, PK_{u,2}) = (g_1^{x_u}, g_1^{1/x_u})$. Note that by this setting, the real secret value of the user u is x_u which is known to $\mathcal{B}$

    Then, $\mathcal{B}$ sends PK_u to $\mathcal{A}$ and inserts PK_u and its corresponding x_u in $\mathcal{L}$.
  - Partial-Private-Key-Query: When $\mathcal{A}$ requests the partial private key of a user u, i.e. d_u, $\mathcal{B}$ firstly checks the list $\mathcal{L}$ to find it. If d_u exists in $\mathcal{L}$, $\mathcal{B}$ returns it to $\mathcal{A}$. Otherwise, $\mathcal{B}$ obtains d_u by executing the ExtPPK algorithm (note that $\mathcal{B}$ can run this algorithm since he knows α). then $\mathcal{B}$ sends d_u to $\mathcal{A}$ and also inserts it in $\mathcal{L}$.
  - Private-Key-Query: When $\mathcal{A}$ requests the private key of a user u, i.e. SK_u, $\mathcal{B}$ firstly checks the list $\mathcal{L}$ to find it. If SK_u exists in $\mathcal{L}$, $\mathcal{B}$ returns it to $\mathcal{A}$. Otherwise, $\mathcal{B}$ acts as follows:
    - If $F(u) = 0 \bmod p$, $\mathcal{B}$ aborts.
    - If $F(u) \neq 0 \bmod p$, $\mathcal{B}$ checks whether (x_u, PK_u) exists in $\mathcal{L}$ or not. If so, $\mathcal{B}$ picks it. Otherwise, $\mathcal{B}$ creates (x_u, PK_u) such that explained in responding to the Public-Key-Query. Then $\mathcal{B}$ obtains d_u by executing the ExtPPK algorithm (note that $\mathcal{B}$ can run this algorithm since he knows α). Then $\mathcal{B}$ can generate SK_u by executing the SetPrK algorithm by the knowledge of x_u and d_u. So, $\mathcal{B}$ creates SK_u, sends it to $\mathcal{A}$ and inserts it in $\mathcal{L}$.
  - Signcrypt-Query: When $\mathcal{A}$ requests for a signcryption of a message m from a sender A to a receiver B, $\mathcal{B}$ obtains the private key of the sender SK_A (as explained in responding to the Private-Key-Query) and produces a signcryption $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ on m by executing the SC algorithm. Then $\mathcal{B}$ sends δ to $\mathcal{A}$. If $\mathcal{B}$ cannot simulate SK_A (i.e. $F(A) = 0 \bmod p$), $\mathcal{B}$ aborts the simulation.
  - Unigncrypt-Query: When $\mathcal{A}$ requests for an unsigncryption of $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ from A to B, $\mathcal{B}$ firstly executes the verification part of the USC algorithm according to Eq. (9). If the verification fails, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ checks whether SK_B exists in the list $\mathcal{L}$ or not. If so, $\mathcal{B}$ picks it. Otherwise, $\mathcal{B}$ obtains SK_B such that explained in responding to the Private-Key-Query (by the assumption of $F(B) \neq 0 \bmod p$). Afterwards, $\mathcal{B}$ executes the unsigncrypt part of the USC algorithm according to Eq. (10) to obtain m and sends it to $\mathcal{A}$.
- **Output:** In this step, $\mathcal{A}$ (who has a signcryption δ^* on a message m^* from A^* to B^* and knows r_1^*, r_2^*) returns a message m^{**}. Then $\mathcal{A}$ sends δ^*, m^{**} and r_1^* to $\mathcal{B}$.

  Finally, when Game 5 between $\mathcal{A}$ and $\mathcal{B}$ terminates, $\mathcal{B}$ acts as follows:
  - If the simulation is aborted in any steps, $\mathcal{B}$ aborts.
  - Otherwise:
    - If $F(A^*) \neq 0 \bmod p$ or $F(B^*) \neq 0$

mod p, $\mathcal{B}$ aborts.

– If $F(A^*) = 0 \bmod p$ and $F(B^*) = 0 \bmod p$, $\mathcal{B}$ retrieves x_{A^*} and x_{B^*}, then computes:

$$e(h,h)^{a^5} = \left(\frac{\delta_1^*}{m^{**}.e(PK_{B^*,1}, PK_{B^*,1})^{r_1^*}}\right)^{\frac{1}{\alpha^2 x_{B^*}^2 x_{A^*}}}, \tag{E.1}$$

as the output of the $(\mathcal{S}_2, 5)$-CBDHE-Set problem. Note that as $F(A^*) = 0 \bmod p$ and $F(B^*) = 0 \bmod p$, $PK_{A^*} = (PK_{A^*,1}, PK_{A^*,2}) = (E^{\alpha x_{A^*}}, C^{\alpha/x_{A^*}})$ and $PK_{B^*} = (PK_{B^*,1}, PK_{B^*,2}) = (E^{\alpha x_{B^*}}, C^{\alpha/x_{B^*}})$. In other words, the real secret values of A^* and B^* are respectively ax_{A^*} and ax_{B^*}, which are both unknown to $\mathcal{B}$. By these descriptions, it is easy to check the correctness of Eq. (E.1) as:

