Certificateless Signcryption without Pairing

Wenjian Xie[*] Zhang Zhang

College of Mathematics and Computer Science
Guangxi University for Nationalities, Nanning 530006, China

Abstract. Certificateless public key cryptography is receiving significant attention because it is a new paradigm that simplifies the traditional PKC and solves the inherent key escrow problem suffered by ID-PKC. Certificateless signcryption is one of the most important security primitives in CL-PKC. However, to the best of our knowledge, all constructions of certificateless signcryption (CLSC) in the literature are built from bilinear maps which need costly operations. In the paper, motivated by certificateless encryption schemes proposed in [3, 21], we present a pairing-free CLSC scheme, which is more efficient than all previous constructions.

Keywords: Certificateless; Signcryption scheme; Bilinear pairing

1 Introduction

In a traditional public key cryptography (PKC), any user of the system who wants to communicate with others must obtain their authorized public key, that means any public key should be associated with the owner by a certificate, which is a signature issued by the trusted Certificate Authority (CA). However this brings a large amount of computation, communication cost and certificate management problems. In order to solve those problems, Shamir [20] firstly introduced the concept of identity based cryptography (ID-PKC) in 1984. A user can use an email address, an IP address or any other information related his identity, that is publicly know and unique in the whole system, as his public key. The advantage of an identity based cryptography is that anyone can simply use the user's identity to communicate with each other. This can be done even before the user gets its private key from the Key Generation Center (KGC). However, the user must completely trust KGC, which can impersonate any user to sign or decrypt of any message. This issue is generally referred to as key escrow problem in identity based cryptography.

[*]Corresponding author (W. Xie). E-mail: wjxieem@gmail.com.

In 2003, Al-Riyami and Paterson [1] introduced the concept of certificateless public key cryptography (CL-PKC), which eliminate the use of certificates as in the traditional PKC and solve the key escrow problem that is inherent in identity based cryptography. In CL-PKC, the KGC is involved to issue a user's partial key. Then the user independently generates his public/private key pair (pk_{ID}, sk_{ID}) use the partial key and a secret value chosen by himself, and publishes pk_{ID}.

In 2008, Barbosa and Farshim introduced the notion of certificateless signcryption (CLSC), which is one of the most important security primitives in CL-PKC, and proposed the first CLSC scheme [5]. And aimed at designing an efficient CLSC scheme, Wu and Chen proposed an new CLSC scheme [23], which was found insecure by Selvi et al. [18]. Recently, Liu et al. [14] proposed the first CLSC scheme in the standard model from Waters's identity-based encryption scheme [22], which can resist the malicious-but-passive KGC attacks [2], but unfortunately it was found insecure by Selvi et al. [19]. And Xie et al. [24] proposed another CLSC scheme from bilinear Maps, which requires two pairing operations in the signcrypt and unsigncrypt phases.

Our Contribution. To the best of our knowledge, all concrete constructions of certificateless signcryption in the literature are built from bilinear maps. We note that in pairing based cryptosystems, although numerous papers discuss the complexity of pairings and how to speed up the pairing computation [6, 8, 11, 12], the computation of the pairing still remains time-consuming. In this paper, motivated by certificateless encryption scheme proposed in [3, 21]. we present a pairing-free certificateless signcryption scheme, which is more efficient than all previous constructions [5, 14, 23, 24].

Organization. The rest of this paper is organized as follows: In next Section, we describe some preliminaries, including our complexity assumptions, security definition of signature and the notion of certificateless signcryption scheme. We describe its security models in Section 3 and propose our certificateless signcryption scheme in Section 4. In Section 5, we present its security analysis. Finally, we conclude this paper in Section 6.

2 Preliminaries

2.1 Definitions

Definition 1. The Gap Diffie-Hellman Problem (GDHP) [4] is given, in addition to $(p, q, g, g^{\alpha}, g^{\beta})$ for unknown $\alpha, \beta \in \mathbb{Z}_q^*$, access to a Decisional Diffie-Hellman (DDH) oracle $\mathsf{O}^{\mathsf{DDH}}(\cdot,\cdot,\cdot,\cdot)$, which, on input (g, g^a, g^b, z), outputs 1 if $z = g^{ab}$ and 0 otherwise, and tries to find the Diffie-Hellman key $g^{\alpha\beta}$.

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the GDHP is defined as

$$Adv_{\mathcal{A}}^{GDHP} = Pr[\mathcal{A}(p, q, g, g^{\alpha}, g^{\beta} | \mathsf{O}^{\mathsf{DDH}}(\cdot,\cdot,\cdot,\cdot)) = g^{\alpha\beta}].$$

The GDH Assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{GDHP}$ is negligible.

Definition 2. The Gap Discrte Logarithm Problem (GDLP) [4] is given, in addition to (p, q, g, g^{α}) for unknown $\alpha \in \mathbb{Z}_q^*$, access to a restricted DDH oracle $\mathsf{O}^{\mathsf{rDDH}}(g, g^{\alpha}, \cdot, \cdot)$, which, on input (g^{β}, z), outputs 1 if $z = g^{\alpha\beta}$ and 0 otherwise, and to compute α.

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the GDLP is defined as

$$Adv_{\mathcal{A}}^{GDLP} = Pr[\mathcal{A}(p, q, g, g^{\alpha}|\mathsf{O}^{\mathsf{rDDH}}(g, g^{\alpha}, \cdot, \cdot)) = \alpha].$$

The GDL Assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{GDLP}$ is negligible.

Definition 3(Security of Signature). A signature scheme is (ϵ, t, q_h, q_s)-secure(existentially unforgeable under an adaptive chosen message attack [13]) if no t-time adversary $\mathcal{A}$, making at most q_h queries to hash function and q_s signature queries, has an advantage of at least ϵ to produce a valid message-signature pair.

2.2 Certificateless Signcryption Scheme

A certificateless signcryption scheme is defined by the following seven algorithms:

Setup: This algorithm takes a security parameter k as input and returns the system parameters params and a secret master key master-key.

Partial-Key-Extract: This algorithm takes params, master-key and a user's identity ID as input. It returns a partial private key d_{ID} and a partial public key p_{ID} corresponding to the user.

Set-Secret-Value: Taking params and a user's identity ID as input, this algorithm generates a secret value s_{ID}.

Set-Public-Key: Taking params, a user's partial public key p_{ID} and his secret value s_{ID} as input, this algorithm generates pk_{ID} for the user with identity ID.

Set-Private-Key: It takes params, a user's partial private key d_{ID} and his secret value s_{ID} as input, and returns the user's full private key sk_{ID}.

Signcrypt: This algorithm takes as input the sender's private key $sk_{ID_{\mathbb{S}}}$, the receiver's identity $ID_{\mathbb{R}}$ and public key $pk_{ID_{\mathbb{R}}}$, and a message m, and returns a ciphertext σ. We write $\sigma = \mathsf{Signcrypt}(\mathsf{sk}_{\mathsf{ID}_{\mathbb{S}}}, \mathsf{ID}_{\mathbb{R}}, \mathsf{pk}_{\mathsf{ID}_{\mathbb{R}}}, \mathsf{m})$.

Unsigncrypt: It takes the sender's identity $ID_{\mathbb{S}}$ and public key $pk_{ID_{\mathbb{S}}}$, the receiver's private $sk_{ID_{\mathbb{R}}}$ and the corresponding ciphertext σ as input, and outputs the message m if the ciphertext σ is valid, or the symbol $\perp$ otherwise. We write $\varrho = \mathsf{Unsigncrypt}(\mathsf{ID}_{\mathbb{S}}, \mathsf{pk}_{\mathsf{ID}_{\mathbb{S}}}, \mathsf{sk}_{\mathsf{ID}_{\mathbb{R}}}, \sigma)$, where ϱ is the message m or the symbol $\perp$.

params, as an implied inputs to Signcrypt and Unsigncrypt algorithms, is omitted. The Setup and Partial-Key-Extract algorithms are performed by KGC. Once partial private key d_{ID} and partial public key p_{ID} is given to the user via secure channel, the user runs Set-Secret-Value algorithm and generates his own public/private key pair.

3 Security Model for Signcryption

In [5], Barbosa and Farshim defined the formal security notions for certificateless signcryption schemes. These notions are natural adaptations from the security notions of identity-based signcryption [9, 10] by considering two different type adversaries, a Type I adversary $\mathcal{A}_I$ and a Type II adversary $\mathcal{A}_{II}$, and include the indistinguishability against adaptive chosen ciphertext attacks and the existential unforgeability against adaptive chosen message attacks. The adversary $\mathcal{A}_I$ represents a normal third party attacker against the CLSC scheme. That is, $\mathcal{A}_I$ is not allowed to access to the master-key but $\mathcal{A}_I$ may requests public key and replaces public keys with values of its choice. The adversary $\mathcal{A}_{II}$ represents a malicious KGC who generates partial private key of users. The adversary $\mathcal{A}_{II}$ is allowed to have access to the master-key but not replace a public key. Note that, as in [5, 9, 10], we do not consider attacks targeting ciphertext where the sender and receiver identities are the same. In particular we disallow such queries to relevant oracles and do not accept this type of ciphertext as a valid forgery.

