Efficient and Provably Secure Certificateless Signcryption from Bilinear Maps*

Wenjian Xie† Zhang Zhang

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

Abstract. Signcryption is a cryptographic primitive that fulfills both the functions of digital signature and public key encryption simultaneously, at a cost significantly lower than that required by the traditional signature-then-encryption approach. In 2008, Barbosa and Farshim introduced the notion of certificateless signcryption (CLSC) and proposed the first CLSC scheme [2], but which requires six pairing operations in the signcrypt and unsigncrypt phases. In this paper, aimed at designing an efficient CLSC scheme, we propose a new efficient CLSC scheme from bilinear maps, which requires only two pairing operations in the signcrypt and unsigncrypt phases and is more efficient than all the schemes available.
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 [21] 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

*Recently Sharmila Deva Selvi *et al*. pointed out that our scheme are not secure against Type I adversary [20]. We have to re-formalize a new security model which is a little weaker than the original one. In this model, we assume that the ciphertext from the signcryption oracle is valid under the original public key of the sender.

†Corresponding author (W. Xie). E-mail: wjxieem@gmail.com.

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.

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 a certificateless cryptosystem, the KGC is involved to issue a partial private key d_{ID} for a user with identity ID. Then the user independently generates his public/private key pair (pk_{ID}, sk_{ID}) use d_{ID} and a secret value x_{ID} chosen by himself, and publishes pk_{ID}.

In 2008, Barbosa and Farshim introduced the notion of certificateless signcryption (CLSC) and proposed the first CLSC scheme [2], which requires six pairing operations in the signcrypt and unsigncrypt phases. And aimed at designing an efficient CLSC scheme, Wu and Chen proposed a new CLSC scheme [23], which requires four pairing operations in the signcrypt and unsigncrypt phases, but unfortunately it was found insecure by Sharmila *et al*. [19]. We note that in pairing based cryptosystems, although numerous papers discuss the complexity of pairings and how to speed up the pairing computation [5, 8, 14, 15], the computation of the pairing still remains time-consuming.

Our Contribution. It is fair to say that devising an efficient certificateless signcryption still remains an important problem. In this paper, motivated by identity-based signcryption scheme proposed in [4] and certificateless public key encryption scheme [13], we present a new efficient certificateless pairing-based signcryption scheme, which requires only two pairing operations in the signcrypt and unsigncrypt phases. Observations [22] pinpointed problems arising when many provably secure pairing based protocols are implemented using asymmetric pairings and ordinary curves. Our proposal avoids those problems thanks to the fact that it does not require to hash onto an elliptic curve cyclic subgroup.

Organization. The rest of this paper is organized as follows: In next Section, we describe some preliminaries, including bilinear map groups, our complexity assumptions 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 and efficiency analysis. Finally, we conclude this paper in Section 6.

2 Preliminaries

2.1 Bilinear Map Groups

Let k be a security parameter and p be a k-bit prime number. Let us consider groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of the same prime order p and P, Q be generators of respectively $\mathbb{G}_1$ and $\mathbb{G}_2$. We say that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are bilinear map groups if there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow$

$\mathbb{G}_T$ satisfying the following properties:

– Bilinearity: $\forall(S,T) \in \mathbb{G}_1 \times \mathbb{G}_2$, $\forall a,b \in \mathbb{Z}_p^*$, $e(aS,bT) = e(S,T)^{ab}$.
– Non-degeneracy: $\forall S \in \mathbb{G}_1$, $e(S,T) = 1$ for all $T \in \mathbb{G}_2$ iff $S = O$.
– Computability: $\forall(S,T) \in \mathbb{G}_1 \times \mathbb{G}_2$, $e(S,T)$ is efficiently computable.
– There exists an efficient, publicly computable (but not necessarily invertible) isomorphism ψ: $\mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(Q) = P$.

Such bilinear map (called bilinear pairing) groups are known to be instantiable with ordinary elliptic curves such as those suggested in [17] or [6]. In this case, the trace map can be used as an efficient isomorphism ψ as long as $\mathbb{G}_2$ is properly chosen [22]. With supersingular curves, symmetric pairings (i.e. $\mathbb{G}_1 = \mathbb{G}_2$) can be obtained and ψ is the identity.

2.2 Related Complexity Assumptions

Definition 1. The Discrete Logarithm Problem (DLP) in $\mathbb{G}_2$ is, given $(Q,\alpha Q) \in \mathbb{G}_2^2$ for unknown $\alpha \in \mathbb{Z}_p^*$, to compute α.

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the DLP in $\mathbb{G}_2$ is defined as

$$Adv_{\mathcal{A}}^{DLP} = Pr[\mathcal{A}(Q,\alpha Q) = \alpha | \alpha \in \mathbb{Z}_p^*].$$

The DL assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{DLP}$ is negligible.

Definition 2. The Computational Diffie-Hellman Problem (CDHP) in $\mathbb{G}_2$ is, given $(Q,\alpha Q,\beta Q) \in \mathbb{G}_2^3$ for unknown $\alpha,\beta \in \mathbb{Z}_p^*$, to compute $\alpha\beta Q$.

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the CDHP in $\mathbb{G}_2$ is defined as

$$Adv_{\mathcal{A}}^{CDHP} = Pr[\mathcal{A}(Q,\alpha Q,\beta Q) = \alpha\beta Q | \alpha,\beta \in \mathbb{Z}_p^*].$$

The CDH assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{CDHP}$ is negligible.

Definition 3. The q-Strong Diffie-Hellman Problem (q-SDHP) [10] in the groups $(\mathbb{G}_1, \mathbb{G}_2)$ consists of, given a $(q+2)$-tuple $(P, Q, \alpha Q, \alpha^2 Q, \ldots, \alpha^q Q) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ for unknown $\alpha \in \mathbb{Z}_p^*$, finding a pair $(c, \frac{1}{\alpha+c}P)$ with $c \in \mathbb{Z}_p^*$.

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the q-SDHP in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as

$$Adv_{\mathcal{A}}^{q\text{-}SDHP} = Pr[\mathcal{A}(P, Q, \alpha Q, \alpha^2 Q, \ldots, \alpha^q Q) = (c, \frac{1}{\alpha+c}P) | \alpha, c \in \mathbb{Z}_p^*].$$

The q-SDH assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{q\text{-}SDHP}$ is negligible.

Definition 4. The q-Bilinear Diffie-Hellman Inversion Problem (q-BDHIP) [9] in the groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathbb{T})$ consists of, given a $(q+2)$-tuple $(P, Q, \alpha Q, \alpha^2 Q, \ldots, \alpha^q Q) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ for unknown $\alpha \in \mathbb{Z}_p^*$, computing $e(P,Q)^{1/\alpha} \in \mathbb{G}_T$.

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the q-BDHIP in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ is defined as

$$Adv_{\mathcal{A}}^{q\text{-}BDHIP} = Pr[\mathcal{A}(P, Q, \alpha Q, \alpha^2 Q, \ldots, \alpha^q Q) = e(P, Q)^{1/\alpha} | \alpha \in \mathbb{Z}_p^*].$$

The q-BDHI assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{q\text{-}BDHIP}$ is negligible.

