Toward certificateless signcryption scheme without random oracles

Hu Xiong[*,a]

[a]Network and Data Security Key Laboratory of Sichuan Province, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 610054, P.R.China

Abstract

Signcryption is a useful paradigm which simultaneously offers both the functions of encryption and signature in a single logic step. It would be interesting to make signcryption certificateless to ease the heavy burden of certificate management in traditional public key cryptography (PKC) and solve the key escrow problem in Identity-based public key cryptography (ID-PKC). Most certificateless signcryption (CL-SC) schemes are constructed in the random oracle model instead of the standard model. By exploiting Bellare and Shoup's one-time signature, Hwang *et al.*'s certificateless encryption and Li *et al.*'s identity-based signcryption, this paper proposes a new CL-SC scheme secure in the standard model. It is proven that our CL-SC scheme satisfies semantic security and unforgeability against the outside adversary and malicious-but-passive key generation center (KGC) assuming the hardness of bilinear decision Diffie-Hellman (BDDH) and computational Diffie-Hellman (CDH) problems. Our security proofs do not depend on random oracles.

Key words: Information Security; Certificateless cryptography; Signcryption; Standard model

1. Introduction

Public key cryptography (PKC) has been widely accepted and applied since it can simplify the secret key distribution problem in symmetric cryptosystem and provide a means of digital signature [16]. In traditional PKC, every user owns a public/secret key pair where the public key is usually a random string. In this case, a digital certificate issued by a trusted certification authority (CA) is needed to guarantee the relationship between the public key and the identity of the user. In general, the heavy certificate management, including certificate distribution, revocation, storage and verification, is regarded to be expensive. To remove the heavy burden of certificate management, the notion of Identity-based public key cryptography (ID-PKC) has been incepted in the cryptography community [33, 40, 21]. ID-PKC, as distinct from conventional PKC, can eliminate the need of certificates since the public key of the user can be obtained directly from its intrinsic identity information such as E-mail address or driving license number in the ID-PKC

*Corresponding author.
Email address: `xionghu.uestc@gmail.com` (Hu Xiong)

setting. However, ID-PKC faces the key escrow problem, i.e., a fully-trusted Private Key Generator (PKG) will generate the private key for the user according to its identity and thus can sign the document and decrypt the ciphertext on behalf of this user.

To avoid the overhead of certificate management in traditional PKC as well as the key escrow problem in ID-PKC altogether, Al-Riyami and Paterson [1] presented a new notion called certificateless public key cryptography (CL-PKC). CL-PKC can be considered as an extension of ID-PKC such that not merely the identity but also the public key will be used in the cryptographic (verification/encryption) algorithm. A Key Generation Center (KGC) is also involved to generate and distribute the private key to the user in the CL-PKC environment. Different from ID-PKC, the full secret key of each user cannot be accessed by the KGC since it is computed by the private key generated by the KGC and the secret value chosen by the user itself. In this way, the inherent key escrow problem in ID-PKC has been successfully solved in the CL-PKC environment.

Featured with confidentiality and non-repudiation, the concept of signcryption originated by Zheng [42] in 1997 has found wide applications where both confidentiality and authentication are required. Compared with the traditional signature-then-encryption approach, signcryption enjoys a lower computational cost and communication overhead. The formal definition and security proof of the signcryption has been given until 2002 by Baek *et al*. [4]. Furthermore, this primitive have been extensively studied in traditional PKC [5, 43, 17, 23, 41, 34, 29] and ID-PKC [28, 25, 11, 13, 14, 7, 32, 20] settings, respectively. As an extension of signcryption in the CL-PKC setting, Barbosa and Farshim [6] initialized the notion of certificateless signcryption (CL-SC) to gain the merits of CL-PKC and signcryption simultaneously. After that, several CL-SC schemes have also been proposed [2, 8, 37, 38]. The security of all these CL-SC schemes has been proved in the random oracle model [9]. In view of the criticism on the random oracle model [12], the signcryption scheme secure in the standard model receives a lot of attention. To remove the random oracles in the security proof of CL-SC, Liu *et al*. [27] proposed the first efficient and provably-secure CL-SC scheme in the standard model by integrating the idea of certificateless signature [26, 39] and certificateless encryption [19, 15]. However, Liu *et al*.'s scheme has been shown to be insecure against the outsider attack [30] and the malicious-but-passive KGC attack [36] respectively. After that, Jin *et al*. proposed an improvement [22] to remedy the weakness in [26]. Unfortunately, we will show that Jin *et al*.'s CL-SC scheme still does not offer neither semantical security against chosen ciphertext attacks nor existential unforgeability against chosen message attacks once the malicious-but-passive KGC is considered. The basic reason about our attack has also been analyzed. It is fair to say devising a CL-SC scheme secure in the standard model remains an open question until now.

In this paper, we strive to close this open problem by investigating CL-SC schemes which can be proven secure in the standard model. By exploiting Bellare and Shoup's one-time signature [10], Hwang *et al*.'s certificateless encryption [19] and Li *et al*.'s identity-based signcryption [24], this paper proposes a new CL-SC scheme in the standard model. It is proven that our CL-SC scheme satisfies semantic security and unforgeability against outside adversary and malicious-but-passive KGC assuming the hardness of bilinear decision Diffie-Hellman (BDDH) and computational Diffie-Hellman (CDH) problems. The proofs do not rely on random oracles.

The rest of this paper is organized as follows. In Section 2, we describe the formal model of CL-SC scheme and the building blocks. In Section 3 we review and analyze Jin *et al.*'s CL-SC scheme. After that, our CL-SC scheme as well as the security analysis have been given in Section 4 and 5, respectively. Finally, the conclusions are given in Section 6.

2. Preliminaries

In this section, we will review the formal definition of CL-SC scheme and the building blocks of our scheme.

2.1. Definitions of CL-SC Schemes

Generally speaking, a CL-SC scheme consists of a tuple (**Setup**, **Partial-Private-Key-Gen**, **User-Key-Gen**, **Private-Key-Gen**, **Signcrypt**, **Unsigncrypt**) described as follows [27, 22].

1. **Setup.** Given a security parameter $k \in \mathbb{N}$ as input, this algorithm is executed by KGC to generate the public system parameter $params$ and a master public/secret key pair (mpk, msk).
2. **Partial-Private-Key-Gen.** Given the master secret key msk along with the user identity $u \in \{0,1\}^*$, this algorithm is executed by KGC to generate a user partial key psk_u, which will be sent to the corresponding user securely.
3. **User-Key-Gen.** Given the public system parameter and user identity u, this algorithm is executed by the user itself to generate a user public/secret key pair (upk_u, usk_u). We stress that the user secret key which will be used in the Sign algorithm cannot be accessed by the KGC to avoid the key escrow problem in ID-PKC.
4. **Private-Key-Gen.** On input $params$, and entity's partial private key psk_u and secret value usk_u, this algorithm generates the entity's full private key sk_u. Note that this algorithm can be omitted since the full private key sk_u used in the **Signcrypt** or **Unsigncrypt** algorithms can be generated by integrating the partial private key psk_u and user secret key usk_u together in the process of performing **Signcrypt** or **Unsigncrypt** algorithms.
5. **Signcrypt.** Given the system parameters $params$, a message m, a sender's user private key sk_S, identity u_S and user public key upk_S, and a receiver's identity u_R and public key upk_R, this algorithm outputs a ciphertext σ or an error symbol $\perp$.
6. **Unsigncrypt.** Given a ciphertext σ, the receiver's user private key sk_S, and the sender's identity u_S and public key upk_S, this algorithm outputs the plaintext m or an error symbol $\perp$.

2.2. Security models

According to [6, 27], the outside attacker who can only compromise the user private key or replaces the user public key and the malicious-but-passive KGC who is responsible for the generation of the public system parameter and master public/secret key pair should be considered in the security model of CL-SC. In this way, two types of security along with two types of adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ has been defined for the CL-SC scheme with the restriction that $\mathcal{A}_1$ cannot compromise the master secret key nor get access to the user partial key and $\mathcal{A}_2$ cannot mount the key replacement attack. The oracles which can be accessed by the adversaries are described as follows.

1. Request-Public-Key Oracle: Given a query on identity $u \in \{0,1\}^*$, this oracle returns the matching user public key upk.
2. Reveal-Partial-Private-Key Oracle: Given a query on identity u, this oracle outputs the partial secret key psk_u associated with this identity.
3. Reveal-Secret-Key Oracle: Given a query on identity u, this oracle outputs a user secret key usk_u associated with this identity.

4. Replace-Public-Key Oracle: Given a identity u and a new user public key upk_u, this oracle replaces the associated user's public key with the new public key upk'_u.
5. Signcrypt Oracle: Upon receiving a sender with identity u_S, a receiver with identity u_R and a message m, challenger C first runs **Signcrypt**$(params, m, sk_S, u_S, upk_S, u_R, upk_R)$, and then returns the resulting ciphertext to the adversary. Here sk_S denotes the sender's full private key. Note that it is possible for the challenger to be unaware of the sender's user secret value when the associated public key has been replaced by adversary. In this case, we require the adversary to provide the sender's user secret value.
6. Unsigncrypt Oracle: Upon receiving a ciphertext σ, a sender with identity u_S and a receiver with identity u_R, challenger C returns the result of **Unsigncrypt**$(\sigma, sk_R, u_S, pk_S)$. Note that it is possible for the challenger to be unaware of the receiver's user secret value when the associated public key has been replaced by adversary. In this case, we require the adversary to provide the receiver's user secret value.

Regarding to the confidentiality, two games, one for $\mathcal{A}_1$ and the other one for $\mathcal{A}_2$, has been defined as follows to capture the attacks launched by $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively.

Game I: In this game, the outside attacker is modeled as Type I adversary $\mathcal{A}_1$ and the game simulator/challenger is modeled as C.

- **Initial.** C first executes **Setup** to generate the master public/secret key pair and public system parameters, and then publishes the public system *params* and keeps the master secret key secret.
- **Phase 1.** In this phase, C runs $\mathcal{A}_1$ on 1^k and public system parameters. During the simulation, $\mathcal{A}_1$ can make queries onto oracles Request-Public-Key, Reveal-Partial-Private-Key, Reveal-Secret-Key, Replace-Public-Key, Signcrypt and Unsigncrypt.
- **Challenge.** Once $\mathcal{A}_1$ decides that **Phase 1** is over, $\mathcal{A}_1$ generates two equal length messages m_0, m_1, two identities u_{S^*} and u_{R^*} on which he wants to be challenged. Challenger C first chooses a bit γ randomly, and then computes σ^*=**Signcrypt**$(params, m_\gamma, sk_{S^*}, u_{S^*}, upk_{S^*}, u_{R^*}, upk_{R^*})$. Finally, C gives σ^* to $\mathcal{A}_1$.
- **Phase 2.** Adversary $\mathcal{A}_1$ continues to issue queries as in **Phase 1**, and C responds in the same way as in **Phase 1**.
- **Guess.** $\mathcal{A}_1$ produces a bit γ' and wins the game if $\gamma' = \gamma$ and the following conditions are satisfied simultaneously.
  1. $\mathcal{A}_1$ cannot extract the private key for any identity if the corresponding public key has been replaced.
  2. $\mathcal{A}_1$ cannot extract the partial private key for u_{R^*} if $\mathcal{A}_1$ has replaced the public key upk_{R^*} before the challenge phase.
  3. In **Phase 2**, $\mathcal{A}_1$ cannot make an unsigncryption query on the challenge ciphertext σ^* under u_{S^*} and u_{R^*} unless the sender's public key upk_{S^*} or the receiver's public key upk_{R^*}, that were used to signcrypt m_γ, has been replaced after the challenge phase.