3.1 Confidentiality Model for Certificateless Signcryption

The confidentiality property (indistinguishability of encryptions under adaptively chosen ciphertext attacks (IND-CCA2)) required for certificateless signcryption is captured by the following two games against $\mathcal{A}_I$ and $\mathcal{A}_{II}$.

Game IND-CCA2-I. Now we illustrate the first game performed between a challenger C and a Type I adversary $\mathcal{A}_I$ for a certificateless signcryption scheme.

Initialization: C runs the algorithm Setup on input a security parameter k, and obtains master-key and params, and sends params to $\mathcal{A}_I$.

Find stage: The adversary $\mathcal{A}_I$ performs a polynomially bounded number of queries. These queries may be made adaptively, i.e. each query may depend on the answers to the previous queries.

- Hash Queries: $\mathcal{A}_I$ can request the hash values of any input.
- Partial Key Extraction: $\mathcal{A}_I$ is able to ask for the partial private key d_{ID} and partial public key p_{ID} for any ID. C computes the partial private key d_{ID} and partial public key p_{ID} corresponding to the identity ID and returns them to $\mathcal{A}_I$.
- Public Key Extraction: On receiving a public key extraction for any identity ID, C computes the corresponding public key pk_{ID} and sends it to $\mathcal{A}_I$.
- Private Key Extraction: For any ID, C computes the private key sk_{ID} corresponding to the identity ID and sends sk_{ID} to $\mathcal{A}_I$. Here, $\mathcal{A}_I$ is not allowed to query this oracle on any identity for which the corresponding public key has been replaced. This restriction is imposed due to the fact that it is unreasonable to expect that the challenger is able to provide a full private key for a user for which it does not know the secret value.
- Public Key Replacement: For any identity ID, $\mathcal{A}_I$ can can replace the public key for any identity with any value of its choice. The current value of an entity's public key

is used by the challenger in any computation or response to the adversary's requests.

– Signcrypt Queries: $\mathcal{A}_I$ produces a sender's identity $ID_{\mathbb{S}}$, a receiver's identity $ID_{\mathbb{R}}$ and a message m. C returns ciphertext σ=Signcrypt($\mathsf{sk}_{\mathsf{ID}_{\mathbb{S}}}$, $\mathsf{ID}_{\mathbb{R}}$, $\mathsf{pk}_{\mathsf{ID}_{\mathbb{R}}}$, m) to $\mathcal{A}_I$ as the response of signcryption oracle's answer. Note that, it is possible that the public key $pk_{ID_{\mathbb{S}}}$ has been replaced earlier by $\mathcal{A}_I$, In this case, to correctness of the signcryption oracle's answer, we assume that $\mathcal{A}_I$ additionally submits the corresponding secret value to C. And we disallow queries where $ID_{\mathbb{S}} = ID_{\mathbb{R}}$.
– Unsigncrypt Queries: $\mathcal{A}_I$ produces a sender's identity $ID_{\mathbb{S}}$, a receiver's identity $ID_{\mathbb{R}}$ and a ciphertext σ. C sends the result of Unsigncrypt($\mathsf{ID}_{\mathbb{S}}$, $\mathsf{pk}_{\mathsf{ID}_{\mathbb{S}}}$, $\mathsf{sk}_{\mathsf{ID}_{\mathbb{R}}}$, σ) to $\mathcal{A}_I$. Note that, it is possible that the public key $pk_{ID_{\mathbb{R}}}$ has been replaced earlier by $\mathcal{A}_I$. In this case, to correctness of the unsigncryption oracle's answer, we assume that $\mathcal{A}_I$ additionally submits the corresponding secret value to C. Again, we disallow queries where $ID_{\mathbb{S}} = ID_{\mathbb{R}}$.

Challenge: At the end of **Find stage**, $\mathcal{A}_I$ returns two distinct messages m_0 and m_1 (assumed of equal length), a sender identity $ID^*_{\mathbb{S}}$ and a receiver identity $ID^*_{\mathbb{R}}$, on which it wishes to be challenged. The adversary must have made no partial key extraction and private key extraction on $ID^*_{\mathbb{R}}$. C picks randomly a bit $\delta \in \{0,1\}$, computes σ^*=Signcrypt($\mathsf{sk}_{\mathsf{ID}^*_{\mathbb{S}}}$, $\mathsf{ID}^*_{\mathbb{R}}$, $\mathsf{pk}_{\mathsf{ID}^*_{\mathbb{R}}}$, m_δ) and returns it to $\mathcal{A}_I$.

Guess stage: $\mathcal{A}_I$ asks a polynomial number of queries adaptively again as in the **Find stage**. It is not allowed to extract the partial key and private key corresponding to $ID^*_{\mathbb{R}}$ and it is not allowed to make an unsigncrypt query on σ^* with sender $ID^*_{\mathbb{S}}$ and receiver $ID^*_{\mathbb{R}}$ unless the public key $pk_{ID^*_{\mathbb{S}}}$ of the sender or that of the receiver $pk_{ID^*_{\mathbb{R}}}$ has been replaced after the challenge was issued.

Eventually, $\mathcal{A}_I$ outputs a bit δ' and wins the game if δ=δ'.

$\mathcal{A}_I$'s advantage is defined as $Adv^{IND-CCA2-I}_{\mathcal{A}_I} = 2Pr[\delta = \delta'] - 1$.

Game IND-CCA2-II. This is the second game where C interacts with adversary $\mathcal{A}_{II}$ as follows:

Initialization: C runs the algorithm Setup on input a security parameter k to generate master-key and params, and sends master-key and params to $\mathcal{A}_{II}$.

Find stage: In this stage, $\mathcal{A}_{II}$ may adaptively make a polynomially bounded number of queries as in **Game IND-CCA2-I**. The only constraint is that $\mathcal{A}_{II}$ can not replace any public keys. Obviously, $\mathcal{A}_{II}$ can compute the partial private key of any identities by itself with the master-key and can get the partial public key of any identities by extract the corresponding public key.

Challenge: At the end of **Find stage**, $\mathcal{A}_{II}$ returns two distinct messages m_0 and m_1 (assumed of equal length), a sender identity $ID^*_{\mathbb{S}}$ and a receiver identity $ID^*_{\mathbb{R}}$, on which it wishes to be challenged. The adversary must have made no private key extraction on $ID^*_{\mathbb{R}}$. C picks randomly a bit $\delta \in \{0,1\}$, computes σ^*=Signcrypt($\mathsf{sk}_{\mathsf{ID}^*_{\mathbb{S}}}$, $\mathsf{ID}^*_{\mathbb{R}}$, $\mathsf{pk}_{\mathsf{ID}^*_{\mathbb{R}}}$, m_δ) and returns it to $\mathcal{A}_{II}$.

Guess stage: $\mathcal{A}_{II}$ asks a polynomial number of queries adaptively again as in the **Find stage**. It is not allowed to extract the private key corresponding to $ID^*_{\mathbb{R}}$ and it is not allowed to make an unsigncrypt query on σ^* with sender $ID^*_{\mathbb{S}}$ and receiver $ID^*_{\mathbb{R}}$.

Eventually, $\mathcal{A}_{II}$ outputs a bit δ' and wins the game if δ=δ'.

$\mathcal{A}_{II}$'s advantage is defined as $Adv^{IND-CCA2-II}_{\mathcal{A}_{II}} = 2Pr[\delta = \delta'] - 1$.

Note that the security models described above deals with insider security since the adversary is assumed to have access to the private key of the sender of ciphertext σ^*. This means that the confidentiality is preserved even if a sender's private key is compromised.

Definition 5 (IND-CCA2). An CLSC scheme is said to be IND-CCA2 secure if no polynomially bounded adversary $\mathcal{A} \in \{\mathcal{A}_I, \mathcal{A}_{II}\}$ has a non-negligible advantage wins the games described above(**Game IND-CCA2-I**, **Game IND-CCA-II**).

3.2 Unforgeability Model for Certificateless Signcryption

The authenticity property (existential unforgeability against chosen message attacks (EUF-CMA)) for certificateless signcryption is captured by the following two games against $\mathcal{A}_I$ and $\mathcal{A}_{II}$, respectively.

Game EUF-CMA-I. This is the game where $\mathcal{A}_I$ interacts with its Challenger C as follows:

Initialization: C runs the algorithm Setup on input a security parameter k to generate master-key and params, and sends params to $\mathcal{A}_I$.

Queries: The adversary $\mathcal{A}_I$ performs a polynomially bounded number of queries adaptively as in **Game IND-CCA2-I** game.