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

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

Set-Public-Key: Taking params and a user's identity ID and his secret value x_{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 x_{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 $\delta = \mathsf{Unsigncrypt}(\mathsf{ID}_{\mathbb{S}}, \mathsf{pk}_{\mathsf{ID}_{\mathbb{S}}}, \mathsf{sk}_{\mathsf{ID}_{\mathbb{R}}}, \sigma)$, where δ is the message m or the symbol $\perp$.

params, as an implied inputs to Signcrypt and Unsigncrypt algorithms, is omitted. The Setup and Partial-Private-Key-Extract algorithms are performed by KGC. Once a partial private key d_{ID} is given to a 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 [2], 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 [11, 12] 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 [2, 11, 12], 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 Private Key Extraction: $\mathcal{A}_I$ is able to ask for the partial private key d_{ID} for any ID. C computes the partial private key d_{ID} corresponding to the identity ID and returns d_{ID} 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 pick a new secret value x'_{ID} and compute the new public pk'_{ID} corresponding to the new secret value x'_{ID}, and then replace pk_{ID} with pk'_{ID}.
- 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 $\mathcal{A}_I$. Here, the ciphertext σ from the signcryption oracle is valid under the original public key of the sender $ID_{\mathbb{S}}$. This assumption due to the fact that in the real world, $\mathcal{A}_I$ may be able to gain $ID_{\mathbb{S}}$'s some valid ciphertext from eavesdropping or the intended

receivers, but those ciphertext are generated by $ID_{\mathbb{S}}$ using his own secret value and partial private key.

– 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 private key extraction and private key extraction on $ID^*_{\mathbb{R}}$. C picks randomly a bit $\beta \in \{0, 1\}$, computes σ^*=Signcrypt($\mathsf{sk}_{\mathsf{ID}^*_{\mathbb{S}}}$, $\mathsf{ID}^*_{\mathbb{R}}$, $\mathsf{pk}_{\mathsf{ID}^*_{\mathbb{R}}}$, m_β) 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 private 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 β' and wins the game if β=β'.

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

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 keys of any identities by itself with the master-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 $\beta \in \{0, 1\}$, computes σ^*=Signcrypt($\mathsf{sk}_{\mathsf{ID}^*_{\mathbb{S}}}$, $\mathsf{ID}^*_{\mathbb{R}}$, $\mathsf{pk}_{\mathsf{ID}^*_{\mathbb{R}}}$, m_β) 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 β' and wins the game if β=β'.

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

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 schemes 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 $\mathsf{master\text{-}key}$ 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 private key and the private key of $ID^*_{\mathbb{S}}$ was not extract and wins the game if the result of $\mathsf{Unsigncrypt}(\mathsf{ID}^*_{\mathbb{S}},\ \mathsf{pk}_{\mathsf{ID}^*_{\mathbb{S}}},\ \mathsf{sk}_{\mathsf{ID}^*_{\mathbb{R}}},\ \sigma^*)$ 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 $\mathsf{master\text{-}key}$ and params, and sends params and $\mathsf{master\text{-}key}$ 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 $\mathsf{Unsigncrypt}(\mathsf{ID}^*_{\mathbb{S}},\ \mathsf{pk}_{\mathsf{ID}^*_{\mathbb{S}}},\ \mathsf{sk}_{\mathsf{ID}^*_{\mathbb{R}}},\ \sigma^*)$ 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 New efficient CLSC scheme

In this section, we propose a new efficient CLSC scheme which consists of the following seven algorithms.

Setup: Given a security parameter k, the algorithm works as follows:

- Outputs descriptions of bilinear map groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of same prime order $p > 2^k$.
- Chooses an arbitrary generator Q of $\mathbb{G}_2$ and sets $P = \psi(Q) \in \mathbb{G}_1$ and $g = e(P, Q) \in \mathbb{G}_T$.
- Randomly picks $s \in \mathbb{Z}_p^*$ and sets $P_{pub} = sQ$ as, respectively, master-key and system public key.
- Selects three distinct cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, $H_2 : \{0,1\}^n \times \mathbb{G}_2 \times \mathbb{G}_T \times \mathbb{G}_2^3 \rightarrow \mathbb{Z}_p^*$ and $H_3 : \mathbb{G}_T \times \mathbb{G}_2 \rightarrow \{0,1\}^n$ where n is the length of message to be signcrypted.
- The system parameters are params $=< \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, P, Q, g, P_{pub}, \psi, H_1, H_2, H_3 >$ and publishes params.

Partial-Private-Key-Extract: Given params, master-key and an identity $ID \in \{0,1\}^*$, this algorithm works as follows: Compute $Q_{ID} = H_1(ID) \in \mathbb{Z}_p^*$ and $d_{ID} = \frac{1}{s+Q_{ID}}P$ and sends d_{ID} to the user with identity ID as his partial private key via a secure channel. The user can check its correctness by checking whether $e(d_{ID}, P_{pub} + Q_{ID}Q) = g$. For convenience, we define $T_{ID} = P_{pub} + H_1(ID)Q$.

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

Set-Public-Key: Given params, a user's identity ID and the secret value x_{ID}, this algorithm computes his public key $pk_{ID} = x_{ID}(P_{pub} + H_1(ID)Q)$.

Set-Private-Key: Given params, the user's partial private key d_{ID} and his secret value $x_{ID} \in \mathbb{Z}_p^*$, and output a pair (d_{ID}, x_{ID}) as the user's private key sk_{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:

- Randomly picks $r_1 \in_R \mathbb{Z}_p^*$ and computes $u = r_1(P_{pub} + H_1(B)Q)$ and $c = m \oplus H_3(g^{r_1}, r_1 pk_B)$.
- Computes $h_2 = H_2(m, u, g^{r_1}, r_1 pk_B, pk_A, pk_B)$, $v = \frac{r_1+h_2}{r_1} d_A$ and $w = x_A h_2 + r_1$.
- Sets ciphertext $\sigma = (c, u, v, w)$.

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

- Computes $g^{r_1'} = e(d_B, u)$ and $m = c \oplus H_3(g^{r_1'}, x_B u)$.
- Sets $h_2 = H_2(m, u, g^{r_1'}, x_B u, pk_A, pk_B)$ and $r_1' T_A = wT_A - h_2 pk_A$.
- Accept m if and only if $e(v, r_1' T_A) = g^{r_1'} g^{h_2}$ hold, return $\perp$ otherwise.

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

$$g^{r_1'} = e(d_B, u) = e(\frac{1}{s + Q_B}P, r_1(sQ + Q_B Q)) = g^{r_1},$$

$$x_B u = x_B r_1(sQ + Q_B Q) = r_1 x_B(sQ + Q_B Q) = r_1 pk_B$$

and

$$e(v, wT_A - h_2 pk_A) = e(v, r_1 T_A) = e(\frac{r_1 + h_2}{r_1}\frac{1}{s + Q_A}P, r_1(sQ + Q_A Q)) = g^{r_1} g^{h_2}.$$

5 Analysis of the Proposed Scheme

5.1 Proof of security

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

Theorem 1. Under the q-BDHI assumption and CDH assumption, the proposed CLSC scheme is IND-CCA2 secure in the random oracle model.

This theorem follows from Lemmas 1 and 2.

Lemma 1. Assume that 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 ($i = 1, 2, 3$), q_{ppk} partial private key queries, q_{sk} private key queries, q_{pk} public key requests, q_{pkr} public key replacement queries, q_s signcryption queries and q_{un} unsigncryption queries. Then there is an algorithm C to solve the q-BDHIP for $q = q_1 + 1$ with probability

$$\epsilon' \geq \frac{\epsilon}{q_1(q_2 + q_3 + q_s)}(1 - q_s\frac{2q_2 + q_3 + 2q_s}{2^k})(1 - \frac{q_{un}}{2^k})$$

within a time $\mathcal{T}' < \mathcal{T} + [3q_1 + q_{pk} + 4q_s + 3q_u(q_2 + q_s)]\mathcal{T}_{mult} + q_u(q_2 + q_s)\mathcal{T}_{exp} + [2q_2 q_s + 2q_3 q_s + q_s + 3q_u(q_2 + q_s)]\mathcal{T}_p$ where $\mathcal{T}_{mult}$ denotes the time for computing multiplication in $\mathbb{G}_1$ or $\mathbb{G}_2$, $\mathcal{T}_{exp}$ denotes the time for computing exponentiation in $\mathbb{G}_T$ and $\mathcal{T}_p$ is the time for pairing computation.