The advantage of $\mathcal{A}_1$ is defined as $Adv_{\mathcal{A}_1}^{IND-CL-SC-CCA2} = |2Pr[\gamma' = \gamma] - 1|$, where $Pr[\gamma' = \gamma]$ denotes the probability that $\gamma' = \gamma$.

Game II: In this game, the insider attacker (malicious-but-passive KGC) is modeled as Type II adversary $\mathcal{A}_2$ and the game simulator/challenger is modeled as C.

- **Initial.** $\mathcal{A}_2$ executes **Setup** to generate the master public/secret key pair and public system parameters, and send the public system *params* and keeps the master public/secret key pair to challenger C. We should keep in mind that $\mathcal{A}_2$ generates *params* and *msk* by itself.

- **Phase 1.** During the simulation, $\mathcal{A}_2$ can make queries onto oracles Request-Public-Key, Reveal-Secret-Key, Signcrypt and Unsigncrypt. Note that $\mathcal{A}_2$ can compute the partial private key of any identity by itself with the master secret key.

- **Challenge.** Once $\mathcal{A}_2$ decides that **Phase 1** is over, $\mathcal{A}_2$ generates two equal length messages m_0, m_1, two identities u_{S^*} and u_{R^*} on which he wants to be challenged. Challenger C first chooses a bit γ randomly, and then computes σ^*=**Signcrypt**$(params, m_\gamma, sk_{S^*}, u_{S^*}, upk_{S^*}, u_{R^*}, upk_{R^*})$. Finally, C gives σ^* to $\mathcal{A}_2$.

- **Phase 2.** Adversary $\mathcal{A}_2$ continues to issue queries as in **Phase 1**, and C responds in the same way as in **Phase 1**.

- **Guess.** $\mathcal{A}_2$ produces a bit γ' and wins the game if $\gamma' = \gamma$ and the following conditions should be satisfied. In **Phase 2**, $\mathcal{A}_2$ cannot make an unsigncryption query on the challenge ciphertext σ^* under u_{S^*} and u_{R^*} unless the sender's public key upk_{S^*} or the receiver's public key upk_{R^*}, that were used to signcrypt m_γ, has been replaced after the challenge phase.

The advantage of $\mathcal{A}_2$ is defined as $Adv_{\mathcal{A}_2}^{IND-CL-SC-CCA2} = |2Pr[\gamma' = \gamma] - 1|$, where $Pr[\gamma' = \gamma]$ denotes the probability that $\gamma' = \gamma$.

A CL-SC scheme is said to be semantically secure against adaptive chosen ciphertext attacks, if there exists neither polynomial time Type I adversary nor polynomial time Type II adversary who has a non-negligible advantage in game I and game II, respectively.

Regarding to the existential unforgeability, two games, one for $\mathcal{A}_1$ and the other one for $\mathcal{A}_2$, has also been defined as follows to capture the attacks launched by $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively.

Game III: Let C be the game simulator/challenger with the input of security parameter $k \in \mathbb{N}$.

1. **Initial.** C first executes **Setup** to generate the master public/secret key pair and public system parameters, and then publishes the public system *params* and keeps the master secret key secret.
2. **Attack.** In this phase, $\mathcal{A}_1$ adaptively issues a polynomial bounded number of queries as in game I.
3. **Forgery.** Finally, $\mathcal{A}_1$ outputs a new triple $(\sigma^*, u_{S^*}, u_{R^*})$, which is not produced by the Signcrypt query. Adversary $\mathcal{A}_1$ wins this game if the result of Unsigncrypt$(\sigma^*, u_{S^*}, upk_{S^*}, sk_{R^*})$ is not the symbol $\perp$ and the queries are subject to the following constraints:
   (a) $\mathcal{A}_1$ cannot extract the private key for any identity if the corresponding public key has been replaced.
   (b) $\mathcal{A}_1$ cannot extract the partial private key for u_{S^*} if $\mathcal{A}_1$ has replaced the public key upk_{R^*} before the challenge phase.

The advantage of $\mathcal{A}_1$ is defined as $Adv_{\mathcal{A}_1}^{EUF-CL-SC-CMA}$ =Pr[$\mathcal{A}_1$ wins].

Game IV: Let C be the game challenger with the input of security parameter $k \in \mathbb{N}$.

1. **Initial.** $\mathcal{A}_2$ executes **Setup** to generate the master public/secret key pair and public system parameters, and send the public system *params* and keeps the master public/secret key pair to challenger C. We should keep in mind that $\mathcal{A}_2$ generates *params* and *msk* by itself.

2. **Attack.** In this phase, $\mathcal{A}_1$ adaptively issues a polynomial bounded number of queries as in game II.
3. **Forgery.** Finally, $\mathcal{A}_2$ outputs a new triple $(\sigma^*, u_{S^*}, u_{R^*})$, which is not produced by the Signcrypt query. Adversary $\mathcal{A}_1$ wins this game if the result of Unsigncrypt$(\sigma^*, u_{S^*}, upk_{S^*}, sk_{R^*})$ is not the symbol $\perp$.

The advantage of $\mathcal{A}_1$ is defined as $Adv_{\mathcal{A}_2}^{EUF-CL-SC-CMA}$ =Pr[$\mathcal{A}_2$ wins].

A CL-SC scheme is said to be existentially unforgeable under adaptive chosen message attacks, if there exists neither polynomial time Type I adversary nor polynomial time Type II adversary who has a non-negligible success probability in game III and game IV, respectively.

2.3. Bilinear Pairing

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ denote two multiplicative cyclic groups of prime order p . Let $\hat{e}$ be a bilinear map such that $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following properties:

1. Bilinearity: For all $g_1, g_2 \in \mathbb{G}$, and $a, b \in \mathbb{Z}_p$, $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$.
2. Non-degeneracy: $\hat{e}(g_1, g_2) \neq 1_{\mathbb{G}_2}$.
3. Computability: It is efficient to compute $\hat{e}(g_1, g_2)$ for all $g_1, g_2 \in \mathbb{G}$.

Definition 1. *Given two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same prime order p, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ and a generator g of $\mathbb{G}_1$, the bilinear decision Diffie-Hellman (BDDH) problem in $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ is to decide whether $Z = \hat{e}(g, g)^{abc}$ given (g, g^a, g^b, g^c) and an element $Z \in \mathbb{G}_2$. We define the advantage of a distinguisher against the BDDH problem like this*

$$\begin{aligned} Adv(D) &= |P_{a,b,c,\in_R \mathbb{Z}_p, Z \in_R \mathbb{G}_2}[1 \leftarrow D(g^a, g^b, g^c, Z)] \\ &\quad -P_{a,b,c,\in_R \mathbb{Z}_p}[1 \leftarrow D(g^a, g^b, g^c, \hat{e}(g, g)^{abc})]|. \end{aligned}$$

Definition 2. *Given the elements g, g^a and g^b, for some random values $a, b \in \mathbb{Z}_p$ the Computational Diffe-Hellman (CDH) problem consists of computing the element g^{ab}.*

3. Analysis of Jin *et al.*' s scheme

3.1. Overview of Jin et al.' s scheme

Now we review Jin *et al.*'s [22] CL-SC scheme as follows.

Setup. Select a pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ where the order of $\mathbb{G}_1$ is p. Let g be a generator of $\mathbb{G}_1$. Randomly select $\alpha \leftarrow_R \mathbb{Z}_p$, $g_2 \leftarrow_R \mathbb{G}_1$ and then compute $g_1 = g^\alpha$. Also select randomly the following elements: $u', m' \leftarrow_R \mathbb{G}_1$, $u_i \leftarrow_R \mathbb{G}_1$ for $i = 1, \ldots, n_u$, $m_i \leftarrow_R \mathbb{G}_1$ for $i = 1, \ldots, n_m$. Let $\mathbf{U} = (u', u_1, \cdots, u_{n_u})$, $\mathbf{M} = (m', m_1, \cdots, m_{n_m})$. Let $H_1 : \{0,1\}^* \to \{0,1\}^{n_m}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$ be two collision-resistant cryptographic hash functions for some $n_m \in \mathbb{Z}$. The public parameters are $params = \{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_1, g_2, \mathbf{U}, \mathbf{M}, H_1, H_2\}$ and master secret is g_2^α.

Partial-Private-Key-Gen. Let u be a bit string of length n_u representing an identity and let $u[i]$ be the i-th bit of u. Define $\mathcal{U} \subset \{1, \ldots, n_u\}$ to be the set of indices i such that $u[i] = 1$. To construct the partial secret key of identity ID, the KGC randomly pick $r \leftarrow_R \mathbb{Z}_p$ and compute:

$$d_u = (g_2^\alpha(u' \prod_{i \in \mathcal{U}} u_i)^r, g^r).$$

User-Key-Gen. An entity selects a secret value $x_u \leftarrow_R \mathbb{Z}_p$ as his user secret key, the public key and the corresponding signature are $(K, h, pk_u, Y, z) = (u, \hat{e}(g_1, g_2), \hat{e}(g_1, g_2)^{x_u}, \hat{e}(g_1, g_2)^{y_u}, y_u + cx_u \bmod p)$, where $c = H_2(K, Y \parallel params)$.

Private-Key-Gen. The user randomly pick $r' \leftarrow_R \mathbb{Z}_p$ and compute:

$$sk_u = (sk_{u,1}, sk_{u,2}) = (g_2^{\alpha x_u}(u' \prod_{i \in \mathcal{U}} u_i)^{r x_u}(u' \prod_{i \in \mathcal{U}} u_i)^{r'}, g^{r x_u} g^{r'}).$$

Signcrypt. Verify the signature associated with the receiver's public key by checking if the equality $h^z = Y pk_u^c$ holds, where $c = H_2(K, Y \parallel params)$. To send a message $m \in \mathbb{G}_2$, the sender picks $r'' \leftarrow_R \mathbb{Z}_p$ and performs the following steps.

1. Compute $\sigma_1 = m \cdot \hat{e}(g_1, g_2)^{x_R r''}$, $\sigma_2 = g^{r''}$, $\sigma_3 = (u' \prod_{i \in \mathcal{U}_R} u_i)^{r''}$, $\sigma_4 = sk_{S,2}$.
2. Compute $\mathsf{m} = H_1(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_R, pk_R) \in \{0, 1\}^{n_m}$ and assume $\mathsf{m}[i]$ be the i-th bit of m and let $\mathcal{M} \subset \{1, \ldots, n_m\}$ be the set of indices i such that $\mathsf{m}[i] = 1$.
3. Compute $\sigma_5 = sk_{S,1} \cdot (m' \prod_{j \in \mathcal{M}} m_j)^{r''}$.
4. Output the ciphertext $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$.