Output: Finally, $\mathcal{A}_I$ produces a new triple $(ID^*_{\mathbb{S}}, ID^*_{\mathbb{R}}, \sigma^*)$ (i.e. a triple that was not produced by the signcryption oracle) where the partial key and the private key of $ID^*_{\mathbb{S}}$ was not extract and wins the game if the result of Unsigncrypt($\mathsf{ID}^*_{\mathbb{S}}$, $\mathsf{pk}_{\mathsf{ID}^*_{\mathbb{S}}}$, $\mathsf{sk}_{\mathsf{ID}^*_{\mathbb{R}}}$, σ^*) is not the $\perp$ symbol.

The adversary $\mathcal{A}_I$'s advantage is its probability of victory.

Game EUF-CMA-II. This is the game where $\mathcal{A}_{II}$ interacts with its Challenger C as follow:

Initialization: C runs the algorithm Setup on input a security parameter k to generate master-key and params, and sends params and master-key to $\mathcal{A}_I$.

Queries: The adversary $\mathcal{A}_{II}$ performs a polynomially bounded number of queries adaptively as in **Game IND-CCA2-II** game.

Output: Finally, $\mathcal{A}_I$ produces a new triple $(ID^*_{\mathbb{S}}, ID^*_{\mathbb{R}}, \sigma^*)$ (i.e. a triple that was not produced by the signcryption oracle) where the private key of $ID^*_{\mathbb{S}}$ was not extract and wins the game if the result of Unsigncrypt($\mathsf{ID}^*_{\mathbb{S}}$, $\mathsf{pk}_{\mathsf{ID}^*_{\mathbb{S}}}$, $\mathsf{sk}_{\mathsf{ID}^*_{\mathbb{R}}}$, σ^*) is not the $\perp$ symbol.

The adversary $\mathcal{A}_{II}$'s advantage is its probability of victory.

Note that this definition allows the adversary have access to the secret key of the receiver of the forgery, which guarantees the insider security.

Definition 6 (UF-CMA). An CLSC scheme is said to be EUF-CMA secure if no polynomially bounded adversary $\mathcal{A} \in \{\mathcal{A}_I, \mathcal{A}_{II}\}$ has a non-negligible advantage wins the games described above(**Game EUF-CMA-I**, **Game EUF-CMA-II**).

4 A CLSC without Pairing

Our scheme is motivated by certificateless encryption scheme proposed in [3, 21].

Setup: This algorithm takes as input a security parameter k to generate two primes p, q such that $q > 2^k$ and $q|(p-1)$. It then performs the following:

- Pick an element g from $\mathbb{Z}_p^*$ with order q.
- Randomly select $x \in_R \mathbb{Z}_q^*$, as master-key, and compute $y = g^x$.
- Choose cryptographic hash functions $H_1 : \{0,1\}^* \times \mathbb{Z}_p^* \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \times (\mathbb{Z}_p^*)^2 \to \mathbb{Z}_q^*$, $H_3 : \{0,1\}^n \times (\mathbb{Z}_p^*)^7 \to \mathbb{Z}_q^*$, $H_4 : \{0,1\}^n \times (\mathbb{Z}_p^*)^7 \to \mathbb{Z}_q^*$ and $H_5 : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \{0,1\}^n$, where n is the length of message to be signcrypted.
- The system parameters are params $=< p, q, n, g, y, H_1, H_2, H_3, H_4, H_5 >$ and publish params.

Partial-Key-Extract: Given params, master-key and an identity $ID \in \{0,1\}^*$, this algorithm works as follows: selects r_{ID}, $r'_{ID} \in \mathbb{Z}_q^*$ Computes $\omega_{ID} = g^{r_{ID}}$, $d_{ID} = r_{ID} + xH_1(ID, \omega_{ID})$, $\nu_{ID} = g^{r'_{ID}}$, $\sigma_{ID} = r'_{ID} + xH_2(ID, \omega_{ID}, \nu_{ID})$, and returns the partial private key d_{ID} and the partial public key $p_{ID} = (\omega_{ID}, \nu_{ID}, \sigma_{ID})$.

Set-Secret-Value: This algorithm takes as input params and a user's identity ID. It picks a random value $z \in_R \mathbb{Z}_q^*$ and outputs the secret value $s_{ID} = z$.

Set-Private-Key: Taking params, d_{ID} and s_{ID} as input, this algorithm returns the private key $sk_{ID} = (d_{ID}, s_{ID})$.

Set-Public-Key: Taking params, p_{ID} and s_{ID} as input, this algorithm computes $\mu_{ID} = g^{s_{ID}}$ and returns the public key $pk_{ID} = (\mu_{ID}, \omega_{ID}, \nu_{ID}, \sigma_{ID})$.

Signcrypt: To send a message $m \in \{0,1\}^n$ to Bob with identity B and public key pk_B, Alice with private key sk_A works as follow:

- Check whether $g^{\sigma_B} = \nu_B y^{H_2(B, \omega_B, \nu_B)}$. If not, output $\perp$ and abort signcryption.
- Randomly select $r \in_R \mathbb{Z}_q^*$, and computes $t = g^r$ and $c = H_5(\mu_B^r, \omega_B^r y^{H_1(B,\omega_B)r}) \oplus m$.
- Compute $h = H_3(m, t, \mu_B^r, \omega_B^r y^{H_1(B,\omega_B)r}, \mu_A, \omega_A, \mu_B, \omega_B)$, $h' = H_4(m, t, \mu_B^r, \omega_B^r y^{H_1(B,\omega_B)r}, \mu_A, \omega_A, \mu_B, \omega_B)$ and $s = r - hd_A - h' s_A$.
- Set ciphertext $\sigma = (c, s, t)$.

Unsigncrypt: To unsigncrypt a ciphertext $\sigma = (c, s, t)$ from Alice with identity A and public key pk_A, Bob with private key sk_B acts as follows:

- Check whether $g^{\sigma_A} = \nu_A y^{H_2(A, \omega_A, \nu_A)}$. If not, output $\perp$ and abort unsigncryption.
- Compute $m = c \oplus H_5(t^{s_B}, t^{d_B})$.
- Set $h = H_3(m, t, t^{s_B}, t^{d_B}, \mu_A, \omega_A, \mu_B, \omega_B)$ and $h' = H_4(m, t, t^{s_B}, t^{d_B}, \mu_A, \omega_A, \mu_B, \omega_B)$.
- Accept m if and only if $t = g^s \omega_A^h y^{H_1(A,\omega_A)h} \mu_A^{h'}$ hold, return $\perp$ otherwise.

Consistency: The correctness of the proposed scheme can be easily verified with following:

$$\mu_B^r = g^{s_B r} = g^{r s_B} = t^{s_B},$$

$$\omega_B^r y^{H_1(B,\omega_B)r} = g^{r_B r} g^{x H_1(B,\omega_B)r} = g^{r(r_B + x H_1(B,\omega_B))} = g^{r d_B} = t^{d_B}$$

and

$$g^s \omega_A^h y^{H_1(A,\omega_A)h} \mu_A^{h'} = g^{r - h d_A - h' s_A} g^{r_A h} g^{x H_1(A,\omega_A)h} g^{s_A h'} = g^r = t.$$

5 Security Analysis of the Proposed Scheme

In this section, we will provide two formal proofs that our scheme is IND-CCA2 secure under the GDH Assumption and UF-CMA secure under the GDL Assumption. We now present the security analysis of the proposed scheme in the random oracle model [7].

Theorem 1. Under the GDH Assumption, the proposed CLSC scheme is IND-CCA2 secure in the random oracle model.

This theorem follows from Lemmas 1 and 2.

Lemma 1. Let us assume that there exists an IND-CCA2-I adversary $\mathcal{A}_I$ has non-negligible advantage ϵ against our scheme when running in time $\mathcal{T}$, asking q_i queries to random oracles H_i $(i = 1, 2, \cdots, 5)$, q_{pak} partial key queries, q_{sk} private key queries, q_{pk} public key requests, q_{pkr} public key replacement queries, q_s signcryption queries and q_u unsigncryption queries. Assume that the Schnorr signature [16, 17] is $(\epsilon', \mathcal{T}, q_2, q_{pak})$-secure. Then there is an algorithm C to solve the GDHP with probability

$$\epsilon'' \geq \frac{\epsilon}{q_3 + q_5 + q_s} \frac{(1-\epsilon')^{q_{pkr}}}{q_{pk}} (1 - q_s \frac{2q_3 + q_4 + q_5 + 3q_s}{2^k})(1 - \frac{q_u}{2^k})$$

within a time $\mathcal{T}' < \mathcal{T} + (q_1 + q_2)O(1) + (q_3 + q_4 + q_5)(O(1) + q_s \mathcal{T}_{DDH}) + (q_{pk} + q_s)(O(1) + 5\mathcal{T}_{exp}) + q_{pkr}(O(1) + 2\mathcal{T}_{exp}) + q_u(O(1) + 3\mathcal{T}_{exp} + (q_3 + q_s)(O(1) + \mathcal{T}_{DDH} + 3\mathcal{T}_{exp}))$ where $\mathcal{T}_{DDH}$ and $\mathcal{T}_{exp}$ are respectively the costs of using $\mathsf{O}^{\mathsf{DDH}}(\cdot,\cdot,\cdot,\cdot)$ to decision Diffie-Hellman problem and computing exponentiation in $\mathbb{Z}_p^*$.