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 q-BDHIP. Assume that C gets a random instance of q-BDHIP in as follows: Given $(P, Q, \alpha Q, \alpha^2 Q, \ldots, \alpha^q Q) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ for unknown $\alpha \in \mathbb{Z}_p^*$. And its goal is to compute $e(P, Q)^{1/\alpha} \in \mathbb{G}_T$ by interacting with adversary $\mathcal{A}_I$. In the preparation phase, C randomly picks $\omega_0, \omega_1, \cdots, \omega_{q-1} \in_R \mathbb{Z}_p^*$. As in the technique of [4], it builds a generator $G \in \mathbb{G}_1$ such that it knows $q - 1$ pairs $(\omega_i + \omega_0, \frac{1}{\alpha+\omega_i}G)$ for $i \in \{1, 2, \cdots, q - 1\}$. To do so,

- It expands $f(z) = \prod_{i=1}^{q-1}(z + \omega_i)$ to obtain $c_0, c_1, \cdots, c_{q-1} \in \mathbb{Z}_p^*$ so that $f(z) = \sum_{i=0}^{q-1} c_i z^i$.
- It sets generator $H = \sum_{i=0}^{q-1} c_i(\alpha^i Q) = f(\alpha)Q \in \mathbb{G}_2$ and $G = \psi(H) = f(\alpha)P \in \mathbb{G}_1$. The system public key is fixed to $P_{pub} = \sum_{i=1}^{q} c_{i-1}(\alpha^i Q) - \omega_0(\sum_{i=0}^{q-1} c_i(\alpha^i Q))$ so that $P_{pub} = (\alpha - \omega_0)H$ although C does not know $\alpha - \omega_0$.

– For $1 \le i \le q-1$, C expands $f_i(z) = f(z)/(z+\omega_i) = \sum_{j=0}^{q-2} d_j z^j$ and

$$\sum_{j=0}^{q-2} d_j \psi(\alpha^j Q) = f_i(\alpha)P = \frac{f(\alpha)}{\alpha+\omega_i}P = \frac{1}{\alpha+\omega_i}G. \tag{1}$$

The pairs $(\omega_i + \omega_0, \frac{1}{\alpha+\omega_i}G)$ are computed using the left member of (1).

Throughout the game, we assume that H_1 Queries are distinct, that the target identity $ID^*_{\mathbb{R}}$ is submitted to H_1 Queries at some point and that any query involving an identity ID comes after a H_1 Queries on ID. To maintain consistency between queries made by $\mathcal{A}_I$, C keeps the following lists: L_i for $i = 1, 2, 3$ of data for query/response pairs to random oracle H_i; L_{pk} of data for query/response pairs to Public Key Extraction oracle. Then, C randomly picks $\mu \in_R \{1, 2, \cdots, q_1\}$ and runs $\mathcal{A}_I$ on input of $< \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathbb{T}}, e, p, G, H, g, P_{pub}, \psi >$ where $g = e(G, H)$, and answers various oracle queries as follows:

H_1 Queries: 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), if $j \ne \mu$, C sets $Q_{ID_j} = \omega_j + \omega_0$. It then adds $< ID_j, Q_{ID_j}, \frac{1}{\alpha+\omega_j}G >$ to the list L_1 which is initially empty and returns Q_{ID_j}. Otherwise, it returns $Q_{ID_\mu} = \omega_0$ and adds $< ID_\mu, Q_{ID_\mu}, \perp>$ to L_1.

H_2 Queries: For each query $(m, S_1, S_2, S_3, pk_{ID_{\mathbb{S}}}, pk_{ID_{\mathbb{R}}})$, C proceeds as follows:

– If $< m, S_1, S_2, S_3, pk_{ID_{\mathbb{S}}}, pk_{ID_{\mathbb{R}}}, S_4, h_2, c >\in L_2$ for some (S_4, h_2, c), returns h_2.
– C goes through the list L_2 with entries $< m, S_1, S_2, \perp, pk_{ID_{\mathbb{S}}}, pk_{ID_{\mathbb{R}}}, S_4, h_2, c >$(those entries are added in answer $\mathcal{A}_I$'s signcrypt query), for some (S_4, h_2, c), such that $e(\psi(S_1), pk_{ID_{\mathbb{R}}}) = e(\psi(S_4), S_3)$. If such a tuple exists, it returns h_2 and replaces the symbol $\perp$ with S_3.
– If C reaches this point of execution, it returns a random $h_2 \in_R \mathbb{Z}_p^*$. Then, sets $h_3 = H_3(S_2, S_3) \in \{0,1\}^n$ and updates L_2 with input $< m, S_1, S_2, S_3, pk_{ID_{\mathbb{S}}}, pk_{ID_{\mathbb{R}}}, \perp, h_2, c = m \oplus h_3 >$.

H_3 Queries: For each query (S_2, S_3), C proceeds as follows:

– If $< S_2, S_3, h_3, S_1, S_4, S_5 >\in L_3$ for some (h_3, S_1, S_4, S_5), returns h_3.
– C goes through the list L_3 with entries $< S_2, \perp, h_3, S_1, S_4, S_5 >$(those entries are added in answer $\mathcal{A}_I$'s signcrypt query), for some (h_3, S_1, S_4, S_5), such that $e(\psi(S_1), S_5) = e(\psi(S_4), S_3)$. If such a tuple exists, it returns h_3 and replaces the symbol $\perp$ with S_3.
– If C reaches this point of execution, it returns a random $h_3 \in_R \{0,1\}^n$ and updates the list L_3 with input $< S_2, S_3, h_3, \perp, \perp, \perp>$.

Partial Private Key Extraction: For each new query ID_j, if $j = \mu$, then C fails. Otherwise, finds $< ID_j, Q_{ID_j}, \frac{1}{\alpha+\omega_j}G >$ in L_1 and returns $d_{ID_j} = \frac{1}{\alpha+\omega_j}G$.

Public Key Extraction: For each query ID_j, C checks in list L_{pk}, which is initially empty, if $< ID_j, x_{ID_j}, pk_{ID_j} >\in L_{pk}$ for some pk_{ID_j}. If so, returns pk_{ID_j}. Otherwise, C picks $x_{ID_j} \in_R \mathbb{Z}_p^*$ at random, sets $pk_{ID_j} = x_{ID_j}(P_{pub} + H_1(ID_j)H)$, then returns pk_{ID_j} and adds $< ID_j, x_{ID_j}, pk_{ID_j} >$ in L_{pk}.

Private Key Extraction: For each new query ID_j, if $j = \mu$, then C aborts the simulation.

Otherwise finds $< ID_j, Q_{ID_j}, \frac{1}{\alpha+\omega_j}G >$ and $< ID_j, x_{ID_j}, pk_{ID_j} >$ in L_1 and L_{pk}, respectively, and returns $sk_{ID_j} = (\frac{1}{\alpha+\omega_j}G, x_{ID_j})$.

Public Key Replacement: For each query $< ID_j, pk'_{ID_j} >$, C finds $< ID_j, x_{ID_j}, pk_{ID_j} >$ in L_{pk} and sets $pk_{ID_j} = pk'_{ID_j}$. If L_{pk} does not contain $< ID_j, x_{ID_j}, pk_{ID_j} >$, C queries Public Key Extraction oracle with ID_j first.