$$\begin{aligned}
&\left(\frac{\delta_1^*}{m^{**}.e(PK_{B^*,1}, PK_{B^*,1})^{r_1^*}}\right)^{\frac{1}{\alpha^2 x_{B^*}^2 x_{A^*}}} \\
&= \left(\frac{m^*.e(PK_{B^*,1}^{r_1^*+SK_{A^*,3}}, PK_{B^*,1})}{m^{**}.e(PK_{B^*,1}, PK_{B^*,1})^{r_1^*}}\right)^{\frac{1}{\alpha^2 x_{B^*}^2 x_{A^*}}} \\
&= e(PK_{B^*,1}^{SK_{A^*,3}}, PK_{B^*,1})^{\frac{1}{\alpha^2 x_{B^*}^2 x_{A^*}}} \\
&= e(PK_{B^*,1}, PK_{B^*,1})^{\frac{SK_{A^*,3}}{\alpha^2 x_{B^*}^2 x_{A^*}}} \\
&= e(E^{\alpha x_{B^*}}, E^{\alpha x_{B^*}})^{\frac{ax_{A^*}}{\alpha^2 x_{B^*}^2 x_{A^*}}} \\
&= e(h^{a^2 \alpha x_{B^*}}, h^{a^2 \alpha x_{B^*}})^{\frac{ax_{A^*}}{\alpha^2 x_{B^*}^2 x_{A^*}}} \\
&= e(h,h)^{a^5},
\end{aligned}$$

which shows the correctness of Eq. (E.1).

Time Analysis: According to the above descriptions, $\mathcal{B}$ requires the time:

$$\begin{aligned}
t \leq t' + order(&((q_d + q_{SK} + q_{SC} + q_{USC})n_u \\
&+ (q_{SC} + q_{USC})n_m)T_M \\
&+ (q_{PK} + q_d + q_{SK} + q_{SC} + q_{USC})T_E \\
&+ (q_{SC} + q_{USC})T_P),
\end{aligned}$$

for solving the problem.

Probability Analysis: Let $\Pr[\mathcal{B} \text{ wins}]$ be the success probability of $\mathcal{B}$ in solving the $(\mathcal{S}_2, 5)$-CBDHE-Set problem and $\Pr[\mathcal{A} \text{ wins}]$ be the success probability of $\mathcal{A}$ in Game 5. We have:

$$\begin{aligned}
\Pr[\mathcal{B} \text{ wins}] &= \Pr[\overline{\text{abort}} \bigcap \mathcal{A} \text{ wins}] \\
&= \Pr[\mathcal{A} \text{ wins}].\Pr[\overline{\text{abort}}] \geq \varepsilon'.\Pr[\overline{\text{abort}}]
\end{aligned}$$

$\mathcal{B}$ will not abort if all the following independent events happen:

- E_1: $F(u) \neq 0 \bmod p$ for all Partial-Private-Key, Private-Key, Signcrypt and Unsigncrypt queries.
- E_2: $F(A^*) = 0 \bmod p$.
- E_3: $F(B^*) = 0 \bmod p$.

It is easy to see that:

$$\Pr[F(u) = 0 \bmod p] = \frac{1}{l_u(n_u+1)},$$

So:

$$\begin{aligned}
\Pr[\overline{\text{abort}}] &\geq \Pr[E_1 \bigcap E_2 \bigcap E_3] = \Pr[E_1].\Pr[E_2].\Pr[E_3] \\
&\geq (1 - \frac{q_d + q_{SK} + q_{SC} + q_{USC}}{l_u(n_u+1)}).\frac{1}{l_u^2(n_u+1)^2} \\
&\geq (1 - \frac{q_d + q_{SK} + q_{SC} + q_{USC}}{l_u}).\frac{1}{l_u^2(n_u+1)^2} \\
&= \frac{1}{8(q_d + q_{SK} + q_{SC} + q_{USC})^2(n_u+1)^2},
\end{aligned}$$

where the rightmost equality is implied from $l_u = 2(q_d + q_{SK} + q_{SC} + q_{USC})$. Finally, we have:

$$\Pr[\mathcal{B} \text{ wins}] \geq \frac{\varepsilon'}{8(q_d + q_{SK} + q_{SC} + q_{USC})^2(n_u+1)^2}.$$

In other words, if $\mathcal{A}$ wins Game 5 with a non-negligible advantage ε' (i.e. returns m^* correctly with probability at least ε' for a non-negligible value of ε'), then $\mathcal{B}$ can solve an instance of the $(\mathcal{S}_2, 5)$-CBDHE-Set problem with a non-negligible advantage ε (i.e. compute $e(h,h)^{a^5}$ with probability at least ε), where $\varepsilon \geq \frac{\varepsilon'}{8(q_d+q_{SK}+q_{SC}+q_{USC})^2(n_u+1)^2}$, which is a contradiction with the $(\mathcal{S}_2, 5)$-CBDHE-Set assumption in complexity theory.

Parvin Rastegari received the B.Sc., M.Sc. and Ph.D. degrees in electrical engineering from the Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran, in 2008, 2011 and 2019, respectively. Since 2020, she has been with the Electrical and Computer Engineering Group, Golpayegan College of Engineering, Isfahan University of Technology, Golpayegan, Iran, as an assistant professor. Her current special fields of interest include information security, cryptographic protocols and security challenges in smart grids and internet of things.