Proof. Suppose that there exists an adversary $\mathcal{A}_I$ can attack our scheme. We want to build an algorithm C that runs $\mathcal{A}_I$ as a subroutine to solve GDHP. Assume that C is given $(p, q, g, g^\alpha, g^\beta | \mathsf{O}^{\mathsf{DDH}}(\cdot,\cdot,\cdot,\cdot))$ as an instance of the GDHP. And its goal is to compute $g^{\alpha\beta}$ by interacting with adversary $\mathcal{A}_I$.

Without loss of generality, we assume that any query involving an identity ID comes after a Public Key Extraction query on ID. To maintain consistency between queries made by $\mathcal{A}_I$, C keeps the following lists: L_i for $i = 1, 2, \cdots, 5$ of data for query/response pairs to random oracle H_i, L_{sk} of data for query/response pairs to Private Key Extraction oracle, L_{pk} of data for query/response pairs to Public Key Extraction oracle. Then, C randomly picks $\eta \in_R \{1, 2, \cdots, q_{pk}\}$ and runs $\mathcal{A}_I$ on input of $< p, q, n, g, y, H_1, H_2, H_3, H_4, H_5 >$ where $y = g^\alpha$ and n is the length of message to be signcrypted, and answers various oracle queries as follows:

H_1 Queries: For each query (ID, ω_{ID}), C returns the previously assigned value if it exists and a random $e_1 \in_R \mathbb{Z}_q^*$ otherwise. In the latter case, C adds $< (ID, \omega_{ID}), e_1 >$ to L_1, which is which is initially empty.

H_2 Queries: For each query $(ID, \omega_{ID}, \nu_{ID})$, C returns the previously assigned value if it exists and a random $e_2 \in_R \mathbb{Z}_q^*$ otherwise. In the latter case, C adds $< (ID, \omega_{ID}, \nu_{ID}), e_2 >$ to L_2, which is which is initially empty.

H_3 Queries: For each query $(m, k_1, k_2, \cdots, k_7)$, C proceeds as follows:

- If $< (m, k_1, k_2, \cdots, k_7), c, e_3 > \in L_3$ for some (c, e_3), return e_3.
- C go through the list L_3 with entries $< (m, k_1, \perp, k_3, \cdots, k_7), c, e_3 >$(those entries are added in answer $\mathcal{A}_I$'s signcrypt query), for some (c, e_3), such that $\mathsf{O}^{\mathsf{DDH}}(g, k_1, k_6, k_2) = 1$. If such a tuple exist, return e_3 and replace the symbol $\perp$ with k_2.
- If C reach this point of query, return a random $e_3 \in \mathbb{Z}_q^*$. Then, set $h_5 = H_5(k_2, k_3)$ and update the list L_3 with input $< (m, k_1, k_2, \cdots, k_7), c = m \oplus h_5, e_3 >$.

H_4 Queries: For each query $(m, l_1, l_2, \cdots, l_7)$, C proceeds as follows:

- If $< (m, l_1, l_2, \cdots, l_7), e_4 > \in L_4$ for some e_4, return e_4.
- C go through the list L_4 with entries $< (m, l_1, \perp, l_3, \cdots, l_7), e_4 >$(those entries are added in answer $\mathcal{A}_I$'s signcrypt query), for some e_4, such that $\mathsf{O}^{\mathsf{DDH}}(g, l_1, l_6, l_2) = 1$. If such a tuple exist, return e_4 and replace the symbol $\perp$ with l_2.
- If C reach this point of query, return a random $e_4 \in \mathbb{Z}_q^*$ and update the list L_4 with input $< (m, l_1, l_2, \cdots, l_7), e_4 >$.

H_5 Queries: For each query (u_1, u_2), C proceeds as follows:

- If $< (u_1, u_2), u_3, u_4, e_5 > \in L_5$ for some (u_3, u_4, e_5), return e_5.
- C go through the list L_5 with entries $< (\perp, u_2), u_3, u_4, e_5 >$(those entries are added in answer $\mathcal{A}_I$'s signcrypt query), for some (u_3, u_4, e_5), such that $\mathsf{O}^{\mathsf{DDH}}(g, u_3, u_4, u_1) = 1$. If such a tuple exist, return e_5 and replace the symbol $\perp$ with u_1.
- If C reach this point of query, return a random $e_5 \in \{0, 1\}^n$ and update the list L_5 with input $< (u_1, u_2), \perp, \perp, e_5 > \in L_5$.

Public Key Extraction: On the j-th non-repeat query ID_j(from this point on we denote the j-th non-repeat identity query to this oracle with ID_j), C proceeds as follows:

- If there is a tuple $< ID_j, (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j}) >$ exists in L_{pk}, return $pk_{ID_j} = (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j})$ as answer.
- Otherwise, do the following:
  - ∗ if $j \neq \eta$, pick $d_{ID_j}, \sigma_{ID_j}, z_1, z_2, s_{ID_j} \in_R \mathbb{Z}_q^*$ at random and compute $\mu_{ID_j} = g^{s_{ID_j}}$, $\omega_{ID_j} = g^{d_{ID_j}} y^{-z_1}$ and $\nu_{ID_j} = g^{\sigma_{ID_j}} y^{-z_2}$. Add $< (ID_j, \omega_{ID_j}), z_1 >$ and $< (ID_j, \omega_{ID_j}, \nu_{ID_j}), z_2 >$ to L_1 and L_2 respectively (re-choose $d_{ID_j}, \sigma_{ID_j}, z_1, z_2$ if H_1 or H_2 is already defined in the corresponding value). Return $pk_{ID_j} = (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j})$ as answer and add $< ID_j, (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j}) >$ and $< ID_j, (d_{ID_j}, s_{ID_j}) >$ to L_{pk} and L_{sk} respectively.
  - ∗ Otherwise, pick $r_{ID_\eta}, \sigma_{ID_\eta}, z_1, z_2, s_{ID_\eta} \in_R \mathbb{Z}_q^*$ at random and compute $\mu_{ID_\eta} = g^{s_{ID_\eta}}$, $\omega_{ID_\eta} = g^{r_{ID_\eta}}$ and $\nu_{ID_\eta} = g^{\sigma_{ID_\eta}} y^{-z_2}$. Add $< (ID_\eta, \omega_{ID_\eta}), z_1 >$ and

$< (ID_\eta, \omega_{ID_\eta}, \nu_{ID_\eta}), z_2 >$ to L_1 and L_2 respectively (re-choose $r_{ID_\eta}, \sigma_{ID_\eta}, z_1, z_2$ if H_1 or H_2 is already defined in the corresponding value). Return $pk_{ID_\eta} = (\mu_{ID_\eta}, \omega_{ID_\eta}, \nu_{ID_\eta}, \sigma_{ID_\eta})$ as answer and add $< ID_\eta, (\mu_{ID_\eta}, \omega_{ID_\eta}, \nu_{ID_\eta}, \sigma_{ID_\eta}) >$ and $< ID_\eta, (r_{ID_\eta}, s_{ID_\eta}) >$ to L_{pk} and L_{sk} respectively.

Partial Key Extraction: For each query ID_j, C proceeds as follows:

- If $j \neq \eta$, find $< ID_j, (d_{ID_j}, s_{ID_j}) >$ and $< ID_j, (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j}) >$ in L_{sk} and L_{pk} respectively, and return d_{ID_j} and $p_{ID_j} = (\omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j})$ as answer.
- Otherwise, C aborts the simulation.

Private Key Extraction: For each query ID_j, C proceeds as follows:

- If $j = \eta$, C aborts the simulation.
- Otherwise, find $< ID_j, (d_{ID_j}, s_{ID_j}) >$ in L_{sk} and return $sk_{ID_j} = (d_{ID_j}, s_{ID_j})$ as answer.

Public Key Replacement: For each query $< ID_j, (\mu'_{ID_j}, \omega'_{ID_j}, \nu'_{ID_j}, \sigma'_{ID_j}) >$, C finds $< ID_j, (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j}) > \in L_{pk}$, if $(\omega'_{ID_j}, \nu'_{ID_j}, \sigma'_{ID_j}) \neq (\omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j})$ satisfying $g^{\sigma'_{ID_j}} = \nu'_{ID_j} y^{H_2(ID_j, \omega'_{ID_j}, \nu'_{ID_j})}$(meaning that $\mathcal{A}_I$ produces a valid Schnorr signature on ω'_{ID_j} without master-key), C aborts the simulation. Otherwise, C sets $\mu_{ID_j} = \mu'_{ID_j}$, finds $< ID_j, (d_{ID_j}, s_{ID_j}) >$ in L_{sk} and sets $s_{ID_j} = \perp$.