Signcrypt Queries: For each query (ID_a, ID_b, m), where $a, b \in \{1, 2, \cdots, q_1\}$. We observe that, if $a \neq \mu$, C knows the sender's private key $sk_{ID_a} = (\frac{1}{\alpha+\omega_a}G, x_{ID_a})$ and can answer the query according to the specification of Signcrypt algorithm. We thus assume $a = \mu$ and hence $b \neq \mu$ by the irreflexivity assumption. Observe that C knows the receiver's partial private key $d_{ID_b} = \frac{1}{\alpha+\omega_b}G$ by construction. The difficulty is to find a random triple $(c, u, v, w, h_2) \in \{0, 1\}^n \times \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ for which

$$e(v, wT_{ID_a} - h_2 pk_{ID_a}) = e(d_{ID_b}, u)g^{h_2}$$

where $T_{ID_a} = P_{pub} + H_1(ID_a)H$. To do so, C randomly secrets $r_1, r_2, r_3 \in_R \mathbb{Z}_p^*$ and $h_3 \in_R \{0, 1\}^n$ and sets $v = r_1 d_{ID_b}$, $w = r_2$, $h_2 = r_3$, $u = r_1 r_2 T_{ID_a} - r_1 r_3 pk_{ID_a} - r_3 T_{ID_b}$ and $c = m \oplus h_3$, where $pk_{ID_a} = x_{ID_a}(P_{pub} + H_1(ID_a)H)$ is the original public key of ID_a generated by Public Key Extraction oracle. Then, C sets $S_2 = e(d_{ID_b}, u)$, adds $< S_2, \perp, h_3, u, T_{ID_b}, pk_{ID_b} >$ and $< m, u, S_2, \perp, pk_{ID_a}, pk_{ID_b}, T_{ID_b}, h_2, c >$ to L_3 and L_2 respectively (C fails if H_2 or H_3 is already defined in the corresponding value but this only happens with probability small than $(2q_2 + q_3 + 2q_s)/p$) , and returns ciphertext $\sigma = (c, u, v, w)$.

Unsigncrypt Queries: For each query $(ID_a, ID_b, \sigma = (c, u, v, w))$, where $a, b \in \{1, 2, \cdots, q_1\}$. we assume that $b = \mu$ (and hence $a \neq \mu$ by the irreflexivity assumption), because otherwise C knows the receiver's private key $sk_{ID_b} = (\frac{1}{\alpha+\omega_b}G, x_{ID_b})$ and can answer the query according to the specification of Unsigncrypt algorithm. Note that, for all valid ciphertexts $\sigma = (c, u, v, w)$, $\log_{T_{ID_b}} u = \log_{T_{ID_a}}(wT_{ID_a} - h_2 pk_{ID_a})$, where $h_2 = H_2(m, u, g^{r_1}, x_{ID_b}u, pk_{ID_a}, pk_{ID_b})$ is the hash value obtained in the Signcrypt algorithm. Hence, we have the relation

$$e(\psi(T_{ID_a}), u) = e(\psi(T_{ID_b}), wT_{ID_a} - h_2 pk_{ID_a}).$$

C finds $< ID_a, x_{ID_a}, pk_{ID_a} >$ and $< ID_b, x_{ID_b}, pk_{ID_b} >$ in L_{pk}, and searches through list L_2 for entries of the form $< m_k, u, S_{2,k}, x_{ID_b}u, pk_{ID_a}, pk_{ID_b}, S_{4,k}, h_{2,k}, c >$ (Here, we needn't to consider the entries in L_2 with $S_3 = \perp$, because of $a \neq \mu$) indexed by $k \in \{1, 2, \cdots, q_2 + q_s\}$, where x_{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

$$e(\psi(T_{ID_a}), u) = e(\psi(T_{ID_b}), wT_{ID_a} - h_{2,k} pk_{ID_a}). \tag{2}$$

Note that, if (2) is satisfied, means that $h_{2,k}$ is the correct hash value obtained in the Signcrypt algorithm. And after gets the correct hash value $h_{2,k}$, C tests if

$$e(v, wT_{ID_a} - h_{2,k} pk_{ID_a}) = S_{2,k} g^{h_{2,k}}, \tag{3}$$

meaning that $S_{2,k} = g^{r_1} \in \mathbb{G}_T$ is the correct random value used in the Unsigncrypt algorithm. If the unique $k \in \{1, 2, \cdots, q_2 + q_s\}$ satisfying (2) and (3) is detected, the matching m_k

is returned. Overall, an inappropriate rejection occurs with probability smaller than q_{un}/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 $ID^*_{\mathbb{R}} \neq ID_\mu$, C aborts. Otherwise, it randomly picks $c \in_R \{0,1\}^n$, $r_1, w \in_R \mathbb{Z}^*_p$ and $v \in_R \mathbb{G}^*_1$, sets $u = r_1 H$ and $\sigma^* = (c, u, v, w)$, 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_3 on $(e(d_{ID_\mu}, u), x_{ID_\mu} u)$. 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_3 on $(e(d_{ID_\mu}, u), x_{ID_\mu} u)$ if the simulation is indistinguishable from a real attack environment.

To produce a result C fetches a random entry $< S_2, S_3, h_3, S_1, S_4, S_5 >$ from L_3. With probability $1/(q_2 + q_3 + q_s)$ (as L_3 contains no more than $q_2 + q_3 + q_s$ records by construction), the chosen entry will contain the right element $S_2 = e(d_{ID_\mu}, u) = e(\frac{1}{\alpha}G, r_1 H) = e(G, H)^{r_1/\alpha} = e(f(\alpha)P, f(\alpha)Q)^{r_1/\alpha}$, where $f(z) = \sum_{i=0}^{q-1} c_i z^i$ is the polynomial for which $H = f(\alpha)Q$. Then, q-BDHIP solution can be extracted by noting that, if $\gamma^* = e(P, Q)^{1/\alpha}$, then

$$e(G, H)^{1/\alpha} = \gamma^{*(c_0^2)} e(c_0 P, \sum_{i=0}^{q-2} c_{i+1}(\alpha^i Q)) e(\sum_{i=0}^{q-2} c_{i+1}\psi(\alpha^i Q), H).$$

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_μ.
E_2 : A Partial Private Key Extraction or Private Key Extraction query is made on ID_μ.
E_3 : C aborts in answer $\mathcal{A}_I$'s signcrypt query because of collision on H_2 or H_3.
E_4 : C rejects a valid ciphertext at some point of the game.

We clearly have $Pr[\neg E_1] = 1/q_1$ and we know that $\neg E_1$ implies $\neg E_2$. We also already observed that $Pr[E_3] \leq q_s(2q_2 + q_3 + 2q_s)/2^k$ and $Pr[E_4] \leq q_{un}/2^k$. We thus find that

$$Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4] \geq \frac{1}{q_1}(1 - q_s\frac{2q_2 + q_3 + 2q_s}{2^k})(1 - \frac{q_{un}}{2^k}).$$

We obtain the announced bound by noting that C selects the correct element from L_3 with probability $1/(q_2 + q_3 + q_s)$. The running time of the q-BDHI attacker C is bound by $\mathcal{T}' < \mathcal{T} + [3q_1 + q_{pk} + 4q_s + 3q_u(q_2 + q_s)]\mathcal{T}_{mult} + q_u(q_2 + q_s)\mathcal{T}_{exp} + [2q_2 q_s + 2q_3 q_s + q_s + 3q_u(q_2 + q_s)]\mathcal{T}_p$.

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 $(i = 1, 2, 3)$, q_{sk} private key queries, q_{pk} public key requests, q_s signcrypt queries and q_{un} unsigncrypt queries. Then there is an algorithm C to solve the CDHP with probability

$$\epsilon' \geq \frac{\epsilon}{q_1(q_2 + q_3 + q_s)}(1 - q_s\frac{2q_2 + q_3 + 2q_s}{2^k})(1 - \frac{q_{un}}{2^k})$$

within a time $\mathcal{T}' < \mathcal{T} + [1 + q_{pk} + q_{sk} + 5q_s + 2q_u(q_2 + q_s)]\mathcal{T}_{mult} + q_u(q_2 + q_s)\mathcal{T}_{exp} + [q_s + 7q_u(q_2 + q_s)]\mathcal{T}_p$ where $\mathcal{T}_{mult}$, $\mathcal{T}_{exp}$ and $\mathcal{T}_p$ denote the same quantities as in **Lemma 1**.