Unsigncrypt. Given a ciphertext $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ from the user associated with an identity u_S and public key pk_S, a verifier performs the following steps to decrypt the ciphertext:

1. Check the sender's user public key has the right form.
2. Compute $\mathsf{m} = H_1(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_R, pk_R) \in \{0, 1\}^{n_m}$ and assume $\mathsf{m}[i]$ be the i-th bit of m and let $\mathcal{M} \subset \{1, \ldots, n_m\}$ be the set of indices i such that $\mathsf{m}[i] = 1$ and check whether the following equation holds:

$$\hat{e}(\sigma_5, g) = pk_S \cdot \hat{e}(\sigma_4, u' \prod_{i \in \mathcal{U}_S} u_i)\hat{e}(\sigma_2, m' \prod_{j \in \mathcal{M}} m_j).$$

   If the above equation holds, output $m = \sigma_1 \cdot \hat{e}(\sigma_3, sk_{R,2})/\hat{e}(\sigma_2, sk_{R,1})$; otherwise, output $\perp$.

Remark 1. *The algorithm **Private-Key-Gen** can be omitted since the full private key sk_u generated in this algorithm can be created in **Signcrypt** and **Unsigncrypt** algorithms directly.*

3.2. Attack against semantic security

According to [22], their scheme is semantically secure against Type I and Type II adversary in the standard model. However, we will show that their scheme is not semantically secure against chosen-ciphertext attacks by the malicious-but-passive KGC (Type II adversary $\mathcal{A}_2$) in this subsection. The attack is described in detail as follows.

1. In the initial phase, adversary $\mathcal{A}_2$ generates the public parameters *params* and master secret key for challenger C. In particular, adversary $\mathcal{A}_2$ computes $m' \leftarrow_R \mathbb{G}_1$ and $m_i \leftarrow_R \mathbb{G}_1$ for $i = 1, \ldots, n_m$ as follows:
   - Choose random values $\beta', \beta_1, \cdots, \beta_{n_m}$ in Z_p.
   - Compute $m' = g^{\beta'}$ and $m_i = g^{\beta_i}$ for $i = 1, \ldots, n_m$.
2. In phase 1, $\mathcal{A}_2$ needs not issue any query.

3. In the challenge phase, $\mathcal{A}_2$ generates two equal length messages m_0, m_1, two identities u_{S^*} and u_{R^*} on which he wants to be challenged. $\mathcal{A}_2$ has not asked the private key extraction queries on u_{R^*} in **Phase 1**. After that, $\mathcal{A}_2$ sends m_0, m_1 and u_{S^*}, u_{R^*} to C. Then adversary $\mathcal{A}_2$ is given a ciphertext $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*)$ such that

$$\begin{aligned}
\sigma_1^* &= m_\gamma \cdot \hat{e}(g_1, g_2)^{x_{R^*} r''}, \\
\sigma_2^* &= g^{r''}, \sigma_3^* = (u' \prod_{i \in \mathcal{U}_{R^*}} u_i)^{r''}, \sigma_4^* = sk_{S^*,2} \\
\sigma_5^* &= sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}_\gamma} m_j)^{r''},
\end{aligned}$$

where γ denotes the random bit chosen by the challenger C, $\mathcal{U}_{\mathcal{R}^*} \subset \{1, \ldots, n_u\}$ denotes the set of indices i such that $u_{R^*}[i] = 1$, $\mathsf{m}_\gamma = H_1(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, u_{R^*}, pk_{R^*}) \in \{0, 1\}^{n_m}$ and $\mathcal{M}_\gamma \subset \{1, \ldots, n_m\}$ denotes the set of indices i such that $\mathsf{m}_\gamma[i] = 1$. Recall that the goal of $\mathcal{A}_2$ to win this security game is to guess γ correctly.
4. In phase 2, $\mathcal{A}_2$ randomly picks $\hat{r} \leftarrow_R \mathbb{Z}_p$ and generates another ciphertext $\sigma' = (\sigma_1', \sigma_2', \sigma_3', \sigma_4', \sigma_5')$ such that

$$\begin{aligned}
\sigma_1' &= \sigma_1^* \cdot \hat{e}(g_1, g_2)^{x_{R^*} \hat{r}}, \\
\sigma_2' &= \sigma_2^* \cdot g^{\hat{r}}, \sigma_3' = \sigma_3^* \cdot (u' \prod_{i \in \mathcal{U}_{R^*}} u_i)^{\hat{r}}, \sigma_4' = sk_{S^*,2} \\
\sigma_5' &= \frac{\sigma_5^*}{(\sigma_2^*)^{\beta' + \sum_{j \in \mathcal{M}_\gamma} \beta_j}} \cdot (\sigma_2^*)^{\beta' + \sum_{j \in \mathcal{M}_\gamma'} \beta_j} \cdot (m' \prod_{j \in \mathcal{M}_\gamma'} m_j)^{\hat{r}},
\end{aligned}$$

where $\mathsf{m}_\gamma' = H_1(\sigma_1', \sigma_2', \sigma_3', \sigma_4', u_{R^*}, pk_{R^*}) \in \{0, 1\}^{n_m}$ and $\mathcal{M}_\gamma' \subset \{1, \ldots, n_m\}$ denotes the set of indices i such that $\mathsf{m}_\gamma'[i] = 1$.
Observe that $\sigma' = (\sigma_1', \sigma_2', \sigma_3', \sigma_4', \sigma_5')$ is indeed a valid ciphertext under the same message m_γ, the sender u_{S^*} and the receiver u_{R^*} since

$$
\begin{aligned}
\sigma_1' &= \sigma_1^* \cdot \hat{e}(g_1, g_2)^{x_{R^*}\hat{r}} = m_\gamma \cdot \hat{e}(g_1, g_2)^{x_{R^*}(r''+\hat{r})}, \\
\sigma_2' &= \sigma_2^* \cdot g^{\hat{r}} = g^{r''+\hat{r}}, \sigma_3' = \sigma_3^* \cdot (u' \prod_{i \in \mathcal{U}_{R^*}} u_i)^{\hat{r}} = (u' \prod_{i \in \mathcal{U}_{R^*}} u_i)^{r''+\hat{r}}, \sigma_4' = sk_{S^*,2} \\
\sigma_5' &= \frac{\sigma_5^*}{(\sigma_2^*)^{\beta' + \sum_{j \in \mathcal{M}_\gamma} \beta_j}} \cdot (\sigma_2^*)^{\beta' + \sum_{j \in \mathcal{M}'_\gamma} \beta_j} \cdot (m' \prod_{j \in \mathcal{M}'_\gamma} m_j)^{\hat{r}} \\
&= \frac{sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}_\gamma} m_j)^{r''}}{(g^{r''})^{\beta' + \sum_{j \in \mathcal{M}_\gamma} \beta_j}} \cdot (g^{r''})^{\beta' + \sum_{j \in \mathcal{M}'_\gamma} \beta_j} \cdot (m' \prod_{j \in \mathcal{M}'_\gamma} m_j)^{\hat{r}} \\
&= \frac{sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}_\gamma} m_j)^{r''}}{(g^{\beta' + \sum_{j \in \mathcal{M}_\gamma} \beta_j})^{r''}} \cdot (g^{\beta' + \sum_{j \in \mathcal{M}'_\gamma} \beta_j})^{r''} \cdot (m' \prod_{j \in \mathcal{M}'_\gamma} m_j)^{\hat{r}} \\
&= \frac{sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}_\gamma} m_j)^{r''}}{(g^{\beta'} g^{\prod_{j \in \mathcal{M}_\gamma} \beta_j})^{r''}} \cdot (g^{\beta'} g^{\prod_{j \in \mathcal{M}'_\gamma} \beta_j})^{r''} \cdot (m' \prod_{j \in \mathcal{M}'_\gamma} m_j)^{\hat{r}} \\
&= \frac{sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}_\gamma} m_j)^{r''}}{(m' \prod_{j \in \mathcal{M}_\gamma} m_j)^{r''}} \cdot (m' \prod_{j \in \mathcal{M}'_\gamma} m_j)^{r''} \cdot (m' \prod_{j \in \mathcal{M}'_\gamma} m_j)^{\hat{r}} \\
&= sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}'_\gamma} m_j)^{r''+\hat{r}}
\end{aligned}
$$

According to the restrictions specified in the security game, it is legal for $\mathcal{A}_2$ to issue an unsigncryption query to the challenger C by submitting the ciphertext $\sigma' = (\sigma_1', \sigma_2', \sigma_3', \sigma_4', \sigma_5')$ under the sender with identity u_{S^*} and the receiver with identity u_{R^*} since $\sigma^* \neq \sigma'$. Thus, the challenger has to return the underlying message m_γ to $\mathcal{A}_2$. With m_γ, $\mathcal{A}_2$ can certainly obtain the value γ and win the corresponding security game.

3.3. Attack against existential unforgeability

Intuitively, the insecurity of Jin *et al.*'s scheme lies in the fact that, given a ciphertext generated by a sender, a malicious-but-passive KGC (Type II adversary $\mathcal{A}_2$) can derive the sender's full private key, and hence can certainly forge signcryption on behalf of this sender. The attack is described in detail as follows.

1. In the initial phase, adversary $\mathcal{A}_2$ generates the public parameters *params* and master secret key for challenger C. In particular, adversary $\mathcal{A}_2$ computes $m' \leftarrow_R \mathbb{G}_1$ and $m_i \leftarrow_R \mathbb{G}_1$ for $i = 1, \ldots, n_m$ as follows:
   - Choose random values $\beta', \beta_1, \cdots, \beta_{n_m}$ in Z_p.
   - Compute $m' = g^{\beta'}$ and $m_i = g^{\beta_i}$ for $i = 1, \ldots, n_m$.
2. In the attack phase, $\mathcal{A}_2$ issues a signcryption query by submitting a sender with identity u_{S^*} , a receiver with identity u_{R^*} and a message m. Then adversary $\mathcal{A}_2$ is given a ciphertext

$\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ such that

$$\begin{aligned}\sigma_1 &= m \cdot \hat{e}(g_1, g_2)^{x_{R^*} r''},\\ \sigma_2 &= g^{r''}, \sigma_3 = (u' \prod_{i \in \mathcal{U}_{R^*}} u_i)^{r''}, \sigma_4 = sk_{S^*,2}\\ \sigma_5 &= sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}} m_j)^{r''},\end{aligned}$$

where $\mathcal{U}_{\mathcal{R}^*} \subset \{1, \dots, n_u\}$ denotes the set of indices i such that $u_{R^*}[i] = 1$, $\mathsf{m} = H_1(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_{R^*}, pk_{R^*}) \in \{0,1\}^{n_m}$ and $\mathcal{M} \subset \{1, \dots, n_m\}$ denotes the set of indices i such that $\mathsf{m}[i] = 1$. From $\sigma_2 = g^{r''}$ and $\sigma_5 = sk_{S,1} \cdot (m' \prod_{j \in \mathcal{M}} m_j)^{r''}$, adversary $\mathcal{A}_2$ can derive the sender's full private key sk_{S^*} by computing $\frac{\sigma_5}{\sigma_2^{\beta' + \sum_{j \in \mathcal{M}} \beta_j}}$, since

$\frac{\sigma_5}{\sigma_2^{\beta' + \sum_{j \in \mathcal{M}} \beta_j}} = \frac{sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}} m_j)^{r''}}{(g^{r''})^{\beta' + \sum_{j \in \mathcal{M}} \beta_j}} = \frac{sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}} m_j)^{r''}}{(g^{\beta' + \sum_{j \in \mathcal{M}} \beta_j})^{r''}} = \frac{sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}} m_j)^{r''}}{(g^{\beta'} \prod_{j \in \mathcal{M}} g^{\beta_j})^{r''}} = \frac{sk_{S^*,1} \cdot (m' \prod_{j \in \mathcal{M}} m_j)^{r''}}{(m' \prod_{j \in \mathcal{M}} m_j)^{r''}} = sk_{S^*,1}$

Recall that $\sigma_4 = sk_{S^*,2}$. Thus, adversary $\mathcal{A}_2$ can obtain the sender's full private key $sk_{S^*} = (sk_{S^*,1}, sk_{S^*,2})$. Equipped with sender's full private key, $\mathcal{A}_2$ can definitely forge signcryption on behalf of this sender and win can always win the corresponding security game.