Signcrypt Queries: For each query (ID_a, ID_b, m), where $a, b \in \{1, 2, \cdots, q_{pk}\}$. We observe that, if $a \neq \eta$, C knows the sender's private key $sk_{ID_a} = (d_{ID_a}, s_{ID_a})$ and can answer the query according to the specification of Signcrypt algorithm. We thus assume $a = \eta$ and hence $b \neq \eta$ by the irreflexivity assumption. C randomly secrets $s, h, h' \in_R \mathbb{Z}_q^*$ and $h_5 \in_R \{0, 1\}^n$ and sets $t = g^s \omega_{ID_a}^h y^{H_1(ID_a, \omega_{ID_a})h} \mu_{ID_a}^{h'}$ and $c = m \oplus h_5$. Observe that C knows the receiver's partial private key d_{ID_b} by construction. Then, C adds $< (m, t, \perp, t^{d_{ID_b}}, \mu_{ID_a}, \omega_{ID_a}, \mu_{ID_b}, \omega_{ID_b}), c, h >$ to L_3, $< (m, t, \perp, t^{d_{ID_b}}, \mu_{ID_a}, \omega_{ID_a}, \mu_{ID_b}, \omega_{ID_b}), h' >$ to L_4 and $< (\perp, t^{d_{ID_b}}), t, \mu_{ID_b}, h_5 >$ to L_5.(C fails if one of those hash functions is already defined in the corresponding value but this only happens with probability small than $(2q_3 + q_4 + q_5 + 3q_s)/p$), and returns ciphertext $\sigma = (c, s, t)$.

Unsigncrypt Queries: For each query $(ID_a, ID_b, \sigma = (c, s, t))$, where $a, b \in \{1, 2, \cdots, q_{pk}\}$. we assume that $b = \eta$ (and hence $a \neq \eta$ by the irreflexivity assumption), because otherwise C knows the receiver's private key sk_{ID_b} and can answer the query according to the specification of Unsigncrypt algorithm. C finds $< ID_a, (\mu_{ID_a}, \omega_{ID_a}, \nu_{ID_a}, \sigma_{ID_a}) >$ and $< ID_b, (\mu_{ID_b}, \omega_{ID_b}, \nu_{ID_b}, \sigma_{ID_b}) >$ in L_{pk}, and searches through list L_3 for entries of the form $< m_i, t, t^{s_{ID_b}}, k_{3,i}, \mu_{ID_a}, \omega_{ID_a}, \mu_{ID_b}, \omega_{ID_b}, c, e_{3,i} >$ (Here, we needn't to consider the entries in L_3 with $k_2 = \perp$, because of $a \neq \eta$) such that $\mathsf{O}^{\mathsf{DDH}}(g, t, \omega_{ID_b} y^{H_1(ID_b, \omega_{ID_b})}, k_{3,i}) = 1$ indexed by $i \in \{1, 2, \cdots, q_3 + q_s\}$, where s_{ID_b} is the secret value of the ID_b (if pk_{ID_b} has been replaced by $\mathcal{A}_I$, C gets it from $\mathcal{A}_I$). If none is found, σ is invalid, returns $\perp$. Otherwise, each one of them is further examined: for the corresponding indexes, C chicks if

$$t = g^s \omega_{ID_a}^{e_{3,i}} y^{H_1(ID_a, \omega_{ID_a}) e_{3,i}} \mu_{ID_a}^{h'} \tag{1}$$

where $h' = H_4(m_i, t, t^{s_{ID_b}}, k_{3,i}, \mu_{ID_a}, \omega_{ID_a}, \mu_{ID_b}, \omega_{ID_b})$. Note that, if (1) is satisfied, means that $\sigma = (c, s, t)$ is a valid ciphertext from ID_a to ID_b. If the unique $i \in \{1, 2, \cdots, q_3 + q_s\}$ satisfying (1) is detected, the matching m_i is returned. Overall, an inappropriate rejection occurs with probability smaller than q_u/p across the whole game.

At the end of **Find stage**, $\mathcal{A}_I$ outputs two distinct messages m_0 and m_1 (assumed of equal length), a sender identity $ID^*_{\mathbb{S}}$ and a receiver identity $ID^*_{\mathbb{R}}$, on which it wishes to be challenged. If $ID^*_{\mathbb{R}} \neq ID_\eta$, C aborts. Otherwise, it randomly picks $c^* \in_R \{0,1\}^n$ and $s^* \in_R \mathbb{Z}^*_q$, sets $t^* = g^\beta$ and $\sigma^* = (c^*, s^*, t^*)$, and sends σ^* to $\mathcal{A}_I$ as the challenge ciphertext.

At the end of **Guess stage**, $\mathcal{A}_I$ outputs its guess. Note that, $\mathcal{A}_I$ cannot recognize that is not a proper ciphertext unless it queries H_5 on $(g^{\beta s_{ID_\eta}}, g^{\beta d_{ID_\eta}})$. Along the guess stage, $\mathcal{A}_I$'s view is simulated as before and its eventual output is ignored. Standard arguments can show that a successful $\mathcal{A}_I$ is very likely to query H_5 on $(g^{\beta s_{ID_\eta}}, g^{\beta d_{ID_\eta}})$ if the simulation is indistinguishable from a real attack environment.

To produce a result C fetches a random entry $< (u_1, u_2), u_3, u_4, e_5 >$ from L_5. With probability $1/(q_3 + q_5 + q_s)$ (as L_5 contains no more than $q_3 + q_5 + q_s$ records by construction), the chosen entry will contain the right element $u_2 = g^{\beta d_{ID_\eta}} = g^{\beta(r_{ID_\eta} + \alpha e_1)}$, where $e_1 = H_1(ID_\eta, \omega_{ID_\eta})$ and r_{ID_η} can be find in L_{sk}. Then, C returns

$$g^{\alpha\beta} = \left[\frac{u_2}{(g^\beta)^{r_{ID_\eta}}}\right]^{e_1^{-1}}$$

as the solution of GDHP.

In an analysis of C's advantage, we note that it only fails in providing a consistent simulation because one of the following independent events:

E_1 : $\mathcal{A}_I$ does not choose to be challenged on ID_η.
E_2 : A Partial Key Extraction or Private Key Extraction query is made on ID_η.
E_3 : C aborts in a Public Key Replacement query because $\mathcal{A}_I$ outputs a valid Schnorr signature on ω'_{ID_η} without master-key.
E_4 : C aborts in answer $\mathcal{A}_I$'s Signcrypt query because of a collision on H_3, H_4 or H_5.
E_5 : C rejects a valid ciphertext at some point of the game.

We clearly have $Pr[\neg E_1] = 1/q_{pk}$, $Pr[\neg E_3] \geq (1-\epsilon')^{q_{pkr}}$ and we know that $\neg E_1$ implies $\neg E_2$. We also already observed that $Pr[E_4] \leq q_s(2q_3 + q_4 + q_5 + 3q_s)/2^k$ and $Pr[E_5] \leq q_u/2^k$. We thus find that

$$Pr[\neg E_1 \wedge \neg E_3 \wedge \neg E_4 \wedge \neg E_5] \geq \frac{(1-\epsilon')^{q_{pkr}}}{q_{pk}}(1 - q_s\frac{2q_3 + q_4 + q_5 + 3q_s}{2^k})(1 - \frac{q_u}{2^k}).$$

We obtain the announced bound by noting that C selects the correct element from L_5 with probability $1/(q_3 + q_5 + q_s)$.

The running time of the GDH attacker C is bound by $\mathcal{T}' < \mathcal{T} + (q_1 + q_2)O(1) + (q_3 + q_4 + q_5)(O(1) + q_s\mathcal{T}_{DDH}) + (q_{pk} + q_s)(O(1) + 5\mathcal{T}_{exp}) + q_{pkr}(O(1) + 2\mathcal{T}_{exp}) + q_u(O(1) + 3\mathcal{T}_{exp} + (q_3 + q_s)(O(1) + \mathcal{T}_{DDH} + 3\mathcal{T}_{exp}))$ where $\mathcal{T}_{DDH}$ and $\mathcal{T}_{exp}$ are respectively the costs of using $\mathsf{O}^{\mathsf{DDH}}(\cdot,\cdot,\cdot,\cdot)$ to decision Diffie-Hellman problem and computing exponentiation in $\mathbb{Z}^*_p$.

Lemma 2. Assume that an IND-CCA2-II adversary $\mathcal{A}_{II}$ has non-negligible advantage ϵ against our scheme when running in time $\mathcal{T}$, asking q_i queries to random oracles H_i ($i = 1, 2, \cdots, 5$), q_{sk} private key queries, q_{pk} public key requests, q_s signcryption queries and q_u unsigncryption queries. Then there is an algorithm C to solve the GDHP with probability

$$\epsilon' \geq \frac{\epsilon}{(q_3 + q_5 + q_s)q_{pk}}(1 - q_s\frac{2q_3 + q_4 + q_5 + 3q_s}{2^k})(1 - \frac{q_u}{2^k})$$

within a time $\mathcal{T}' < \mathcal{T} + (q_1 + q_2 + q_3 + q_4 + q_5)O(1) + q_{pk}(O(1) + 3\mathcal{T}_{exp}) + q_s(O(1) + 6\mathcal{T}_{exp}) + q_u(O(1) + 2\mathcal{T}_{exp} + (q_3 + q_s)(O(1) + \mathcal{T}_{DDH} + 3\mathcal{T}_{exp}))$ where $\mathcal{T}_{DDH}$ and $\mathcal{T}_{exp}$ denote the same quantities as in **Lemma 1**.