Proof. Suppose that there exists an adversary $\mathcal{A}_{II}$ can attack our scheme. We want to build an algorithm C that runs $\mathcal{A}_{II}$ as a subroutine to solve CDHP. Assume that C gets a random instance of CDHP as follows: Given $(Q, \alpha Q, \beta Q) \in \mathbb{G}_2^3$ for unknown $\alpha, \beta \in \mathbb{Z}_p^*$. And its goal is to compute $\alpha\beta Q$ by interacting with adversary $\mathcal{A}_{II}$.

Throughout the game, we assume that H_1 Queries are distinct, that the target identity $ID^*_{\mathbb{R}}$ is submitted to H_1 Queries at some point and that any query involving an identity ID comes after a H_1 Queries on ID. To maintain consistency between queries made by $\mathcal{A}_{II}$, C keeps the lists: L_i for $i = 1, 2, 3$ and L_{pk} as in the proof of **Lemma 1**. C randomly picks $s \in_R \mathbb{Z}_p^*$ as the master-key, computes $P_{pub} = sQ$, and sends $< \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, P, Q, g, P_{pub}, \psi >$ and the master-key s to $\mathcal{A}_{II}$. Then, C randomly picks $\mu \in_R \{1, 2, \cdots, q_1\}$ and answers various oracle queries as follows:

H_1 Queries: 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 randomly picks $Q_{ID_j} \in_R \mathbb{Z}_p^*$ and adds $< ID_j, Q_{ID_j} >$ to L_1, which is initially empty, then returns Q_{ID_j}.

H_2 Queries: For each query $(m, S_1, S_2, S_3, pk_{ID_{\mathbb{S}}}, pk_{ID_{\mathbb{R}}})$, C proceeds as follows:

– If $< m, S_1, S_2, S_3, pk_{ID_{\mathbb{S}}}, pk_{ID_{\mathbb{R}}}, h_2, c > \in L_2$ for some (h_2, c), returns h_2.
– Otherwise, C returns a random $h_2 \in_R \mathbb{Z}_p^*$. Then, sets $h_3 = H_3(S_2, S_3) \in \{0, 1\}^n$ and updates L_2 with input $< m, S_1, S_2, S_3, pk_{ID_{\mathbb{S}}}, pk_{ID_{\mathbb{R}}}, h_2, c = m \oplus h_3 >$.

H_3 Queries: For each query (S_2, S_3), C returns the previously assigned value if it exists and a random $h_3 \in_R \{0, 1\}^n$ otherwise. In the latter case, C adds $< S_2, S_3, h_3 >$ to L_3, which is which is initially empty.

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

– If $< ID_j, x_{ID_j}, pk_{ID_j} > \in L_{pk}$ for some pk_{ID_j}, returns pk_{ID_j}.
– Else, if $j = \mu$, C returns $pk_{ID_j} = (s + H_1(ID_j))\alpha Q$ and adds $< ID_j, \perp, pk_{ID_j} >$ in L_{pk}.
– Else C picks $x_{ID_j} \in_R \mathbb{Z}_p^*$ at random, sets $pk_{ID_j} = x_{ID_j}(P_{pub} + H_1(ID_j)Q)$, then returns pk_{ID_j} and adds $< ID_j, x_{ID_j}, pk_{ID_j} >$ in L_{pk}.

Private Key Extraction: For each new query ID_j, if $j = \mu$, then C aborts the simulation. Otherwise finds $< ID_j, x_{ID_j}, pk_{ID_j} >$ in L_{pk}, and returns $sk_{ID_j} = (\frac{1}{s+H_1(ID_j)}P, x_{ID_j})$.

Signcrypt Queries: For each query (ID_a, ID_b, m), where $a, b \in \{1, 2, \cdots, q_1\}$. We observe that, if $a \neq \mu$, C knows the sender's private key $sk_{ID_a} = (\frac{1}{s+Q_{ID_a}}P, x_{ID_a})$ and can answer the query according to the specification of Signcrypt algorithm, we thus assume $a = \mu$. Observe that C knows the receiver's private key $sk_{ID_b} = (\frac{1}{s+Q_{ID_b}}P, x_{ID_b})$ by construction. The difficulty is to find a random triple $(c, u, v, w, h_2) \in \{0, 1\}^n \times \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ for which

$$e(v, wT_{ID_a} - h_2 pk_{ID_a}) = e(u, d_{ID_b})g^{h_2}$$

where $T_{ID_a} = P_{pub} + H_1(ID_a)Q$. To do so, C randomly secrets $r_1, r_2, r_3 \in_R \mathbb{Z}_p^*$ and $h_3 \in_R \{0, 1\}^n$ and sets $v = r_1 d_{ID_b}$, $w = r_2$, $h_2 = r_3$, $u = r_1 r_2 T_{ID_a} - r_1 r_3 pk_{ID_a} - r_3 T_{ID_b}$ and $c = m \oplus h_3$. Then, C computes $S_2 = e(d_{ID_b}, u)$ and $S_3 = x_{ID_b} u$, adds $< S_2, S_3, h_3 >$ and $< m, u, S_2, S_3, pk_{ID_a}, pk_{ID_b}, h_2, c >$ to L_3 and L_2 respectively (C fails if H_2 or H_3 is already defined in the corresponding value but this only happens with probability small than $(2q_2 + q_3 + 2q_s)/p$), and returns ciphertext $\sigma = (c, u, v, w)$.

Unsigncrypt Queries: For each query $(ID_a, ID_b, \sigma = (c, u, v, w))$, where $a, b \in \{1, 2, \cdots, q_1\}$. we assume that $b = \mu$, because otherwise C knows the receiver's private key $sk_{ID_b} = (\frac{1}{s+Q_{ID_b}}P, x_{ID_b})$ and can answer the query according to the specification of Unsigncrypt algorithm. Note that, for all valid ciphertexts $\sigma = (c, u, v, w)$, $\log_{T_{ID_b}} u = \log_{T_{ID_a}}(wT_{ID_a} - h_2 pk_{ID_a})$, where $h_2 = H_2(m, u, g^{r_1}, x_{ID_b}u, pk_{ID_a}, pk_{ID_b})$ is the hash value obtained in the Signcrypt algorithm. Hence, we have the relation

$$e(\psi(T_{ID_a}), u) = e(\psi(T_{ID_b}), wT_{ID_a} - h_2 pk_{ID_a}).$$

C searches through list L_2 for entries of the form $< m_k, u, e(\frac{1}{s+H_1(ID_b)}P, u), S_{3,k}, pk_{ID_a}, pk_{ID_b}, h_{2,k}, c >$ such that $e(\psi(u), pk_{ID_b}) = e(\psi(T_{ID_b}), S_{3,k})$, indexed by $k \in \{1, 2, \cdots, q_2 + q_s\}$, If none is found, σ is invalid, returns $\perp$. Otherwise, each one of them is further examined: for the corresponding indexes, C chicks if

$$e(\psi(T_{ID_a}), u) = e(\psi(T_{ID_b}), wT_{ID_a} - h_{2,k} pk_{ID_a}). \tag{4}$$

Note that, if (4) is satisfied, means that $h_{2,k}$ is the correct hash value obtained in the Signcrypt algorithm. And after gets the correct hash value $h_{2,k}$, C tests if

$$e(v, wT_{ID_a} - h_{2,k} pk_{ID_a}) = e(\frac{1}{s + H_1(ID_b)}P, u) g^{h_{2,k}}, \tag{5}$$

meaning that $\sigma = (c, u, v, w)$ is a valid ciphertext from ID_a to ID_b. If the unique $k \in \{1, 2, \cdots, q_2 + q_s\}$ satisfying (4) and (5) is detected, the matching m_k is returned. Overall, an inappropriate rejection occurs with probability smaller than q_{un}/p across the whole game.