Our result shows that Jin *et al.*'s scheme can not offer semantic security and existential unforgeability against in the standard model. The basic reason of our attack is that the part of the ciphertext including the messages to be signed is irrelevant to the sender's user secret key. More specifically, the KGC can add or remove $(m' \prod_{j \in \mathcal{M}} m_j)^r$ without affecting the validity of the ciphertext, where r is the blind factor in the **Signcrypt** algorithm.

4. Construction of our scheme

We construct a new CL-SC scheme against Type I and Type II adversaries in the standard model by incorporating the idea of Bellare and Shoup's one-time signature [10], Hwang *et al.*'s certificateless encryption [19] and Li *et al.*'s identity-based signcryption [24]. To fight against the malicious KGC attack, our scheme use a different user public/secret key generation algorithm and embed the sender's public key in the ciphertext. In this way, the unforgeability of our scheme will be guaranteed since the secret key of the sender cannot be extracted by the malicious KGC.

Setup. Select a pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ where the order of $\mathbb{G}_1$ is p. Let g be a generator of $\mathbb{G}_1$. Randomly select $\alpha \leftarrow_R \mathbb{Z}_p$, $g_2 \leftarrow_R \mathbb{G}_1$, $K \in_R \{0,1\}^k$, and then compute $g_1 = g^\alpha$, $h = \hat{e}(g_1, g_2)$. Also select randomly the following elements: $u', m' \leftarrow_R \mathbb{G}_1$, $u_i \leftarrow_R \mathbb{G}_1$ for $i = 1, \dots, n_u$, $m_i \leftarrow_R \mathbb{G}_1$ for $i = 1, \dots, n_m$. Let $\mathbf{U} = (u_i)$, $\mathbf{M} = (m_i)$. Let $H_1 : \{0,1\}^* \to \mathbb{Z}_p$, $H_2 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_{\mathrm{m}} : \{0,1\}^* \to \{0,1\}^{n_m}$ be three collision-resistant cryptographic hash functions for some $n_m \in \mathbb{Z}$. The public parameters are $params = \{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_1, g_2, h, u', \mathbf{U}, m', \mathbf{M}, K, H_1, H_2, H_{\mathrm{m}}\}$ and master secret is g_2^α.

Partial-Private-Key-Gen. Let u be a bit string of length n_u representing an identity and let $u[i]$ be the i-th bit of u. Define $\mathcal{U} \subset \{1, \dots, n_u\}$ to be the set of indices i such that $u[i] = 1$. To construct the partial secret key of identity ID, the KGC randomly pick $r_u \leftarrow_R \mathbb{Z}_p$ and compute:

$$(g_2^\alpha (u' \prod_{i \in \mathcal{U}} u_i)^{r_u}, g^{r_u}) = (psk_{u,1}, psk_{u,2}).$$

Therefore, the sender and the receiver's partial private keys are

$$(g_2^{\alpha}(u'\prod_{i\in\mathcal{U}_S} u_i)^{r_S}, g^{r_S}) = (psk_{S,1}, psk_{S,2}).$$

and

$$(g_2^{\alpha}(u'\prod_{i\in\mathcal{U}_R} u_i)^{r_R}, g^{r_R}) = (psk_{R,1}, psk_{R,2}).$$

User-Key-Gen. An entity selects two secret value $x_u, y_u \leftarrow_R \mathbb{Z}_p$ as his user secret key usk_u such that $usk_u = (x_u, y_u)$, and computes the corresponding user public key as $upk_u = (upk_{u,1}, upk_{u,2}, upk_{u,3}) = (h^{x_u}, g_2^{y_u}, g^{y_u})$. After that, the corresponding signature associated with the public key are computed as $z_u = y_u + c_{1,u}x_u$ where $c_{1,u} = H_1(K, upk_u \parallel params)$. According to [19], this one-time signature can be generated applying the technique of Fiat-Shamir transform without random oracles as described in [10].

Signcrypt. To send a message $m \in \mathbb{G}_2$ to the receiver associated with identity u_R and user public key upk_R, the sender associated with identity u_S, user public key upk_S, partial private key $(psk_{S,1}, psk_{S,2})$ and user secret key usk_S first checks the receiver's user public key has the right form such that $\hat{e}(g_1, g_2)^{z_R} = upk_{R,1}^{c_{1,R}} \cdot \hat{e}(g_1, upk_{R,2})$ and $\hat{e}(upk_{R,2}, g) = \hat{e}(upk_{R,3}, g_2)$ where $c_{1,R} = H_1(K, upk_R \parallel params)$. After that, the sender picks $k \leftarrow_R \mathbb{Z}_p$ and performs the following steps.

1. Compute $\sigma_1 = m \cdot upk_{R,1}^k$, $\sigma_2 = g^k$, $\sigma_3 = (u'\prod_{i\in\mathcal{U}_R} u_i)^k$, $\sigma_4 = psk_{S,2}^{x_S}$.
2. Compute $\mathsf{m} = H_{\mathrm{m}}(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R) \in \{0,1\}^{n_m}$ and assume $\mathsf{m}[i]$ be the i-th bit of m and let $\mathcal{M} \subset \{1, \ldots, n_m\}$ be the set of indices i such that $\mathsf{m}[i] = 1$.
3. Compute $\sigma_5 = psk_{S,1}^{x_S} \cdot upk_{S,3}^{kc_2} \cdot (m'\prod_{j\in\mathcal{M}} m_j)^k$, where $c_2 = H_2(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R)$.
4. Output the ciphertext $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$.

Unsigncrypt. Given a ciphertext $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ from the user associated with an identity u_S and public key upk_S, a verifier associated with partial private key $(psk_{R,1}, psk_{R,2})$ and user secret key usk_R performs the following steps to decrypt the ciphertext:

1. Check the sender's user public key has the right form such that $\hat{e}(g_1, g_2)^{z_S} = upk_{S,1}^{c_{1,S}} \cdot \hat{e}(g_1, upk_{S,2})$ and $\hat{e}(upk_{S,2}, g) = \hat{e}(upk_{S,3}, g_2)$ where $c_{1,S} = H_1(K, upk_S \parallel params)$.
2. Compute $\mathsf{m} = H_{\mathrm{m}}(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R) \in \{0,1\}^{n_m}$ and assume $\mathsf{m}[i]$ be the i-th bit of m and let $\mathcal{M} \subset \{1, \ldots, n_m\}$ be the set of indices i such that $\mathsf{m}[i] = 1$ and check whether the following equation holds:

$$\hat{e}(\sigma_5, g) = upk_{S,1} \cdot \hat{e}(\sigma_4, u'\prod_{i\in\mathcal{U}_S} u_i)\hat{e}(\sigma_2, upk_{S,3}^{c_2} \cdot (m'\prod_{j\in\mathcal{M}} m_j))$$

where $c_2 = H_2(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R)$. If the above equation holds, output $m = \sigma_1 \cdot \hat{e}(\sigma_3, psk_{R,2}^{x_R})/\hat{e}(\sigma_2, psk_{R,1}^{x_R})$; otherwise, output $\perp$.

5. Analysis of our scheme

Theorem 1. *Our CL-SC scheme is existentially unforgeable against chosen message attacks (EUF-CL-SC-CMA) in the standard model assuming the CDH problem is hard.*

This theorem follows Lemmas 1 and 2.

Lemma 1. *(Type I Existential Unforgeability). Our CL-SC scheme is (ϵ, t)-existential unforgeable against Type I adversary with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-CDH assumption holds in $\mathbb{G}_1$, where $\epsilon' \geq \frac{\epsilon}{16(q_{pp}+q_s+q_u)q_u(n_u+1)(n_m+1)}$ and $t' = t + O((q_{pp}n_u + q_s(n_u + n_m))\rho + (q_k + q_{pp} + q_s)\tau)$ where q_{pp} is the number of queries made to the* Reveal-Partial-Private-Key *oracle, q_s is the number of queries made to the* Signcrypt *oracle, q_u is the number of queries made to the* Unsigncrypt *oracle, q_k is the number of queries made to the* Reveal-Secret-Key *and* Request-Public-Key *oracles altogether, and ρ and τ are the time for a multiplication and an exponentiation in $\mathbb{G}_1$ respectively.*

Proof. Let C be a CDH attacker who receives a random instance (g, g^a, g^b) of the CDH problem in $\mathbb{G}_1$ and has to compute the value of abP, where g is a generator of $\mathbb{G}_1$ and a, b are chosen randomly from $\mathbb{Z}_p^*$. $\mathcal{A}_1$ is a type I adversary who interacts with C. We show how C can use $\mathcal{A}_1$ as a subroutine to solve the CDH problem, i.e. to compute abP.

Initial. C sets $l_\mathrm{u} = 2(q_e + q_s)$ and $l_m = 2q_s$, and randomly chooses two integers k_u and k_m, with $0 \leq k_u \leq n_u$ and $0 \leq k_m \leq n_m$. We will assume that $l_\mathrm{u}(n_u + 1) < p$ and $l_m(n_m + 1) < p$ for the given values of q_e, q_s, q_k, n_u and n_m. The simulator then chooses an integer $x' \leftarrow_R \mathbb{Z}_{l_\mathrm{u}}$ and a vector (x_i) of length n_u, with $x_i \leftarrow_R \mathbb{Z}_{l_\mathrm{u}}$ for all i. Likewise, it chooses another integer $z' \leftarrow_R \mathbb{Z}_{l_m}$ and a vector (z_j) of length n_m, with $z_j \leftarrow_R \mathbb{Z}_{l_m}$ for all j. Lastly, C chooses two integer $y', w' \leftarrow_R \mathbb{Z}_p$ and two vectors, (y_i) and (w_j), of length n_u and n_m, respectively, with $y_i, w_j \leftarrow_R \mathbb{Z}_p$ for all i and j. Two pairs of functions are defined for an identity u and a message m respectively:

$$F(u) = x' + \sum_{i \in \mathcal{U}} x_i - l_\mathrm{u}k_u \quad J(u) = y' + \sum_{i \in \mathcal{U}} y_i \quad K(m) = z' + \sum_{j \in \mathcal{M}} z_j - l_m k_m \quad L(m) = w' + \sum_{j \in \mathcal{M}} w_j,$$

The challenger C assigns $g_1 = g^\alpha$, $g_2 = g^b$, $u' = g_2^{-l_\mathrm{u}k_u + x'} g^{y'}$, $u_i = g_2^{x_i} g^{y_i}$ $(1 \leq i \leq n_u)$, $m' = g_2^{-l_m k_m + z'} g^{w'}$, $m_j = g_2^{z_j} g^{w_j}$ $(1 \leq j \leq n_m)$, picks $K \in_R \{0, 1\}^k$, and sends the system parameters $params = (g_1, g_2, u', (u_i), m', (m_j), K)$ to $\mathcal{A}_1$. Moreover, this assignment of parameter means that the master secret will be $g_2^\alpha = g_2^a = g^{ab}$ and we have the following equations:

$$u' \prod_{i \in \mathcal{U}} u_i = g_2^{F(u)} g^{J(u)} \quad \text{and} \quad m' \prod_{i \in \mathcal{M}} m_i = g_2^{K(m)} g^{L(m)}$$

Attack. C maintains a list $\mathcal{L} = \{u, psk_u, usk_u, upk_u, z_u\}$ which is initially empty and simulates all oracles as follows:

Request-Public-Key Oracle: On receiving an identity u of length n_u, C looks up the list $\mathcal{L}$ to find out the corresponding entry. If it does not exist, C runs **Partial-Private-Key-Gen** and **User-Key-Gen** algorithms to generate the partial secret key psk_u, user public/secret key (upk_u, usk_u) along with the corresponding one time signature z_u, respectively. Here, $usk_u = (x_u, y_u)$ and $upk_u = (upk_{u,1}, upk_{u,2}, upk_{u,3}) = (\hat{e}(g_2, g^a)^{\alpha x_u}, g_2^{y_u}, g^{y_u})$. z_u can be simulated in the same manner as in the signing oracle of the one-time signature. It then stores $\{u, psk_u, usk_u, upk_u, z_u\}$ into list $\mathcal{L}$. In both cases, upk_u is returned.