Proof. The proof of this lemma is very similar to the proof of **Lemma 1** and we omit the details. We just point out that for the attacker $\mathcal{A}_{II}$, C uses g^{α} to generate the public key associated with the challenge identity and sets g^{β} as a part of the challenge ciphertext. The KGC's public key is set up as g^x where C knows random $x \in_R \mathbb{Z}_q^*$. This way, C can give the master-key of the KGC to $\mathcal{A}_{II}$.

Theorem 2. Under the GDL Assumption, the proposed CLSC scheme is EUF-CMA secure in the random oracle model.

This theorem follows from Lemmas 3 and 4.

Lemma 3. Let us assume that there exists an EUF-CMA-I adversary $\mathcal{A}_I$ that makes q_i queries to random oracles H_i $(i = 1, 2, \cdots, 5)$, q_{pak} partial key queries, q_{sk} private key queries, q_{pk} public key requests, q_{pkr} public key replacement queries, q_s signcryption queries and q_u unsigncryption queries and that within time $\mathcal{T}$, $\mathcal{A}_I$ produces a forgery with probability $\epsilon \geq 10(q_s + 1)(q_s + q_3)/2^k$. Assume also that the Schnorr signature [16, 17] is $(\epsilon', \mathcal{T}, q_2, q_{pak})$-secure. Then, there is an algorithm C to solve the GDLP with probability

$$\epsilon' \geq \frac{1}{9q_{pk}}(1 - \epsilon')^{q_{pkr}}$$

in expected time $\mathcal{T}' \leq 23q_3[\mathcal{T} + (q_1 + q_2)O(1) + (q_3 + q_4 + q_5)[O(1) + q_s(\mathcal{T}_{DDH} + 2\mathcal{T}_{exp})] + q_{pk}(O(1) + 3\mathcal{T}_{exp}) + q_{pkr}(O(1) + 2\mathcal{T}_{exp}) + q_s(O(1) + 5\mathcal{T}_{exp}) + q_u(O(1) + 3\mathcal{T}_{exp} + (q_3 + q_s)(O(1) + \mathcal{T}_{DDH} + 4\mathcal{T}_{exp}))][\epsilon(1 - q_u/2^k)(1 - q_s(q_3 + q_4 + q_5 + 2q_s)/2^k)]^{-1} + \mathcal{T}_{exp}$ where $\mathcal{T}_{DDH}$ and $\mathcal{T}_{exp}$ are respectively the costs of using $\mathsf{O}^{\mathsf{rDDH}}(g, g^{\alpha}, \cdot, \cdot)$ to decision Diffie-Hellman problem and computing exponentiation in $\mathbb{Z}_p^*$.

Proof. Suppose that there exists an adversary $\mathcal{A}_I$ can attack our scheme. We want to build an algorithm C that runs $\mathcal{A}_I$ as a subroutine to solve GDLP. Assume that C gets a random instance of GDLP as follows: Given $(p, q, g, g^{\alpha}|\mathsf{O}^{\mathsf{rDDH}}(g, g^{\alpha}, \cdot, \cdot))$ for unknown $\alpha \in \mathbb{Z}_q^*$. And its goal is to compute α by interacting with adversary $\mathcal{A}_I$.

Without loss of generality, we assume that any query involving an identity ID comes after a Public Key Extraction query on ID. To maintain consistency between queries made by $\mathcal{A}_I$, C keeps the following lists: L_i for $i = 1, 2, \cdots, 5$, L_{sk} and L_{pk} as in the proof of **Lemma 1**. C randomly picks $x \in_R \mathbb{Z}_p^*$ as the master-key, computes $y = g^x$. Then, C randomly picks $\eta \in_R \{1, 2, \cdots, q_{pk}\}$ and runs $\mathcal{A}_I$ on input of $< p, q, n, g, y, H_1, H_2, H_3, H_4, H_5 >$ where n is the length of message to be signcrypted, and answers various oracle queries as follows:

H_1 Queries: For each query (ID, ω_{ID}), C returns the previously assigned value if it exists and a random $e_1 \in_R \mathbb{Z}_q^*$ otherwise. In the latter case, C adds $< (ID, \omega_{ID}), e_1 >$ to L_1, which is which is initially empty.

H_2 Queries: For each query $(ID, \omega_{ID}, \nu_{ID})$, C returns the previously assigned value if it exists and a random $e_2 \in_R \mathbb{Z}_q^*$ otherwise. In the latter case, C adds $< (ID, \omega_{ID}, \nu_{ID}), e_2 >$ to L_2, which is which is initially empty.

H_3 Queries: For each query $(m, k_1, k_2, \cdots, k_7)$, C proceeds as follows:

- If $< (m, k_1, k_2, \cdots, k_7), c, e_3, k_8 > \in L_3$ for some (c, e_3, k_8), return e_3.
- C go through the list L_3 with entries $< (m, k_1, \perp, k_3, \cdots, k_7), c, e_3, k_8 >$(those entries are added in answer $\mathcal{A}_I$'s signcrypt query), for some (c, e_3, k_8), such that $\mathsf{O}^{\mathsf{rDDH}}(g, g^\alpha, k_6, [k_2/(k_6^{k_8})]^{e_3^{-1}}) = 1$. If such a tuple exist, return e_3 and replace the symbol $\perp$ with k_2.
- If C reach this point of query, return a random $e_3 \in \mathbb{Z}_q^*$. Then, set $h_5 = H_5(k_2, k_3)$ and update the list L_3 with input $< (m, k_1, k_2, \cdots, k_7), c = m \oplus h_5, e_3, \perp >$.

H_4 Queries: For each query $(m, l_1, l_2, \cdots, l_7)$, C proceeds as follows:

- If $< (m, l_1, l_2, \cdots, l_7), e_4, l_8, l_9 > \in L_4$ for some (e_4, l_8, l_9), return e_4.
- C go through the list L_4 with entries $< (m, l_1, \perp, l_3, \cdots, l_7), e_4, l_8, l_9 >$(those entries are added in answer $\mathcal{A}_I$'s signcrypt query), for some e_4, such that $\mathsf{O}^{\mathsf{rDDH}}(g, g^\alpha, l_6, [l_2/(l_6^{l_8})]^{l_9^{-1}}) = 1$. If such a tuple exist, return e_4 and replace the symbol $\perp$ with l_2.
- If C reach this point of query, return a random $e_4 \in \mathbb{Z}_q^*$ and update the list L_4 with input $< (m, l_1, l_2, \cdots, l_7), e_4, \perp, \perp >$.

H_5 Queries: For each query (u_1, u_2), C proceeds as follows:

- If $< (u_1, u_2), u_3, e_5, u_4, u_5 > \in L_5$ for some (u_3, e_5, u_4, u_5), return e_5.
- C go through the list L_5 with entries $< (\perp, u_2), u_3, e_5, u_4, u_5 >$(those entries are added in answer $\mathcal{A}_I$'s signcrypt query), for some (u_3, e_5, u_4, u_5), such that $\mathsf{O}^{\mathsf{rDDH}}(g, g^\alpha, u_3, [u_1/(u_3^{u_4})]^{u_5^{-1}}) = 1$. If such a tuple exist, return e_5 and replace the symbol $\perp$ with u_1.
- If C reach this point of query, return a random $e_5 \in_R \{0,1\}^n$ and update the list L_5 with input $< (u_1, u_2), \perp, e_5, \perp, \perp > \in L_5$.