At the end of Find stage, $\mathcal{A}_{II}$ 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 $ID^*_{\mathbb{R}} \neq ID_\mu$, C aborts. Otherwise, it randomly picks $c \in_R \{0, 1\}^n$, $w \in_R \mathbb{Z}^*_p$ and $v \in_R G^*_1$, sets $u = (s + H_1(ID_\mu))\beta Q$ and $\sigma^* = (c, u, v, w)$, and sends σ^* to $\mathcal{A}_{II}$ as the challenge ciphertext.

At the end of Guess stage, $\mathcal{A}_{II}$ outputs its guess. Note that, $\mathcal{A}_{II}$ cannot recognize that is not a proper ciphertext unless it queries H_3 on $(e(d_{ID_\mu}, u), x_{ID_\mu}u)$. Along the guess stage, $\mathcal{A}_{II}$'s view is simulated as before and its eventual output is ignored. Standard arguments can show that a successful $\mathcal{A}_{II}$ is very likely to query H_3 on $(e(d_{ID_\mu}, u), x_{ID_\mu}u)$ if the simulation is indistinguishable from a real attack environment.

To produce a result C fetches a random entry $< S_2, S_3, h_3 >$ from L_3. With probability $1/(q_2 + q_3 + q_s)$ (as L_3 contains no more than $q_2 + q_3 + q_s$ records by construction), the chosen entry will contain the right element $S_3 = x_{ID_\mu}u = (s + H_1(ID_\mu))\alpha\beta Q$. Then, C returns $\alpha\beta Q = (s + H_1(ID_\mu))^{-1} S_3$ as the solution of CDHP.

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 : $\mathcal{A}_{II}$ does not choose to be challenged on ID_μ.

E_2 : A Private Key Extraction query is made on ID_μ.
E_3 : C aborts in answer $\mathcal{A}_{II}$'s signcrypt query because of collision on H_2 or H_3.
E_4 : C rejects a valid ciphertext at some point of the game.

We clearly have $Pr[\neg E_1] = 1/q_1$ and we know that $\neg E_1$ implies $\neg E_2$. We also already observed that $Pr[E_3] \leq q_s(2q_2 + q_3 + 2q_s)/2^k$ and $Pr[E_4] \leq q_{un}/2^k$. We thus find that

$$Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4] \geq \frac{1}{q_1}(1 - q_s\frac{2q_2 + q_3 + 2q_s}{2^k})(1 - \frac{q_{un}}{2^k}).$$

We obtain the announced bound by noting that C selects the correct element from L_3 with probability $1/(q_2 + q_3 + q_s)$. The running time of the CDH attacker C is bound by $\mathcal{T}' < \mathcal{T} + [1 + q_{pk} + q_{sk} + 5q_s + 2q_u(q_2 + q_s)]\mathcal{T}_{mult} + q_u(q_2 + q_s)\mathcal{T}_{exp} + [q_s + 7q_u(q_2 + q_s)]\mathcal{T}_p$.

Theorem 2. Under the q-SDH assumption and DL assumption, the proposed CLSC scheme is EUF-CMA secure in the random oracle model.

This theorem follows from Lemmas 3 and 4.

Lemma 3. Assume that there exists an EUF-CMA-I adversary $\mathcal{A}_I$ that makes q_i queries to random oracles H_i $(i = 1, 2, 3)$, q_{ppk} partial key queries, q_{sk} private key queries, q_{pk} public key requests, q_{pkr} public key replacement queries, q_s signcrypt queries and q_{un} unsigncrypt queries. Assume also that, within a time $\mathcal{T}$, $\mathcal{A}_I$ produces a forgery with probability $\epsilon \geq 10(q_s + 1)(q_s + q_2)/2^k$. Then, there is an algorithm C to solve the q-SDHP for $q = q_1 + 1$ with probability

$$\epsilon' \geq \frac{1}{9q_1}(1 - \frac{q-1}{2^k})$$

in expected time $\mathcal{T}' \leq 23q_2[\mathcal{T} + (q_{pk} + 4q_s + 3q_u(q_2 + q_s))\mathcal{T}_{mult} + q_u(q_2 + q_s)\mathcal{T}_{exp} + (2q_2q_s + 2q_3q_s + q_s + 3q_u(q_2 + q_s))\mathcal{T}_p][\epsilon(1 - q_u/2^k)(1 - q_s(q_2 + q_3 + q_s)/2^k)]^{-1} + 3q_1\mathcal{T}_{mult}$ where $\mathcal{T}_{mult}$, $\mathcal{T}_{exp}$ and $\mathcal{T}_p$ denote the same quantities as in **Lemma 1**.

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 q-SDHP. Assume that C gets a random instance of q-SDHP as follows: Given $(P, Q, \alpha Q, \alpha^2 Q, \ldots, \alpha^q Q) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ for unknown $\alpha \in \mathbb{Z}_p^*$. And its goal is to find a pair $(c, \frac{1}{c+\alpha}P)$ by interacting with adversary $\mathcal{A}_I$. In the preparation phase, as in the technique of [4], C builds a generator $G \in \mathbb{G}_1$ such that it knows $q - 1$ pairs $(\omega_i, \frac{1}{\alpha+\omega_i}G)$ for $i \in \{1, 2, \cdots, q-1\}$. To do so,

- It randomly picks $\omega_1, \omega_2, \cdots, \omega_{q-1} \in_R \mathbb{Z}_p^*$ and expands $f(z) = \prod_{i=1}^{q-1}(z + \omega_i)$ to obtain $c_0, c_1, \cdots, c_{q-1} \in \mathbb{Z}_p^*$ so that $f(z) = \sum_{i=0}^{q-1} c_i z^i$.
- It sets generators $H = \sum_{i=0}^{q-1} c_i(\alpha^i Q) = f(\alpha)Q \in \mathbb{G}_2$ and $G = \psi(H) = f(\alpha)P \in \mathbb{G}_1$. The system public key is fixed to $P_{pub} = \sum_{i=1}^{q} c_{i-1}(\alpha^i Q)$ so that $P_{pub} = \alpha H$ although C does not know α.
- For $1 \leq i \leq q-1$, C expands $f_i(z) = f(z)/(z + \omega_i) = \sum_{j=0}^{q-2} d_j z^j$ and

$$\sum_{j=0}^{q-2} d_j \psi(\alpha^j Q) = f_i(\alpha)P = \frac{f(\alpha)}{\alpha + \omega_i}P = \frac{1}{\alpha + \omega_i}G. \qquad (6)$$

The pairs $(\omega_i, \frac{1}{\alpha+\omega_i}G)$ are computed using the left member of (6).

Throughout the game, we assume that H_1 Queries are distinct, that the target identity $ID^*_{\mathbb{S}}$ is submitted to H_1 Queries at some point and that any query involving an identity ID comes after a H_1 Queries on ID. To maintain consistency between queries made by $\mathcal{A}_{II}$, C keeps the lists: L_i for $i = 1, 2, 3$ and L_{pk} as in the proof of **Lemma 1**. Then, C randomly picks $\mu \in_R \{1, 2, \cdots, q_1\}$ and runs $\mathcal{A}_I$ on input of $< \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, G, H, g, P_{pub}, \psi >$ where $g = e(G, H)$, and answers various oracle queries as follows:

H_1 Queries: 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), if $j \neq \mu$, C sets $Q_{ID_j} = \omega_j$. It then adds $< ID_j, Q_{ID_j}, \frac{1}{\omega_j+\alpha}G >$ to the list L_1 which is initially empty and returns Q_{ID_j}. Otherwise, it returns a random $\omega^* \in_R \mathbb{Z}^*_p$ and adds $< ID_\mu, Q_{ID_\mu} = \omega^*, \perp>$ to L_1.