Reveal-Partial-Private-Key Oracle: On receiving a query for the partial private key of an identity u, C can construct a partial private key by choosing $r_u \leftarrow_R \mathbb{Z}_p$ randomly and computing $(psk_{u,1}, psk_{u,2}) = (g_1^{-\frac{J(u)}{F(u)}}(u' \prod_{i \in \mathcal{U}} u_i)^{r_u}, g_1^{-\frac{1}{F(u)}} g^{r_u})$ without knowing the master secret. Here, $F(u) \neq 0 \bmod p$. It is obvious that a partial private key $(psk_{u,1}, psk_{u,2})$ associated with identity u defining in this manner is valid in case $\tilde{r}_u = r_u - a/F(u)$, since that

$$\begin{aligned} psk_{u,1} &= g_1^{-\frac{J(u)}{F(u)}}(u'\prod_{i\in\mathcal{U}} u_i)^{r_u} = g_2^a(g_2^{F(u)}g^{J(u)})^{-a/F(u)}(g_2^{F(u)}g^{J(u)})^{r_u} = g_2^a(g_2^{F(u)}g^{J(u)})^{r_u-a/F(u)} \\ &= g_2^a(u'\prod_{i\in\mathcal{U}} u_i)^{\tilde{r}_u} \end{aligned}$$

and $psk_{u,2} = g_1^{-\frac{1}{F(u)}}g^{r_u} = g^{r_u-a/F(u)} = g^{\tilde{r}_u}$. From the point of view of $\mathcal{A}_1$, all the partial private keys computed by C will be indistinguishable from the keys generated by a true challenger. However, C will abort the above simulation provided $F(u) = 0 \bmod p$. Assuming $l_\mathrm{u}(n_u + 1) < p$ which indicates $0 \leq l_\mathrm{u}k_u < p$ and $0 \leq x' + \sum_{i\in\mathcal{U}} x_i < p$, it is trivial to know $F(u) = 0 \bmod p$ which infers that $F(u) = 0 \bmod l_\mathrm{u}$. In this case, $F(u) \neq 0 \bmod l_\mathrm{u}$ implies $F(u) \neq 0 \bmod p$.

Reveal-Secret-Key Oracle: On receiving a query for a public key of an identity u, C looks up the list $\mathcal{L}$ to find out the corresponding entry. If it does not exist, C runs **User-Key-Gen** algorithm to generate the user public/secret key pair (upk_u, usk_u) and the corresponding one time signature z_u associated with the user public key upk_u. It stores the key pair along with the one time signature in list $\mathcal{L}$ and returns the secret key usk_u.

Replace-Public-Key Oracle: On receiving a public key replacement request on an identity u with a new public/secret key pair (upk'_u, usk'_u) and one time signature z'_u, C locates the entry $\{u, psk_u, usk_u, upk_u, z_u\}$ in list $\mathcal{L}$ and updates this entry as $\{u, psk_u, usk'_u, upk'_u, z'_u\}$. If it does not exist, C creates a new entry for this identity by invoking the algorithm **User-Key-Gen**.

Signcrypt Oracle: On receiving a query for a ciphertext on a sender with identity u_S, a receiver with identity u_R and a message m, C first finds the items $\{u_R, psk_R, usk_R, upk_R, z_R\}$ and $\{u_S, psk_S, usk_S, upk_S, z_S\}$ in list $\mathcal{L}$. After that, C checks the validity of the signature z_R of the public key upk_R and whether the user public key upk_S and the one time signature z_S have been replaced or not. If z_R is invalid or (upk_S, z_S) have been replaced, the challenger C aborts the simulation. Otherwise, C constructs a partial-secret key as in the Reveal-Partial-Private-Key oracle in case $F(u_S) \neq 0 \bmod l_\mathrm{u}$. C then checks from $\mathcal{L}$ whether the user secret key usk_S has been created or not. If it is not been created, runs the Reveal-Secret-Key oracle and stores the secret/public key pair in $\mathcal{L}$. If it has been created, it just invokes the **Signcrypt** algorithm to create a ciphertext on u and m.

Unsigncrypt Oracle: On receiving a given query of an unsigncryption on a ciphertext $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$, a sender associated with an identity u_S, and a receiver associated with an identity u_R, a public key $upk_R = (upk_{R,1}, upk_{R,2})$ along with the one time signature z_R, C first checks the validity of the signature z_R. If the verification is valid, C performs the unsigncryption as follows.

- If the public key upk_R has not been replaced, C accesses the list $\mathcal{L}$ to find (psk_R, usk_R), performs the **Unsigncrypt** algorithm to recover the message m, and sends it to $\mathcal{A}_1$. If the corresponding item does not exist, C runs the **Partial-Private-Key-Gen** and **User-Key-Gen** algorithms to generate the partial private key psk_R and the user secret key usk_R (assuming $F(u_R^*) \neq 0 \bmod l_\mathrm{u}$), then performs the **Unsigncrypt** algorithm to recover the message m.

- If the public key upk_R has already been replaced or $F(u_R^*) = 0 \bmod l_\mathrm{u}$, C will access the list $\mathcal{L}$ to obtain sender u_S's partial private key psk_S and user secret key $usk_S = (x_S, y_S)$ such that $upk_S = (upk_{S,1}, upk_{S,2}, upk_{S,3}) = (h^{x_S}, g_2^{y_S}, g^{y_S})$ and retrieve receiver u_R's user secret key $usk_R = (x_R, y_R)$ such that $upk_R = (upk_{R,1}, upk_{R,2}, upk_{R,3}) = (h^{x_R}, g_2^{y_R}, g^{y_R})$ (or the user secret key usk_R associated with the current public upk_R can be provided by the adversary in case the public key has been replaced). With $\sigma_5 = psk_{S,1}^{x_S} \cdot upk_{S,3}^{kc_2} \cdot (g_2^{K(m)} g^{L(m)})^k$ and $\sigma_2 = g^k$ for some $k \leftarrow_R \mathbb{Z}_p$ and $c_2 = H_2(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R)$, C can extract $g_2^k = (\sigma_5/(psk_{S,1}^{x_S} \cdot \sigma_2^{dy_S} \cdot \sigma_2^{L(m)}))^{1/K(m)}$ and computes $m = \sigma_1/\hat{e}(g_1, g_2^k)^{x_R}$.

Forgery. If C does not abort as a consequence of one of the queries above, $\mathcal{A}_1$ will, with probability at least ϵ, return the sender's identity u_{S^*}, the receiver's identity u_{R^*}, a message m^*, and valid forgery $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*)$ on m^*. If $F(u_{S^*}) \neq 0 \bmod p$ or $K(m^*) \neq 0 \bmod p$, then C will abort. If, on the other hand, $F(u_{S^*}) = 0 \bmod p$ and $K(m^*) = 0 \bmod p$, C retrieves u_{S^*}'s user secret key $usk_{S^*} = (x_{S^*}, y_{S^*})$ and computes

$$\frac{\sigma_5^*}{\sigma_4^{*J(u_{S^*})} \sigma_2^{*L(m^*)} \sigma_2^{*c_2^* y_{S^*}}} = \frac{g_2^{ax_{S^*}} (u' \prod_{i \in \mathcal{U}_{S^*}} u_i)^{k_u} (m' \prod_{k \in \mathcal{M}} m_k)^{k_m} g^{y_{S^*} k_m c_2^*}}{g^{J(u_{S^*}) k_u} g^{L(m^*) k_m} g^{k_m y_{S^*} c_2^*}} = g^{abx_{S^*}},$$

where $c_2^* = H_2(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, u_{S^*}, u_{R^*}, upk_{S^*}, upk_{R^*})$. Thus, C can outputs g^{ab} as the solution to the CDH problem instance.

Probability Analysis. For the simulation to complete without aborting, we require the following conditions fulfilled:

1. All Reveal-Partial-Private-Key queries on an identity u have $F(u) \neq 0 \bmod l_{\mathrm{u}}$.
2. All Signcrypt queries of an sender u_S have $F(u_S) \neq 0 \bmod l_{\mathrm{u}}$.
3. All Unsigncrypt queries (u_S, u_S, σ) will either have $F(u_R) \neq 0 \bmod l_{\mathrm{u}}$ or $K(\mathsf{m}) \neq 0 \bmod l_m$ where $\mathsf{m} = H_{\mathrm{m}}(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R) \in \{0,1\}^{n_m}$.
4. $F(u_R^*) = 0 \bmod l_{\mathrm{u}}$ and $K(m^*) = 0 \bmod l_m$.

In order to make the analysis more simple, we will bound the probability of a subcase of this event. Let $u_1, \ldots, u_{q_I}$ be the identities appearing in these queries not involving the challenge identity and let $m_1, \ldots, m_{q_M}$ be the messages in the unsigncryption queries involving the challenge identity u^*. Clearly, we will have $q_I \leq q_{pp} + q_s + q_u$ and $q_M \leq q_u$. Define the events A_i, A^*, B_j, B^* as

$$A_i : F(u_i) \neq 0 \bmod l_{\mathrm{u}}, \text{where} \quad i = 1, \ldots, q_I, \quad A^* : F(u_R^*) = 0 \bmod p$$

$$B_j : K(m_j) \neq 0 \bmod l_m, \text{where} \quad j = 1, \ldots, q_M, \quad B^* : K(m^*) = 0 \bmod p$$

The probability of C not aborting is : Pr[not abort]$\geq Pr[(\bigwedge_{i=1}^{q_I} A_i \wedge A^*) \wedge (\bigwedge_{j=1}^{q_M} B_j \wedge B^*)]$. It is obvious to observe that the events $(\bigwedge_{i=1}^{q_I} A_i \wedge A^*)$ and $(\bigwedge_{j=1}^{q_M} B_j \wedge B^*)$ are independent.