Public Key Extraction: On the j-th non-repeat query ID_j(from this point on we denote the j-th non-repeat identity query to this oracle with ID_j), C proceeds as follows:

- If there is a tuple $< ID_j, (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j}) >$ exists in L_{pk}, return $pk_{ID_j} = (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j})$ as answer.
- Otherwise, do the following:
  - ∗ if $j \neq \eta$, pick $r_{ID_j}, r'_{ID_j}, z_1, z_2, s_{ID_j} \in_R \mathbb{Z}_q^*$ at random and compute $\mu_{ID_j} = g^{s_{ID_j}}$, $\omega_{ID_j} = g^{r_{ID_j}}$, $d_{ID_j} = r_{ID_j} + xz_1$, $\nu_{ID_j} = g^{r'_{ID_j}}$ and $\sigma_{ID_j} = r'_{ID_j} + xz_2$. Add $< (ID_j, \omega_{ID_j}), z_1 >$ and $< (ID_j, \omega_{ID_j}, \nu_{ID_j}), z_2 >$ to L_1 and L_2 respectively (re-choose $r_{ID_j}, r'_{ID_j}, z_1, z_2$ if H_1 or H_2 is already defined in the corresponding value). Return $pk_{ID_j} = (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j})$ as answer and add $< ID_j, (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j}) >$ and $< ID_j, (d_{ID_j}, s_{ID_j}) >$ to L_{pk} and L_{sk} respectively.
  - ∗ Otherwise, pick $r'_{ID_\eta}, z_2, s_{ID_\eta} \in_R \mathbb{Z}_q^*$ at random and compute $\mu_{ID_\eta} = g^{s_{ID_\eta}}$, $\nu_{ID_\eta} = g^{r'_{ID_\eta}}$ and $\sigma_{ID_\eta} = r'_{ID_\eta} + xz_2$. Add $< (ID_\eta, g^\alpha, \nu_{ID_\eta}), z_2 >$ to L_2 (re-choose r'_{ID_η}, z_2 if

H_2 is already defined in the corresponding value). Return $pk_{ID_\eta} = (\mu_{ID_\eta}, \omega_{ID_\eta} = g^\alpha, \nu_{ID_\eta}, \sigma_{ID_\eta})$ as answer and add $< ID_\eta, (\mu_{ID_\eta}, \omega_{ID_\eta}, \nu_{ID_\eta}, \sigma_{ID_\eta}) >$ and $< ID_\eta, (\perp, s_{ID_\eta}) >$ to L_{pk} and L_{sk} respectively.

Partial Key Extraction: For each query ID_j, C proceeds as follows:

- If $j \neq \eta$, find $< ID_j, (d_{ID_j}, s_{ID_j}) >$ and $< ID_j, (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j}) >$ in L_{sk} and L_{pk} respectively, and return d_{ID_j} and $p_{ID_j} = (\omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j})$ as answer.
- Otherwise, C aborts the simulation.

Private Key Extraction: For each query ID_j, C proceeds as follows:

- If $j = \eta$, C aborts the simulation.
- Otherwise, find $< ID_j, (d_{ID_j}, s_{ID_j}) >$ in L_{sk} and return $sk_{ID_j} = (d_{ID_j}, s_{ID_j})$ as answer.

Public Key Replacement: For each query $< ID_j, (\mu'_{ID_j}, \omega'_{ID_j}, \nu'_{ID_j}, \sigma'_{ID_j}) >$, C finds $< ID_j, (\mu_{ID_j}, \omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j}) > \in L_{pk}$, if $(\omega'_{ID_j}, \nu'_{ID_j}, \sigma'_{ID_j}) \neq (\omega_{ID_j}, \nu_{ID_j}, \sigma_{ID_j})$ satisfying $g^{\sigma'_{ID_j}} = \nu'_{ID_j} y^{H_2(ID_j, \omega'_{ID_j}, \nu'_{ID_j})}$(meaning that $\mathcal{A}_I$ produces a valid Schnorr signature on ω'_{ID_j} without master-key), C aborts the simulation. Otherwise, C sets $\mu_{ID_j} = \mu'_{ID_j}$, finds $< ID_j, (d_{ID_j}, s_{ID_j}) >$ in L_{sk} and sets $s_{ID_j} = \perp$.

Signcrypt Queries: For each query (ID_a, ID_b, m), where $a, b \in \{1, 2, \cdots, q_{pk}\}$. We observe that, if $a \neq \eta$, C knows the sender's private key $sk_{ID_a} = (d_{ID_a}, s_{ID_a})$ and can answer the query according to the specification of Signcrypt algorithm. We thus assume $a = \eta$ and hence $b \neq \eta$ by the irreflexivity assumption. C randomly secrets $s, h, h' \in_R \mathbb{Z}_q^*$ and $h_5 \in_R \{0, 1\}^n$ and sets $t = g^s \omega_{ID_a}^h y^{h_1 h} \mu_{ID_a}^{h'} = g^{s + x h_1 h + s_{ID_a} h'} g^{\alpha h}$ and $c = m \oplus h_5$, where $h_1 = H_1(ID_a, \omega_{ID_a})$ and s_{ID_a} is the secret value of the ID_a (if pk_{ID_a} has been replaced by $\mathcal{A}_I$, C gets it from $\mathcal{A}_I$). Observe that C knows the receiver's partial private key d_{ID_b} by construction. Then, C adds $< (m, t, \perp, t^{d_{ID_b}}, \mu_{ID_a}, \omega_{ID_a}, \mu_{ID_b}, \omega_{ID_b}), c, h, s + x h_1 h + s_{ID_a} h' >$ to L_3, $< (m, t, \perp, t^{d_{ID_b}}, \mu_{ID_a}, \omega_{ID_a}, \mu_{ID_b}, \omega_{ID_b}), h', s + x h_1 h + s_{ID_a} h', h >$ to L_4 and $< (\perp, t^{d_{ID_b}}), \mu_{ID_b}, h_5, s + x h_1 h + s_{ID_a} h', h >$ to L_5, and returns ciphertext $\sigma = (c, s, t)$.

Unsigncrypt Queries: For each query $(ID_a, ID_b, \sigma = (c, s, t))$, where $a, b \in \{1, 2, \cdots, q_{pk}\}$. we assume that $b = \eta$ (and hence $a \neq \eta$ by the irreflexivity assumption), because otherwise C knows the receiver's private key sk_{ID_b} and can answer the query according to the specification of Unsigncrypt algorithm. C finds $< ID_a, (\mu_{ID_a}, \omega_{ID_a}, \nu_{ID_a}, \sigma_{ID_a}) >$ and $< ID_b, (\mu_{ID_b}, \omega_{ID_b}, \nu_{ID_b}, \sigma_{ID_b}) >$ in L_{pk}, and searches through list L_3 for entries of the form $< (m_i, t, t^{s_{ID_b}}, k_{3,i}, \mu_{ID_a}, \omega_{ID_a}, \mu_{ID_b}, \omega_{ID_b}), c, e_{3,i}, k_8 >$ (Here, we needn't to consider the entries in L_3 with $k_2 = \perp$, because of $a \neq \eta$) such that $\mathsf{O}^{\mathsf{rDDH}}(g, g^\alpha, t, k_{3,i}/t^{x h_1}) = 1$ indexed by $i \in \{1, 2, \cdots, q_3 + q_s\}$, where $h_1 = H_1(ID_b, \omega_{ID_b})$ and s_{ID_b} is the secret value of the ID_b (if pk_{ID_b} has been replaced by $\mathcal{A}_I$, C gets it from $\mathcal{A}_I$). If none is found, σ is invalid, returns $\perp$. Otherwise, each one of them is further examined: for the corresponding indexes, C chicks if

$$t = g^s \omega_{ID_a}^{e_{3,i}} y^{H_1(ID_a, \omega_{ID_a}) e_{3,i}} \mu_{ID_a}^{h'} \tag{2}$$

where $h' = H_4(m_i, t, t^{s_{ID_b}}, k_{3,i}, \mu_{ID_a}, \omega_{ID_a}, \mu_{ID_b}, \omega_{ID_b})$. Note that, if (2) is satisfied, means that $\sigma = (c, s, t)$ is a valid ciphertext from ID_a to ID_b. If the unique $i \in \{1, 2, \cdots, q_3 + q_s\}$ satisfying (2) is detected, the matching m_i is returned. Overall, an inappropriate rejection occurs with probability smaller than q_u/p across the whole game.

Eventually, $\mathcal{A}_I$ outputs a valid ciphertext $\sigma = (c, s, t)$ from $ID^*_{\mathbb{S}}$ to $ID^*_{\mathbb{R}}$. If $ID^*_{\mathbb{S}} \neq ID_\eta$, C aborts. Otherwise, having the knowledge of $sk_{ID^*_{\mathbb{R}}}$, C computes $h_3 = H_3(m^*, t, t^{s_{ID^*_{\mathbb{R}}}}, t^{d_{ID^*_{\mathbb{R}}}}, \mu_{ID_\eta}, \omega_{ID_\eta}, \mu_{ID^*_{\mathbb{R}}}, \omega_{ID^*_{\mathbb{R}}})$, where $m^* = \mathsf{Unsigncrypt}(\mathsf{ID}_\eta, \mathsf{pk}_{\mathsf{ID}_\eta}, \mathsf{sk}_{\mathsf{ID}^*_{\mathbb{R}}}, \sigma)$ (For simplicity, we denote $\sigma = (c, s, t, h_3)$ as $\mathcal{A}_I$'s outputs). Then, using the oracle replay technique [15], C generates one more valid ciphertext from $\sigma = (c, s, t, h_3)$ which is named as $\sigma' = (c, s', t, h'_3)$. This is achieved by running the turing machine again with the same random tape but with the different hash value.