$\mathcal{A}_I$'s queries to other oracle (except H_1 Queries oracle) are answered as in the proof of **Lemma 1**.

Eventually, $\mathcal{A}_I$ outputs a valid ciphertext $\sigma = (c, u, v, w)$ from $ID^*_{\mathbb{S}}$ to $ID^*_{\mathbb{R}}$. If $ID^*_{\mathbb{S}} \neq ID_\mu$, C aborts. Otherwise, having the knowledge of $sk_{ID^*_{\mathbb{R}}}$, C computes $h_2 = H_2(m^*, u, e(d_{ID^*_{\mathbb{R}}}, u), x_{ID^*_{\mathbb{R}}}u, pk_{ID^*_{\mathbb{S}}}, pk_{ID^*_{\mathbb{R}}})$, where $m^* = \mathsf{Unsigncrypt}(\mathsf{ID}^*_{\mathbb{S}}, \mathsf{pk}_{\mathsf{ID}^*_{\mathbb{S}}}, \mathsf{sk}_{\mathsf{ID}^*_{\mathbb{R}}}, \sigma)$ (For simplicity, we denote $\sigma = (c, u, v, w, h_2)$ as $\mathcal{A}_I$'s outputs). Then, using the oracle replay technique [18], C generates one more valid ciphertext from $\sigma = (c, u, v, w, h_2)$ which is named as $\sigma' = (c, u, v', w', h'_2)$. This is achieved by running the turing machine again with the same random tape but with the different hash value.

Since $\sigma = (c, u, v, w, h_2)$ and $\sigma' = (c, u, v', w', h'_2)$ are both valid ciphertext for the same message m^* and randomness r_1, we obtain the relations

$$w'T_{ID^*_{\mathbb{S}}} - h'_2 pk_{ID^*_{\mathbb{S}}} = r_1 T_{ID^*_{\mathbb{S}}} = wT_{ID^*_{\mathbb{S}}} - h_2 pk_{ID^*_{\mathbb{S}}}.$$

Then,we have

$$(w' - w)T_{ID^*_{\mathbb{S}}} = (h'_2 - h_2)pk_{ID^*_{\mathbb{S}}}.$$

Hence, C can compute $x_{ID^*_{\mathbb{S}}} = (w' - w)/(h'_2 - h_2)$. From the specification of $\mathsf{Unsigncrypt}$ algorithm, we know that

$$e(v, wT_{ID^*_{\mathbb{S}}} - h_2 pk_{ID^*_{\mathbb{S}}}) = g^{r_1} g^{h_2} = e((r_1 + h_2)G, H)$$

and

$$e(v', w'T_{ID^*_{\mathbb{S}}} - h'_2 pk_{ID^*_{\mathbb{S}}}) = g^{r_1} g^{h'_2} = e((r_1 + h'_2)G, H).$$

Hence, we have

$$\begin{aligned}
e(v, wT_{ID^*_{\mathbb{S}}})e(v, -h_2 pk_{ID^*_{\mathbb{S}}})e(-h_2 G, H) &= e(v', w'T_{ID^*_{\mathbb{S}}})e(v', -h'_2 pk_{ID^*_{\mathbb{S}}})e(-h'_2 G, H), \\
e(wv, T_{ID^*_{\mathbb{S}}})e(-h_2 v, pk_{ID^*_{\mathbb{S}}})e(-h_2 G, H) &= e(w'v', T_{ID^*_{\mathbb{S}}})e(-h'_2 v', pk_{ID^*_{\mathbb{S}}})e(-h'_2 G, H), \\
e(wv - w'v', T_{ID^*_{\mathbb{S}}})e(h'_2 v' - h_2 v, pk_{ID^*_{\mathbb{S}}}) &= e((h_2 - h'_2)G, H), \\
e(wv - w'v', T_{ID^*_{\mathbb{S}}})e(x_{ID^*_{\mathbb{S}}}(h'_2 v' - h_2 v), T_{ID^*_{\mathbb{S}}}) &= e((h_2 - h'_2)G, H),
\end{aligned}$$

and

$$e((h_2 - h'_2)^{-1}[wv - w'v' + x_{ID^*_{\mathbb{S}}}(h'_2 v' - h_2 v)], T_{ID^*_{\mathbb{S}}}) = e(G, H).$$

Hence, C can compute $\frac{1}{\alpha+\omega^*}G = (h_2 - h_2')^{-1}[wv - w'v' + (w' - w)(h_2'v' - h_2v)/(h_2' - h_2)]$. From $\frac{1}{\alpha+\omega^*}G$, C can proceed as in [4] to extract $\frac{1}{\alpha+\omega^*}P$: it first obtains $\gamma_{-1}, \gamma_0, \cdots, \gamma_{q-2} \in \mathbb{Z}_p^*$ for which $f(z)/(z+\omega^*) = \gamma_{-1}/(z+\omega^*) + \sum_{i=0}^{q-2} \gamma_i z^i$ and eventually computes

$$\frac{1}{\alpha+\omega^*}P = \frac{1}{\gamma_{-1}}\left[\frac{1}{\alpha+\omega^*}G - \sum_{i=0}^{q-2}\gamma_i\psi(\alpha^i Q)\right]$$

before returning the pair $(\omega^*, \frac{1}{\alpha+\omega^*}P)$ as the solution of q-SDHP.

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_μ.
E_2 : A Partial Private Key Extraction or Private Key Extraction query is made on ID_μ.
E_3 : C fails in using the oracle replay technique [18] to generate one more valid ciphertext.
E_4 : $\omega^* = \omega_i$ for $i \in \{1, 2, \cdots, q-1\}$.

We clearly have $Pr[\neg E_1] = 1/q_1$ and $Pr[E_4] \le (q-1)/2^k$ and we know that $\neg E_1$ implies $\neg E_2$. From **Lemma 12** in [18], we know that $Pr[\neg E_3] \ge 1/9$. We obtain the announced bound by noting that

$$Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4] \ge \frac{1}{9q_1}(1 - \frac{q-1}{2^k}).$$

From the proof of **Lemma 12** in [18], we know that the total running time $\mathcal{T}'$ of solving the q-SDHP with probability $\epsilon' \ge \frac{1}{9q_1}(1 - \frac{q-1}{2^k})$ is bound by $23q_2[\mathcal{T} + (q_{pk} + 4q_s + 3q_u(q_2 + q_s))\mathcal{T}_{mult} + q_u(q_2+q_s)\mathcal{T}_{exp} + (2q_2q_s + 2q_3q_s + q_s + 3q_u(q_2+q_s))\mathcal{T}_p][\epsilon(1 - q_u/2^k)(1 - q_s(q_2 + q_3 + q_s)/2^k)]^{-1} + 3q_1\mathcal{T}_{mult}$ where the last term accounts for the cost of 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 $(i = 1, 2, 3)$, q_{sk} private key queries, q_{pk} public key requests, q_s signcrypt queries and q_{un} unsigncrypt queries. Assume also that, within a time $\mathcal{T}$, $\mathcal{A}_{II}$ produces a forgery with probability $\epsilon \ge 10(q_s+1)(q_s+q_2)/2^k$. Then, there is an algorithm C to solve the DLP with probability

$$\epsilon' \ge \frac{1}{9q_1}$$

in expected time $\mathcal{T}' \le 23q_2[\mathcal{T} + (q_{pk} + q_{sk} + 5q_s + 2q_u(q_2+q_s))\mathcal{T}_{mult} + q_u(q_2+q_s)\mathcal{T}_{exp} + (q_s + 7q_u(q_2+q_s))\mathcal{T}_p][\epsilon(1-q_u/2^k)(1-q_s(q_2+q_3+q_s)/2^k)]^{-1} + \mathcal{T}_{mult}$ where $\mathcal{T}_{mult}$, $\mathcal{T}_{exp}$ and $\mathcal{T}_p$ denote the same quantities as in **Lemma 1**

Proof. Suppose that there exists an adversary $\mathcal{A}_{II}$ can attack our scheme. We want to build an algorithm C that runs $\mathcal{A}_{II}$ as a subroutine to solve DLP. Assume that C gets a random instance of DLP as follows: Given $(Q, \alpha Q) \in \mathbb{G}_2^2$ for unknown $\alpha \in \mathbb{Z}_p^*$. And its goal is to compute α by interacting with adversary $\mathcal{A}_{II}$.