The assumption $l_{\mathrm{u}}(n_u + 1) < p$ implies if $F(u) = 0 \bmod p$ then $F(u) = 0 \bmod l_{\mathrm{u}}$. In addition, it also implies that if $F(u) = 0 \bmod l_{\mathrm{u}}$, there will be a unique choice of k_u with $0 \leq k_u \leq n_u$ such that $F(u) = 0 \bmod p$. Since k_u, x' and vector (x_i) of length n_u are randomly chosen, we have

$$\begin{aligned} Pr[A^*] &= Pr[F(u^*) = 0 \bmod p \wedge F(u^*) = 0 \bmod l_{\mathrm{u}}] \\ &= Pr[F(u^*) = 0 \bmod l_{\mathrm{u}}] Pr[F(u^*) = 0 \bmod p | F(u^*) = 0 \bmod l_{\mathrm{u}}] = \frac{1}{l_{\mathrm{u}}} \frac{1}{n_u + 1} \end{aligned}$$

On the other hand, we have $Pr[\bigwedge_{i=1}^{q_I} A_i|A^*] = 1 - Pr[\bigvee_{i=1}^{q_I} \overline{A}_i|A^*] \geq 1 - \sum_{i=1}^{q_I} Pr[\overline{A}_i|A^*]$ where $\overline{A}_i$ denote the event $F(u_i) = 0 \bmod l_u$.

If F is evaluated on two different identities, u_{i1} and u_{i2}, then the sums appearing in $F(u_{i1})$ and $F(u_{i2})$ will differ in at least one randomly chosen value, and the events $F(u_{i1}) = 0 \bmod l_u$ and $F(u_{i2}) = 0 \bmod l_u$ will be independent. Also since the events A_i and A^* are independent for any i, we have $Pr[\overline{A}_i|A^*] = 1/l_u$. Hence , we have

$$Pr[\bigwedge_{i=1}^{q_I} A_i \wedge A^*] = Pr[A^*]Pr[\bigwedge_{i=1}^{q_I} A_i|A^*] \geq \frac{1}{l_u(n_u+1)}(1 - \frac{q_{pp}+q_s+q_u}{l_u})$$

and setting $l_u = 2(q_{pp} + q_s + q_u)$ as in the simulation gives $Pr[\bigwedge_{i=1}^{q_I} A_i \wedge A^*] \geq \frac{1}{4(q_{pp}+q_s+q_u)(n_u+1)}$

A similar analysis for the sign queries gives the result $Pr[\overline{B}_j|B^*] \geq \frac{1}{4q_u(n_m+1)}$ and we get that Pr[not abort]$\geq Pr[\overline{A}_i|A^*]Pr[\overline{B}_j|B^*] \geq \frac{1}{16(q_{pp}+q_s+q_u)q_u(n_u+1)(n_m+1)}$

If the simulation does not abort, $\mathcal{A}_1$ will produce a forged signature with probability at least ϵ. Thus C can solve for the CDH problem instance with probability $\epsilon' \geq \frac{\epsilon}{16(q_{pp}+q_s+q_u)q_u(n_u+1)(n_m+1)}$

Lemma 2. *(Type II Existential Unforgeability). Our CL-SC scheme is (ϵ, t)-existential unforgeable against Type II adversary with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-CDH assumption holds in $\mathbb{G}_1$, where $\epsilon' \geq \frac{\epsilon}{16(q_s+q_u)q_u(n_u+1)(n_m+1)}$ and $t' = t + O((q_{pp}n_u + q_s(n_u+n_m))\rho + (q_k+q_s)\tau)$.*

Proof. Let C be a CDH attacker who receives a random instance (g, g^a, g^b) of the CDH problem in $\mathbb{G}_1$ and has to compute the value of abP, where g is a generator of $\mathbb{G}_1$ and a, b are chosen randomly from $\mathbb{Z}_p^*$. $\mathcal{A}_2$ is a type II adversary who interacts with C. We show how C can use $\mathcal{A}_2$ as a subroutine to solve the CDH problem, i.e. to compute abP.

Initial. C sets $l_u = 2(q_e + q_s)$ and $l_m = 2q_s$, and randomly chooses two integers k_u and k_m, with $0 \leq k_u \leq n_u$ and $0 \leq k_m \leq n_m$. We will assume that $l_u(n_u + 1) < p$ and $l_m(n_m + 1) < p$ for the given values of q_e, q_s, q_k, n_u and n_m. The simulator then chooses an integer $x' \leftarrow_R \mathbb{Z}_{l_u}$ and a vector (x_i) of length n_u, with $x_i \leftarrow_R \mathbb{Z}_{l_u}$ for all i. Likewise, it chooses another integer $z' \leftarrow_R \mathbb{Z}_{l_m}$ and a vector (z_j) of length n_m, with $z_j \leftarrow_R \mathbb{Z}_{l_m}$ for all j. Lastly, C chooses two integer $y', w' \leftarrow_R \mathbb{Z}_p$ and two vectors, (y_i) and (w_j), of length n_u and n_m, respectively, with $y_i, w_j \leftarrow_R \mathbb{Z}_p$ for all i and j. Two pairs of functions are defined for an identity u and a message m respectively:

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

The challenger C selects $\alpha \in_R \mathbb{Z}_p^*$ and assigns $g_1 = g^\alpha$, $g_2 = g^b$, $u' = g_2^{-l_u k_u + x'} g^{y'}$, $u_i = g_2^{x_i} g^{y_i}$ $(1 \leq i \leq n_u)$, $m' = g_2^{-l_m k_m + z'} g^{w'}$, $m_j = g_2^{z_j} g^{w_j}$ $(1 \leq j \leq n_m)$, picks $K \in_R \{0, 1\}^k$, and sends the system parameters $params = (g_1, g_2, u', (u_i), m', (m_j), K)$ as well as the master secret $g_2^\alpha = (g^b)^\alpha$ to $\mathcal{A}_2$. Moreover, this assignment of parameter means that:

$$U = u' \prod_{i\in\mathcal{U}} u_i = g_2^{F(u)} g^{J(u)} \quad \text{and} \quad m' \prod_{i\in\mathcal{M}} m_i = g_2^{K(m)} g^{L(m)}$$

Attack. C maintains a list $\mathcal{L} = \{u, psk_u, usk_u, upk_u, z_u\}$ which is initially empty and simulates all oracles as follows:

Request-Public-Key Oracle: On receiving an identity u of length n_u, C looks up the list $\mathcal{L}$ to find out the corresponding entry. If it does not exist, C runs **Partial-Private-Key-Gen** and **User-Key-Gen** algorithms to generate the partial secret key psk_u, user public/secret key (upk_u, usk_u)

along with the corresponding one time signature z_u, respectively. Here, $usk_u = (x_u, y_u)$ and $upk_u = (upk_{u,1}, upk_{u,2}, upk_{u,3}) = (\hat{e}(g_2, g^a)^{\alpha x_u}, g_2^{y_u}, g^{y_u})$. z_u can be simulated in the same manner as in the signing oracle of the one-time signature. It then stores $\{u, psk_u, usk_u, upk_u, z_u\}$ into list $\mathcal{L}$. In both cases, upk_u is returned.

Reveal-Partial-Private-Key Oracle: On receiving a query for the partial private key of an identity u, C can construct a partial private key by choosing $r_u \leftarrow_R \mathbb{Z}_p$ randomly and computing $(psk_{u,1}, psk_{u,2}) = ((g^{a\alpha})^{-\frac{J(u)}{F(u)}}(u' \prod_{i \in \mathcal{U}} u_i)^{r_u}, (g^{a\alpha})^{-\frac{1}{F(u)}} g^{r_u}) = (g_2^{a\alpha}(u' \prod_{i \in \mathcal{U}} u_i)^{\tilde{r}_u}, g^{\tilde{r}_u})$. Here, $F(u) \neq 0 \bmod p$. It is obvious that a partial private key $(psk_{u,1}, psk_{u,2})$ associated with identity u defining in this manner is valid in case $\tilde{r}_u = r_u - a\alpha/F(u)$. From the point of view of $\mathcal{A}_2$, all the partial private keys computed by C will be indistinguishable from the real one. However, C will abort the above simulation provided $F(u) = 0 \bmod p$. Assuming $l_{\mathrm{u}}(n_u + 1) < p$ which indicates $0 \leq l_{\mathrm{u}} k_{\mathcal{U}} < p$ and $0 \leq x' + \sum_{i \in \mathcal{U}} x_i < p$, it is trivial to know $F(u) = 0 \bmod p$ which infers that $F(u) = 0 \bmod l_{\mathrm{u}}$. In this case, $F(u) \neq 0 \bmod l_{\mathrm{u}}$ implies $F(u) \neq 0 \bmod p$.

Reveal-Secret-Key Oracle: On receiving a query for a public key of an identity u, C looks up the list $\mathcal{L}$ to find out the corresponding entry. If it does not exist, C runs **User-Key-Gen** algorithm to generate the user public/secret key pair (upk_u, usk_u) and the corresponding one time signature z_u associated with the user public key upk_u. It stores the key pair along with the one time signature in list $\mathcal{L}$ and returns the secret key usk_u.

Signcrypt Oracle: On receiving a query for a ciphertext on a sender with identity u_S, a receiver with identity u_R and a message m, C first finds the items $\{u_R, psk_R, usk_R, upk_R, z_R\}$ and $\{u_S, psk_S, usk_S, upk_S, z_S\}$ in list $\mathcal{L}$. After that, C checks the validity of the signature z_R of the public key upk_R and whether the user public key upk_S and the one time signature z_S have been replaced or not. If z_R is invalid or (upk_S, z_S) have been replaced, the challenger C aborts the simulation. Otherwise, C constructs a partial-secret key as in the Reveal-Partial-Private-Key oracle in case $F(u_S) \neq 0 \bmod l_{\mathrm{u}}$. C then checks from $\mathcal{L}$ whether the user secret key usk_S has been created or not. If it is not been created, runs the Reveal-Secret-Key oracle and stores the secret/public key pair in $\mathcal{L}$. If it has been created, it just invokes the **Signcrypt** algorithm to create a ciphertext on u and m.

Unsigncrypt Oracle: On receiving a given query of an unsigncryption on a ciphertext $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$, a sender associated with an identity u_S, and a receiver associated with an identity u_R, a public key $upk_R = (upk_{R,1}, upk_{R,2})$ along with the one time signature z_R, C first checks the validity of the signature z_R. If the verification is valid, C performs the unsigncryption as follows.

- If the public key upk_R has not been replaced, C accesses the list $\mathcal{L}$ to find (psk_R, usk_R), performs the **Unsigncrypt** algorithm to recover the message m, and sends it to $\mathcal{A}_1$. If the corresponding item does not exist, C runs the **Partial-Private-Key-Gen** and **User-Key-Gen** algorithms to generate the partial private key psk_R and the user secret key usk_R (assuming $F(u_R^*) \neq 0 \bmod l_{\mathrm{u}}$), then performs the **Unsigncrypt** algorithm to recover the message m.

- If the public key upk_R has already been replaced or $F(u_R^*) = 0 \bmod l_{\mathrm{u}}$, C will access the list $\mathcal{L}$ to obtain sender u_S's partial private key psk_S and user secret key $usk_S = (x_S, y_S)$ such that $upk_S = (upk_{S,1}, upk_{S,2}, upk_{S,3}) = (h^{x_S}, g_2^{y_S}, g^{y_S})$ and retrieve receiver u_R's user secret key $usk_R = (x_R, y_R)$ such that $upk_R = (upk_{R,1}, upk_{R,2}, upk_{R,3}) = (h^{x_R}, g_2^{y_R}, g^{y_R})$ (or the user secret key usk_R associated with the current public upk_R can be provided by the adversary in case the public key has been replaced). With $\sigma_5 = psk_{S,1}^{x_S} \cdot upk_{S,3}^{k} \cdot (g_2^{K(m)} g^{L(m)})^k$ and

$\sigma_2 = g^k$ for some $k \leftarrow_R \mathbb{Z}_p$ and $c_2 = H_2(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R)$, C can extract $g_2^k = (\sigma_5/(psk_{S,1}^{x_S} \cdot \sigma_2^{y_S c_2} \cdot \sigma_2^{L(m)}))^{1/K(m)}$ and computes $m = \sigma_1/\hat{e}(g_1, g_2^k)^{x_R \alpha}$.