Since $\sigma = (c, s, t, h_3)$ and $\sigma' = (c, s', t, h'_3)$ are both valid ciphertext for the same message m^* and randomness r, we obtain the relations

$$g^s \omega_{ID_\eta}^{h_3} y^{H_1(ID_\eta, \omega_{ID_\eta}) h_3} \mu_{ID_\eta}^{h_4} = t = g^{s'} \omega_{ID_\eta}^{h'_3} y^{H_1(ID_\eta, \omega_{ID_\eta}) h'_3} \mu_{ID_\eta}^{h_4},$$

where $h_4 = H_4(m^*, t, t^{s_{ID^*_{\mathbb{R}}}}, t^{d_{ID^*_{\mathbb{R}}}}, \mu_{ID_\eta}, \omega_{ID_\eta}, \mu_{ID^*_{\mathbb{R}}}, \omega_{ID^*_{\mathbb{R}}})$. Then,we have

$$\begin{aligned}
g^s \omega_{ID_\eta}^{h_3} y^{H_1(ID_\eta, \omega_{ID_\eta}) h_3} \mu_{ID_\eta}^{h_4} &= g^{s'} \omega_{ID_\eta}^{h'_3} y^{H_1(ID_\eta, \omega_{ID_\eta}) h'_3} \mu_{ID_\eta}^{h_4}, \\
g^{s-s'} y^{H_1(ID_\eta, \omega_{ID_\eta})(h_3 - h'_3)} &= \omega_{ID_\eta}^{h'_3 - h_3}, \\
g^{s-s'} g^{x H_1(ID_\eta, \omega_{ID_\eta})(h_3 - h'_3)} &= g^{\alpha(h'_3 - h_3)}, \\
g^{[s-s' + x H_1(ID_\eta, \omega_{ID_\eta})(h_3 - h'_3)](h'_3 - h_3)^{-1}} &= g^\alpha.
\end{aligned}$$

Hence, C can compute $\alpha = [s - s' + xH_1(ID_\eta, \omega_{ID_\eta})(h_3 - h'_3)](h'_3 - h_3)^{-1}$ as the solution of GDLP.

In an analysis of C's advantage, we note that it only fails because one of the following independent events:

E_1 : $\mathcal{A}_I$ does not choose to be challenged on ID_η.
E_2 : A Partial Key Extraction or Private Key Extraction query is made on ID_η.
E_3 : C aborts in a Public Key Replacement query because $\mathcal{A}_I$ outputs a valid Schnorr signature on ω'_{ID_η} without master-key.
E_4 : C fails in using the oracle replay technique [15] to generate one more valid ciphertext.

We clearly have $Pr[\neg E_1] = 1/q_{pk}$, $Pr[\neg E_3] \geq (1 - \epsilon')^{q_{pkr}}$ and we know that $\neg E_1$ implies $\neg E_2$. From **Lemma 12** in [15], we know that $Pr[\neg E_4] \geq 1/9$. We obtain the announced bound by noting that

$$Pr[\neg E_1 \wedge \neg E_3 \wedge \neg E_4] \geq \frac{1}{9 q_{pk}} (1 - \epsilon')^{q_{pkr}}.$$

From the proof of **Lemma 12** in [15], we know that the total running time $\mathcal{T}'$ of solving the GDLP with probability $\epsilon' \geq \frac{1}{9q_{pk}}(1 - \epsilon')^{q_{pkr}}$ is bound by $23q_3[\mathcal{T} + (q_1 + q_2)O(1) + (q_3 + q_4 + q_5)[O(1) + q_s(\mathcal{T}_{DDH} + 2\mathcal{T}_{exp})] + q_{pk}(O(1) + 3\mathcal{T}_{exp}) + q_{pkr}(O(1) + 2\mathcal{T}_{exp}) + q_s(O(1) + 5\mathcal{T}_{exp}) + q_u(O(1) + 3\mathcal{T}_{exp} + (q_3 + q_s)(O(1) + \mathcal{T}_{DDH} + 4\mathcal{T}_{exp}))][\epsilon(1 - q_u/2^k)(1 - q_s(q_3 + q_4 + q_5 + 2q_s)/2^k)]^{-1} + \mathcal{T}_{exp}$ where the last term accounts for the cost of compute the system public key in the preparation phase. Thus, this completes the proof.

Lemma 4. Assume that there exists an EUF-CMA-II adversary $\mathcal{A}_{II}$ that makes q_i queries to random oracles H_i $(i = 1, 2, \cdots, 5)$, q_{sk} private key queries, q_{pk} public key requests, q_s

signcryption queries and q_u unsigncryption queries and that within time $\mathcal{T}$, $\mathcal{A}_{II}$ produces a forgery with probability $\epsilon \geq 10(q_s + 1)(q_s + q_3)/2^k$. Then, there is an algorithm C to solve the GDLP with probability

$$\epsilon' \geq \frac{1}{9q_{pk}}$$

in expected time $\mathcal{T}' \leq 23q_3[\mathcal{T} + (q_1 + q_2 + q_3 + q_4 + q_5)O(1) + q_{pk}(O(1) + 3\mathcal{T}_{exp}) + q_s(O(1) + 6\mathcal{T}_{exp}) + q_u(O(1) + 3\mathcal{T}_{exp} + (q_3 + q_s)(O(1) + \mathcal{T}_{DDH} + 4\mathcal{T}_{exp}))][\epsilon(1 - q_u/2^k)(1 - q_s(q_3 + q_4 + q_5 + 2q_s)/2^k)]^{-1} + \mathcal{T}_{exp}$ where $\mathcal{T}_{DDH}$ and $\mathcal{T}_{exp}$ denote the same quantities as in **Lemma 3**.

Proof. The proof of this lemma is very similar to the proof of **Lemma 3** and we omit the details.

6 Conclusion

Certificateless public key cryptography is receiving significant attention because it is a new paradigm that simplifies the traditional PKC and solves the inherent key escrow problem suffered by ID-PKC. Certificateless signcryption is one of the most important security primitives in CL-PKC. In this paper, we present a pairing-free certificateless signcryption scheme. The security of our scheme is based on the hardness assumptions of GDHP and GDLP.

References

[1] S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. In *Advances in Cryptology-ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 452–473. Springer-Verlag, 2003.

[2] M.H. Au, Y. Mu, J. Chen, D.S. Wong, J.K. Liu, and G. Yang. Malicious kgc attacks in certificateless cryptography. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*. ACM, 2007. 302-311.

[3] J. Baek, R. Safavi-Naini, and W. Susilo. Certificateless public key encryption without pairing. In *Information Security*, volume 3650 of *LNCS*, pages 134–148. Springer-Verlag, 2005.

[4] J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. *Journal of Cryptology*, 20(2):203–235, 2007.

[5] M. Barbosa and P. Farshim. Certificateless signcryption. Cryptology ePrint Archive: Report 2008/143, Available from: http://eprint.iacr.org/2008/143.

[6] P.S.L.M. Barreto, B. Lynn, and M. Scott. Efficient implementation of pairing-based cryptosystems. *Journal of Cryptology*, 17(4):321–334, 2004.

[7] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[8] I.F. Blake, V.K. Murty, and G. Xu. Refinements of miller's algorithm for computing the weil/tate pairing. *Journal of Algorithms*, 58(2):134–149, 2006.

[9] X. Boyen. Multipurpose identity-based signcryption: A swiss army knife for identity-based cryptography. In *Advances in Cryptology-CRYPTO 2003*, volume 2729 of *LNCS*, pages 383–399. Springer-Verlag, 2003.

[10] L. Chen and J. Malone-Lee. Improved identity-based signcryption. In *Public Key Cryptography-PKC 2005*, volume 3386 of *LNCS*, pages 362–379. Springer-Verlag, 2005.

[11] I. Duursma and H. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In *Advances in Cryptology-ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 111–123. Springer-Verlag, 2003.

[12] S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the tate pairing. In *Algorithmic Number Theory*, volume 2369 of *LNCS*, pages 69–86. Springer-Verlag, 2002.

[13] S. Goldwasser, S. Micali, and R.L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

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

[15] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[16] C.P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology-CRYPTO'89 Proceedings*, volume 435 of *LNCS*, pages 239–252. Springer-Verlag, 1990.

[17] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[18] S.S.D. Selvi, S.S. Vivek, and C.P. Ragan. On the security of certificateless signcryption schemes. Cryptology ePrint Archive: Report 2009/298, Available from: http://eprint.iacr.org/2009/298.

[19] S.S.D. Selvi, S.S. Vivek, and C.P. Ragan. Security weaknesses in two certificateless signcryption schemes. Cryptology ePrint Archive: Report 2010/092, Available from: http://eprint.iacr.org/2010/092.

[20] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1985.

[21] Y. Sun, F. Zhang, and J. Baek. Strongly secure certificateless public key encryption without pairing. In *Cryptology and Network Security*, volume 4856 of *LNCS*, pages 194–208. Springer-Verlag, 2007.

[22] B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology-EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer-Verlag, 2005.

[23] C. Wu and Z. Chen. A new efficient certificateless signcryption scheme. In *International Symposium on Information Science and Engieering, 2008. ISISE'08.*, volume 1, pages 661–664, 2008.

[24] W. Xie and Z. Zhang. Efficient and provably-secure certificateless signcryption from bilinear maps. Cryptology ePrint Archive: Report 2009/578, Available from: http://eprint.iacr.org/2009/578.