Throughout the game, we assume that H_1 Queries are distinct, that the target identity $ID_{\mathbb{S}}^*$ is submitted to H_1 Queries at some point and that any query involving an identity ID comes after a H_1 Queries on ID. To maintain consistency between queries made by $\mathcal{A}_{II}$, C

keeps the lists: L_i for $i = 1, 2, 3$ and L_{pk} as in the proof of **Lemma 1**. C randomly picks $s \in_R \mathbb{Z}_p^*$ as the $\mathsf{master\text{-}key}$, computes $P_{pub} = sQ$. and sends $< \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, P, Q, g, P_{pub}, \psi >$ and the $\mathsf{master\text{-}key}$ s to $\mathcal{A}_{II}$. Then, C randomly picks $\mu \in_R \{1, 2, \cdots, q_1\}$ and answers various oracle queries as in the proof of **Lemma 2**.

Eventually, $\mathcal{A}_I$ outputs a valid ciphertext $\sigma = (c, u, v, w)$ from $ID_{\mathbb{S}}^*$ to $ID_{\mathbb{R}}^*$. If $ID_{\mathbb{S}}^* \neq ID_\mu$, C aborts. Otherwise, having the knowledge of $sk_{ID_{\mathbb{R}}^*}$, C computes $h_2 = H_2(m^*, u, e(d_{ID_{\mathbb{R}}^*}, u), x_{ID_{\mathbb{R}}^*}u, pk_{ID_{\mathbb{S}}^*}, pk_{ID_{\mathbb{R}}^*})$, where $m^* = \mathsf{Unsigncrypt}(\mathsf{ID}_{\mathbb{S}}^*, \mathsf{pk}_{\mathsf{ID}_{\mathbb{S}}^*}, \mathsf{sk}_{\mathsf{ID}_{\mathbb{R}}^*}, \sigma)$ (For simplicity, we denote $\sigma = (c, u, v, w, h_2)$ as $\mathcal{A}_{II}$'s outputs). Then, using the oracle replay technique [18], C generates one more valid ciphertext from $\sigma = (c, u, v, w, h_2)$ which is named as $\sigma' = (c, u, v', w', h_2')$. This is achieved by running the turing machine again with the same random tape but with the different hash value.

Since $\sigma = (c, u, v, w, h_2)$ and $\sigma' = (c, u, v', w', h_2')$ are both valid ciphertext for the same message m^* and randomness r_1, we obtain the relations

$$w' T_{ID_{\mathbb{S}}^*} - h_2' pk_{ID_{\mathbb{S}}^*} = r_1 T_{ID_{\mathbb{S}}^*} = w T_{ID_{\mathbb{S}}^*} - h_2 pk_{ID_{\mathbb{S}}^*}.$$

Then, we have

$$\begin{aligned} (w' - w) T_{ID_{\mathbb{S}}^*} &= (h_2' - h_2) pk_{ID_{\mathbb{S}}^*}, \\ (h_2' - h_2)^{-1}(w' - w) T_{ID_{\mathbb{S}}^*} &= \alpha T_{ID_{\mathbb{S}}^*}. \end{aligned}$$

Hence, C can compute $\alpha = (h_2' - h_2)^{-1}(w' - w)$ as the solution of DLP.

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_μ.
E_2 : A $\mathsf{Private\ Key\ Extraction}$ query is made on ID_μ.
E_3 : C fails in using the oracle replay technique [18] to generate one more valid ciphertext.

We clearly have $Pr[\neg E_1] = 1/q_1$ and we know that $\neg E_1$ implies $\neg E_2$. From **Lemma 12** in [18], we know that $Pr[\neg E_3] \geq 1/9$. We obtain the announced bound by noting that

$$Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] \geq \frac{1}{9q_1}.$$

From the proof of **Lemma 12** in [18], we know that the total running time $\mathcal{T}'$ of solving the DLP with probability $\epsilon' \geq \frac{1}{9q_1}$ is bound by $23q_2[\mathcal{T} + (q_{pk} + q_{sk} + 5q_s + 2q_u(q_2 + q_s))\mathcal{T}_{mult} + q_u(q_2 + q_s)\mathcal{T}_{exp} + (q_s + 7q_u(q_2 + q_s))\mathcal{T}_p][\epsilon(1 - q_u/2^k)(1 - q_s(q_2 + q_3 + q_s)/2^k)]^{-1} + \mathcal{T}_{mult}$ where the last term accounts for the cost of compute the system public key in the preparation phase. Thus, this completes the proof.

5.2 Efficiency

There are almost three CLSC schemes in the literature [2], [16] and [23]. We now compare our CLSC scheme with the scheme proposed in [2], [16] and [23] from the aspect of computational cost in Table 1. In Table 1 we use *Hash*, *Mult*, *Exp*, and *Paring* as abbreviations for hash operations, point multiplications in $\mathbb{G}_1$ or $\mathbb{G}_2$, exponentiations in $\mathbb{G}_T$ and pairing computations respectively.

Table 1: Comparison of the CLSC Schemes

Schemes	*Hash*	*Mult*	*Exp*	*Paring*
Barbosa-Farshim [2]	6	5	1	6
Wu-Chen [23]	3	3	8	4
Liu-Hu-Zhang-Ma [16]	2	3	1	5
Our Scheme	4	7	2	2

According to the result in [3, 5], the pairing operation is several times more expensive than the multiplication in $\mathbb{G}_1$ and $\mathbb{G}_2$ and exponentiation in $\mathbb{G}_T$. Hence reducing the number of pairing operations is critical. As shown in Table 1, our CLSC scheme only requires two pairing operations in signcrypt and unsigncrypt phases. Above all, our scheme is more efficient than all the schemes available.

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 proposed a new efficient certificateless signcryption scheme based on bilinear pairing, which requires only two pairing operations in the signcrypt and unsigncrypt phases. The security of our scheme is based on the hardness assumptions of DLP, CDHP, q-SDHP and q-BDHIP.

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

[3] P. S. L. M. Barreto, S. D. Galbraith, C. Ó' hÉigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. *Designs, Codes and Cryptography*, 42(3):239–271, 2007.

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

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

[6] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. Cryptology ePrint Archive: Report 2005/133, Available from: http://eprint.iacr.org/2005/133.

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

[9] D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology-EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag, 2004.

[10] D. Boneh and X. Boyen. Short signatures without random oracles. In *Advances in Cryptology-EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, 2004.

[11] 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.

[12] 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.

[13] Y. Chen and F. Zhang. A new certificateless public key encryption scheme. *Wuhan University Journal of Natural Sciences*, 13(6):721–726, 2008.

[14] 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.

[15] 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.

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

[17] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundamentals*, E84-A(5):1234–1243, 2001.

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

[19] 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.

[20] 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.

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

[22] N. P. Smart and F. Vercauteren. On computable isomorphisms in efficient asymmetric pairing based systems. Cryptology ePrint Archive: Report 2005/116, Available from: http://eprint.iacr.org/2005/116.

[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.