Forgery. If C does not abort as a consequence of one of the queries above, $\mathcal{A}_2$ will, with probability at least ϵ, return the sender's identity u_{S^*}, the receiver's identity u_{R^*}, a message m^*, and valid forgery $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*)$ on m^*. If $F(u_{S^*}) \neq 0 \bmod p$ or $K(m^*) \neq 0 \bmod p$, then C will abort. If, on the other hand, $F(u_{S^*}) = 0 \bmod p$ and $K(m^*) = 0 \bmod p$, C retrieves u_S's user secret key $usk_{S^*} = (x_{S^*}, y_{S^*})$ and computes

$$\frac{\sigma_5^*}{\sigma_4^{*J(u^*)} \sigma_2^{*L(m^*)} \sigma_2^{*c_2^* y_{S^*}}} = \frac{g_2^{a\alpha x_{S^*}} (u' \prod_{i \in \mathcal{U}_{S^*}} u_i)^{k_u} (m' \prod_{k \in \mathcal{M}} m_k)^{k_m} g^{y_{S^*} c_2^* k_m}}{g^{J(u_{S^*}) k_u} g^{L(m^*) k_m} g^{k_m y_{S^*} c_2^*}} = g^{ab\alpha x_{S^*}},$$

where $c_2^* = H_2(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, u_{S^*}, u_{R^*}, upk_{S^*}, upk_{R^*})$. Thus, C can outputs g^{ab} as the solution to the CDH problem instance.

The probability analysis is similar to the proof in Lemma 1 to avoid repetition.

Theorem 2. *Our CL-SC scheme is indistinguishable against chosen ciphertext attacks (IND-CL-SC-CCA) in the standard model assuming the decisional BDDH problem is hard.*

This theorem follows Lemmas 3 and 4.

Lemma 3. *(Type I confidentiality). Our CL-SC scheme is (ϵ, t)-indistinguishable against Type I chosen ciphertext adversary with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-BDDH assumption holds in $\mathbb{G}_1$, where $\epsilon' \geq \frac{\epsilon}{32(q_e+q_s)q_s(n_u+1)(n_m+1)}$ and $t' = t + O((q_e n_u + q_s(n_u + n_m))\rho + (q_k + q_e + q_s)\tau)$.*

Proof. Let C be a BDH attacker who receives a random instance (g, g^a, g^b, g^c, Z) of the BDH problem and has to output a guess β, to show whether the challenge is a BDH tuple. Here g is a generator of $\mathbb{G}_1$, Z is randomly chosen from $\mathbb{G}_1$ and a, b, c are chosen randomly from $\mathbb{Z}_p$. $\mathcal{A}_1$ is a type I adversary who interacts with C. We show how C can use $\mathcal{A}_1$ as a subroutine to solve the BDH problem.

Initial. C first sets the system parameters described in Lemma 1. Note that in the **Initial** phase, C assigns $g_1 = g^\alpha$, $g_2 = g^b$. After that, C defines the functions $F(u), J(u), K(m), L(m)$ and $u', (u_i), m', (m_j)$ such that

$$u' \prod_{i \in \mathcal{U}} u_i = g_2^{F(u)} g^{J(u)} \quad \text{and} \quad m' \prod_{i \in \mathcal{M}} m_i = g_2^{K(m)} g^{L(m)}$$

Phase 1. $\mathcal{A}_1$ can perform a polynomially bounded number of queries including the Request-Public-Key, Reveal-Partial-Private-Key, Reveal-Secret-Key, Replace-Public-Key, Signcrypt and Unsigncrypt queries. The challenger C answers the queries of $\mathcal{A}_1$ identical to Lemma 1.

Challenge. $\mathcal{A}_1$ generates two equal length messages m_0, m_1, two identities u_{S^*} and u_{R^*} on which he wants to be challenged. $\mathcal{A}_1$ has not asked the private key extraction queries on u_{R^*} in **Phase 1**. After that, $\mathcal{A}_1$ sends m_0, m_1 and u_{S^*}, u_{R^*} to C. If $F(u_{R^*}) \neq 0 \bmod l_u$, C aborts the simulation. Otherwise, C takes a bit $\gamma \in_R \{0, 1\}$ and constructs a ciphertext of m_γ as follows. Let $upk_{S^*} = (upk_{S^*,1}, upk_{S^*,2}, upk_{S^*,3}) = (h^{x_{S^*}}, g_2^{y_{S^*}}, g^{y_{S^*}})$ and $upk_{R^*} = (upk_{R^*,1}, upk_{R^*,2}, upk_{R^*,3}) = (h^{x_{R^*}}, g_2^{y_{R^*}}, g^{y_{R^*}})$ be u_{S^*} and u_{R^*}'s current user public keys. C retrieves the corresponding user secret keys $usk_{S^*} = (x_{S^*}, y_{S^*})$ and $usk_{R^*} = (x_{R^*}, y_{R^*})$. Then C sets $\sigma_1^* = m_\gamma \cdot Z^{x_{R^*}}, \sigma_2^* = g^c, \sigma_3^* =$

$(g^c)^{J(u_{R^*})}$, selects $k^* \leftarrow_R \mathbb{Z}_p$ and computes $\sigma_4^* = (g_1^{x_{S^*}})^{-1/F(u_{S^*})} g^{k^*}$. Let $\mathsf{m}_\gamma = H_{\mathrm{m}}(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, u_{S^*}, u_{R^*}, upk_{S^*}, upk_{R^*}) \in \{0,1\}^{n_m}$ and assume $\mathsf{m}_\gamma[i]$ be the i-th bit of m_γ and let $\mathcal{M}_\gamma \subset \{1,\ldots,n_m\}$ be the set of indices i such that $\mathsf{m}_\gamma[i] = 1$. If $K(\mathsf{m}_\gamma) \neq 0 \bmod p$, C aborts the simulation. Otherwise, C sets $\sigma_5^* = (g_1^{x_{S^*}})^{-\frac{J(u_{S^*})}{F(u_{S^*})}} (u' \prod_{i \in \mathcal{U}_{S^*}} u_i)^{k^*} (g^c g^{y_{S^*} c_2^*})^{L(\mathsf{m}_\gamma)}$ and returns $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*)$ to $\mathcal{A}_1$. Here, $c_2^* = H_2(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, u_{S^*}, u_{R^*}, upk_{S^*}, upk_{R^*})$

Phase 2. $\mathcal{A}_1$ can ask a polynomially bounded number of queries adaptively again as in the **Phase 1**. However, he cannot make an unsigncryption query on σ^* under u_{S^*} and u_{R^*} to obtain the corresponding message in this stage.

Guess. Finally, $\mathcal{A}_1$ produces a bit γ'. If $\gamma' = \gamma$, C wins the game and indicates that $Z = \hat{e}(g,g)^{abc}$. Otherwise, C fails to solve the BDH problem.

The probability analysis is similar to the proof in Lemma 1 to avoid repetition.

Lemma 4. *(Type II confidentiality). Our CL-SC scheme is (ϵ, t)-indistinguishable against Type II chosen ciphertext adversary with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-BDDH assumption holds in $\mathbb{G}_1$, where where $\epsilon' \geq \frac{\epsilon}{32(q_s+q_u)q_u(n_u+1)(n_m+1)}$ and $t' = t + O((q_{pp}n_u + q_s(n_u+n_m))\rho + (q_k+q_s)\tau)$.*

Proof. Let C be a BDH attacker who receives a random instance (g, g^a, g^b, g^c, Z) of the BDH problem and has to output a guess β, to show whether the challenge is a BDH tuple. Here g is a generator of $\mathbb{G}_1$, Z is randomly chosen from $\mathbb{G}_1$ and a, b, c are chosen randomly from $\mathbb{Z}_p^*$. $\mathcal{A}_2$ is a type II adversary who interacts with C. We show how C can use $\mathcal{A}_2$ as a subroutine to solve the BDH problem.

Initial. C first sets the system parameters described in Lemma 2. Note that in the **Initial** phase, C selects $\alpha \in_R \mathbb{Z}_p^*$ and assigns $g_1 = g^\alpha$, $g_2 = g^b$. After that, C defines the functions $F(u), J(u), K(m), L(m)$ and $u', (u_i), m', (m_j)$ such that

$$u' \prod_{i \in \mathcal{U}} u_i = g_2^{F(u)} g^{J(u)} \quad \text{and} \quad m' \prod_{i \in \mathcal{M}} m_i = g_2^{K(m)} g^{L(m)}$$

Phase 1. $\mathcal{A}_2$ can perform a polynomially bounded number of queries including the Request-Public-Key, Reveal-Partial-Private-Key, Reveal-Secret-Key, Signcrypt and Unsigncrypt queries. The challenger C answers the queries of $\mathcal{A}_1$ identical to Lemma 2.

Challenge. $\mathcal{A}_2$ generates two equal length messages m_0, m_1, two identities u_{S^*} and u_{R^*} on which he wants to be challenged. $\mathcal{A}_2$ has not asked the private key extraction queries on u_{R^*} in **Phase 1**. After that, $\mathcal{A}_2$ sends m_0, m_1 and u_{S^*}, u_{R^*} to C. If $F(u_{R^*}) \neq 0 \bmod l_u$, C aborts the simulation. Otherwise, C takes a bit $\gamma \in_R \{0,1\}$ and constructs a ciphertext of m_γ as follows. Let $upk_{S^*} = (upk_{S^*,1}, upk_{S^*,2}, upk_{S^*,3}) = (\hat{e}(g_2, g^a)^{\alpha x_{S^*}}, g_2^{y_{S^*}}, g^{y_{S^*}})$ and $upk_{R^*} = (upk_{R^*,1}, upk_{R^*,2}, upk_{R^*,3}) = (\hat{e}(g_2, g^a)^{\alpha x_{R^*}}, g_2^{y_{R^*}}, g^{y_{R^*}})$ be u_{S^*} and u_{R^*}'s current user public keys. C retrieves the corresponding user secret keys $usk_{S^*} = (x_{S^*}, y_{S^*})$ and $usk_{R^*} = (x_{R^*}, y_{R^*})$. Then C sets $\sigma_1^* = m_\gamma \cdot Z^{x_{R^*}\alpha}, \sigma_2^* = g^c, \sigma_3^* = (g^c)^{J(u_{R^*})}$, selects $k^* \leftarrow_R \mathbb{Z}_p$ and computes $\sigma_4^* = (g^a)^{-x_{S^*}\alpha/F(u_{S^*})} g^{k^*}$. Let $\mathsf{m}_\gamma = H_{\mathrm{m}}(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, u_{S^*}, u_{R^*}, upk_{S^*}, upk_{R^*}) \in \{0,1\}^{n_m}$ and assume $\mathsf{m}_\gamma[i]$ be the i-th bit of m_γ and let $\mathcal{M}_\gamma \subset \{1,\ldots,n_m\}$ be the set of indices i such that $\mathsf{m}_\gamma[i] = 1$. If $K(\mathsf{m}_\gamma) \neq 0 \bmod p$, C aborts the simulation. Otherwise, C sets $\sigma_5^* = ((g^a)^{\alpha x_{S^*}})^{-\frac{J(u_{S^*})}{F(u_{S^*})}} (u' \prod_{i \in \mathcal{U}_{S^*}} u_i)^{k^*} (g^c g^{y_{S^*} c_2^*})^{L(\mathsf{m}_\gamma)}$ and returns $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*)$ to $\mathcal{A}_1$. Here, $c_2^* = H_2(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, u_{S^*}, u_{R^*}, upk_{S^*}, upk_{R^*})$

Phase 2. $\mathcal{A}_1$ can ask a polynomially bounded number of queries adaptively again as in the **Phase 1**. However, he cannot make an unsigncryption query on σ^* under u_{S^*} and u_{R^*} to obtain the corresponding message in this stage.

Guess. Finally, $\mathcal{A}_1$ produces a bit γ'. If $\gamma' = \gamma$, C wins the game and indicates that $Z = \hat{e}(g, g)^{abc}$. Otherwise, C fails to solve the BDH problem.

The probability analysis is similar to the proof in Lemma 1 to avoid repetition.

6. Conclusion

It is desirable to devise CL-SC schemes secure in the standard model. We have showed that Jin *et al*'s CL-SC scheme is not secure against the malicious KGC attacks. In this paper, motivated by Bellare and Shoup's one-time signature, Hwang *et al.*'s certificateless encryption and Li *et al.*'s identity-based signcryption, we proposed an CL-SC scheme provably secure in the standard model. It is believed to be the first in the literature to achieve the provably secure CL-SC without random oracles. Our future work consists of constructing CL-SC scheme secure in the standard model without one-time signature.

Acknowledgement

This work is partially supported by National Natural Science Foundation of China under Grant Nos. 61003230, 61370026, 61300191 and 61272029, Chongqing Key Lab of Computer Network and Communication Technology under Grant No. CY-CNCL-2012-02, National Key Technology R&D Program (2009BAG12A10), China Railway Ministry major task (2008G017-A) and the State Key Laboratory of Rail Traffic Control and Safety (RCS2009ZT007). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

[1] S.S. Al-Riyami, K.G. Paterson, Certificateless public key cryptography, Advances in Cryptology-ASIACRYPT 2003, Springer-Verlag, LNCS 2849, pp. 452-473, 2003.

[2] D. Aranha, R. Castro, J. Lopez, *et al.*, Efficient certificateless signcryption, 2008, Available: http://sbseg2008.inf.ufrgs.br/proceedings/data/pdf/st0301resumo.pdf.

[3] M.H. Au, J. Chen, Y. Mu, *et al.*, Malicious KGC attacks in certificateless cryptography, ACM symposium on Information, computer and communications security (ASIACCS'2007), pp. 302-311, 2007

[4] J. Baek, R. Steinfeld, and Y. Zheng, Formal proofs for the security of signcryption, Public Key Cryptography-PKC 2002, Springer-Verlag, LNCS 2274, pp. 80–98, 2002.

[5] F. Bao, R. H. Deng. A signcryption scheme with signature directly verifiable by public key, Public Key Cryptography-PKC'98, Springer-Verlag, LNCS 1431, pp. 55-59, 1998.

[6] M. Barbosa, P. Farshim, Certificateless signcryption, in: M. Abe, V.D. Gligor (Eds.), ASIACCS, ACM, 2008, pp. 369-372.

[7] P.S.L.M. Barreto, B. Libert, N. McCullagh, and J.J. Quisquater, Efficient and provably-secure identity-based signatures and signcryption from bilinear maps, Advances in Cryptology-ASIACRYPT 2005, Springer-Verlag, LNCS 3788, pp. 515-532, 2005.

[8] P. Barreto, A.M. Deusajute, E.D.S Cruz, *et al.*, Toward efficient certificateless signcryption from (and without) bilinear pairings, 2008, Available: http://www.redes.unb.br/ceseg/anais/2008/data/pdf/st03_03_artigo.pdf

[9] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, 1^{st} ACM Conf. on Computer and Communications Security (CCS 1993), pp. 62-72, 1993.

[10] M. Bellare, S. Shoup, Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles, Public Key Cryptography-PKC 2007, LNCS 4450, Springer-Verlag, pp. 201-216, 2007.

[11] X. Boyen, Multipurpose identity-based signcryption: a swiss army knife for identity-based cryptography, Advances in Cryptology-CRYPTO 2003, Springer-Verlag, LNCS 2729, pp. 383-399, 2003.

[12] R. Canetti, O. Goldreich, S. Halevi, The random oracle methodology, revisited, Proc. 30^{th} Annual Symp. on Theory of Computing (STOC'98), pp. 209-218, 1998.

[13] L. Chen and J. Malone-Lee, Improved identity-based signcryption, Public Key Cryptography-PKC 2005, Springer-Verlag, LNCS 3386, pp. 362-379, 2005.

[14] S.S.M. Chow, S.M. Yiu, L.C.K. Hui, and K.P. Chow, Efficient forward and provably secure ID-based signcryption scheme with public verifiability and public ciphertext authenticity, 6^{th} International Conference on Information Security and Cryptology (ICISC 2003), Springer-Verlag, LNCS 2971, pp. 352–369, 2004.

[15] A. W. Dent, B. Libert, K. G. Paterson, Certificateless Encryption Schemes Strongly Secure in the Standard Model, Public Key Cryptography-PKC 2008, Springer-Verlag, LNCS 4939, pp. 344-359, 2008.

[16] W. Diffie, M. E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory, 22(6): 644-654, 1976.

[17] C. Gamage, J. Leiwo, and Y. Zheng, Encrypted message authentication by firewalls, Public Key Cryptography-PKC'99, Springer-Verlag, LNCS 1560, pp. 69-81, 1999.

[18] Q. Huang, D.S. Wong, Generic Certificateless Encryption in the Standard Model, 2^{nd} IWSEC workshop (IWSEC 2007), Springer-Verlag, LNCS 4752, pp. 278C291, 2007.

[19] Y. H. Hwang, J. K. Liu, S. S.M. Chow, Certificateless Public Key Encryption Secure against Malicious KGC Attacks in the Standard Model, Journal of Universal Computer Science, 14(3): 463-480, 2008.

[20] F. Li, M. K. Khan, K. Alghathbar, T. Takagi, Identity-based online/offline signcryption for low power devices, Journal of Network and Computer Applications, vol. 35, no. 1, pp. 340-347, 2012.

[21] X. Li, Y. Xiong, J. Ma, W. Wang, An efficient and security dynamic identity based authentication protocol for multi-server architecture using smart cards, Journal of Network and Computer Applications, vol. 35, no. 2, pp. 763-769, 2012.

[22] Z. Jin, Q. Wen, H. Zhang, A supplement to Liu et al.'s certificateless signcryption scheme in the standard model, Cryptology ePrint Archive, Report 2010/252, 2010. Available: http://eprint.iacr.org/2010/252

[23] H.Y. Jung, D.H. Lee, J.I. Lim, and K.S. Chang, Signcryption schemes with forward secrecy, 2^{nd} International Workshop on Information Security Application (WISA 2001), pp. 463-475, Seoul, Korea, 2001.

[24] X. Li, H. Qian, J. Weng, Y. Yu, Fully secure identity-based signcryption scheme with shorter signcryptext in the standard model, Mathematical and Computer Modelling, 57(3-4): 503-511, 2013.

[25] B. Libert and J.J. Quisquater, A new identity based signcryption schemes from pairings' Proc. 2003 IEEE information theory workshop, pp. 155–158, Paris, France, 2003.

[26] J.K. Liu, M.H. Au, W. Susilo, Self-Generated-Certificate Public Key Cryptography and certificateless signature/encryption scheme in the standard model, 2^{nd} ACM symposium on Information, computer and communications security (ASIACCS 2007), pp. 273-283, 2007.

[27] Z. Liu, Y. Hu, X. Zhang, H. Ma, Certificateless signcryption scheme in the standard model, Information Sciences, 180(3): 452-464, 2010.

[28] J. Malone-Lee, Identity based signcryption, Cryptology ePrint Archive, Report 2002/098, 2002. Available: http://eprint.iacr.org/2002/098, 2002.

[29] J. Malone-Lee, W. Mao, Two birds one stone: signcryption using RSA, The Cryptographers' Track at the RSA Conference 2003 (CT-RSA 2003), Springer-Verlag, LNCS 2612, pp. 211-226, 2003.

[30] S. Miao, F. Zhang, S. Li, Y. Mu, On security of a certificateless signcryption scheme, Information Sciences, 232(20): 475-481, 2013.

[31] K. G. Paterson, J. C. N. Schuldt, Efficient Identity-based Signatures Secure in the Standard Model, 11^{th} Australasian Conference on Information Security and Privacy (ACISP 2006), Springer-Verlag, LNCS 4058, pp. 207-222, 2006.

[32] B. Qin, H. Wang, Q. Wu, J. Liu, J. Domingo-Ferrer, A New Identity Based Signcryption Scheme in the Standard Model, 4^{th} International Conference on Intelligent Networking and Collaborative Systems (InCoS 2012), pp. 606-611, 2012.

[33] A. Shamir, Identity-based cryptosystems and signature schemes, Advances in Cryptology-CRYPTO 1984, Springer-Verlag, LNCS 196, pp. 47-53, 1984.

[34] J. B. Shin, K. Lee, and K. Shim, New DSA-verifiable signcryption schemes, 5^{th} International Conference on Information Security and Cryptology (ICISC 2002), Springer-Verlag, LNCS 2587, pp. 35-47, 2002.

[35] B. Waters, Efficient identity based encryption without random oracles, Advances in Cryptology-EUROCRYPT 2005, Springer-Verlag, LNCS 3494, pp. 114-127, 2005.

[36] J. Weng, G. Yao, R.H. Deng, M.R. Chen, X. Li, Cryptanalysis of a certificateless signcryption scheme in the standard model, Information Sciences, 181(3): 661C667, 2011.

[37] C. Wu, Z. Chen, A new efficient certificateless signcryption scheme, Proceedings of IEEE International Symposium on Information Science and Engineering, Shanghai, China, 2008, pp. 661C664.

[38] W. Xie, Z. Zhang, Efficient and Provably Secure Certificateless Signcryption from Bilinear Maps, Cryptology ePrint Archive, Report 2009/578, 2009. Available: http://eprint.iacr.org/2009/578.

[39] H. Xiong, Z. Qin, F. Li, An improved certificateless signature scheme secure in the standard model, Fundamenta Informaticae, 88(1-2), pp. 193-206, 2008.

[40] H. Xiong, Z. Qin, F. Li, New Identity-based Three-party authenticated key agreement protocol with Provable

Security, Journal of Network and Computer Applications, vol. 36, no. 2, pp. 927-932, 2013.

[41] D.H. Yum, P.J. Lee, New signcryption schemes based on KCDSA, 4^{th} International Conference on Information Security and Cryptology (ICISC 2001), Springer-Verlag, LNCS 2288, pp. 305-317, 2002.

[42] Y. Zheng, Digital signcryption or how to achieve cost (signature & encryption) $\ll$ cost (signature) + cost(encryption), Advances in Cryptology-CRYPTO'97, Springer-Verlag, LNCS 1294, pp. 165–179, 1997.

[43] Y. Zheng, H. Imai, How to construct efficient signcryption schemes on elliptic curves, Information Processing Letters, 68(5): 227-233, 1998.