Certificate-Based Signcryption: Security Model and Efficient Construction

Yang Lu, Jiguo Li

College of Computer and Information Engineering, Hohai University, Nanjing, China
Email: luyangnsd@163.com, lijiguo@hhu.edu.cn

Abstract: Signcryption is an important cryptographic primitive that simultaneously achieves confidentiality and authentication in an efficient manner. In 2008, Luo *et al.* introduced the notion of certificate-based signcryption and proposed the first construction of certificate-based signcryption. However, their scheme is insecure under the key replacement attack and also does not provide insider security. To overcome these disadvantages, we introduce a strengthened security model of certificate-based signcryption in this paper. The new security model accurately models insider security and the key replacement attacks that might be attempted by an adversary in a real certificate-based signcryption system. We also propose a new certificate-based signcryption scheme that reaches insider security and resists key replacement attacks. We show that this scheme is both chosen-ciphertext secure and existentially unforgeable in the random oracle model. Furthermore, performance analysis shows that the proposed scheme is efficient and practical.

Keywords: Certificate-based signcryption, Key replacement attack, Insider security, Strengthened security model, Chosen-ciphertext security, Existential unforgeability.

1. Introduction

Public key cryptography (PKC) is an important technique to realize network and information security. In traditional PKC, a Public Key Infrastructure (PKI) is used to provide an assurance to the users about the relationship between a public key and the holder of the corresponding private key by certificates. However, the need for PKI-supported certificates is considered the main difficulty in the deployment and management of traditional PKC. To simplify the management of the certificates, Shamir [24] introduced the concept of identity-based cryptography (IBC) in which the public key of each user is derived directly from his identity, such as an IP address or an e-mail address, and the corresponding private key is generated by a trusted third party called Private Key Generator (PKG). The main practical benefit of IBC lies in the reduction of need for public key certificates. However, if the KGC becomes dishonest, it can impersonate any user using its knowledge of the user's private key. This is due to the key escrow problem inherent in IBC. In addition, private keys must be sent to the users over secure channels, so private key distribution in IBC becomes a very daunting task.

In Eurocrypt 2003, Gentry [10] introduced the notion of certificate-based encryption (CBE), which represents an interesting and potentially useful balance between IBC and traditional PKC. As in traditional PKC, each user in CBE generates his own public/private key pair and requests a certificate from a CA. The difference is that a certificate in CBE acts not only as a certificate (as in the traditional PKI) but also as a decryption key (as in IBC). This additional functionality provides an efficient implicit certificate mechanism so that a receiver needs both his private key and an up-to-date certificate from the CA to decrypt a ciphertext sent to him, while senders need not be concerned about the certificate revocation problem. The feature of implicit certificate allows us to eliminate third-party queries for the certificate status and simplify the certificate revocation in the traditional PKI. As a result, CBE does not need infrastructures like Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP). Furthermore, CBE overcomes the key escrow problem (since the CA does not know the private keys of users) and key distribution problem (since the certificates need not be kept secret) inherent in IBC. In parallel to CBE, Kang *et al.* [11] proposed the notion of certificate-based signature (CBS) that follows the idea of CBE presented by Gentry [10].

The topic of certificated-based cryptography has undergone quite rapid development in the recent years, with many schemes being proposed for encryption [1, 10, 16, 18, 19, 22, 26, 27, 29] and signature [3, 11, 12, 13, 15, 17, 28]. As an extension of the signcryption [30] in the certificate-based setting, Luo *et al.* [20] introduced the concept of certificate-based signcryption (CBSC) that simultaneously provides the functionalities of CBE and CBS. They also proposed the first construction of CBSC and partly proved its security in the random oracle model [6, 8]. However, the security model of CBSC defined in [20] does not consider the key replacement attack which refers to the attack that an adversary replaces a user's public key with a false public key of its choice and dupes any other third party to encrypt messages or verify signatures using this false public key. At the first glance, one may think that this kind of attack does not exist in certificate-based cryptosystems due to the use of certificates. However, as introduced by Li *et al.* in [12], it does exist. In a certificate-based cryptosystem, the CA does issue the certificates. But, only the holder of a certificate needs to check the validity of its certificate and other users do not need. Therefore, certificate-based cryptosystems are susceptible to the key replacement attack. Unfortunately, Luo *et al.*'s CBSC scheme is insecure under this kind of attack. We will give the concrete attack on their scheme in Section 3 of this paper. Furthermore, Luo *et al.*'s security model does not consider insider security [2] too. Insider security is a necessary security requirement for a signcryption scheme to achieve. It requires that even if the sender's private key is compromised, an adversary should not be able to unsigncrypt the message and even with the receiver's private key, a forger should not be able to generate a valid signcryption as if generated by the sender. Our key replacement attack also shows that Luo *et al.*'s scheme fails in providing insider security. Besides Luo *et al.*'s scheme, Li *et al.* [14] proposed another CBSC scheme. They claimed that their scheme was both chosen-ciphertext secure and existentially unforgeable in the random oracle model. However, no strict proof was given in [14]. As far as the authors know, there exist only these two CBSC schemes in the literature.

1.1 Our Results

In this paper, we first introduce a strengthened security model of CBSC that accurately models the key replacement attack and insider security. We show that Luo *et al.*'s CBSC scheme [20] is insecure in our strengthened security model. We then develop a new CBSC scheme based on the CBE scheme proposed by Lu *et al.* [19] and prove it to be both chosen-ciphertext secure and existentially unforgeable in the random oracle model. The proposed CBSC scheme not only reaches insider security, but also resists key replacement attacks. Compared with the previous CBSC schemes in the literature, our scheme enjoys better performance, especially in the computation efficiency.

1.2 Paper Organization

The rest of this paper is organized as follows. In the next section, we briefly review some preliminaries required in this paper. In Section 3, we present our strengthened security model of CBSC. In Section 4, we show that Luo *et al.*'s CBSC scheme is insecure under the key replacement attack. In Section 5, we propose a new CBSC scheme and prove its security in the proposed security model. We also compare our scheme with other existing CBSC schemes in terms of security and performance. Finally, we draw our conclusions in Section 6.

2. Preliminaries

Let k be a security parameter and p be a k-bit prime number. Let G be an additive cyclic group of prime order p and G_T be a multiplicative cyclic group of the same order, and P be a generator of G. A bilinear paring is a map e: $G \times G \rightarrow G_T$ satisfying the following properties:

(1) Bilinearity: For all $P_1, P_2 \in G$, and all $a, b \in Z_p^*$, we have $e(aP_1, bP_2) = e(P_1, P_2)^{ab}$.
(2) Non-degeneracy: $e(P, P) \neq 1$.
(3) Computability: For all $P_1, P_2 \in G$, $e(P_1, P_2)$ can be efficiently computed.

A bilinear map satisfying the above properties is said to be an admissible bilinear map. Typically, the map e can be derived from either the Weil or Tate paring on an elliptic curve over a finite field [7]. Now we recall the following computational assumptions that are relevant to the security of our CBSC scheme.

Definition 1. The computational Diffie-Hellman (CDH) problem in G is, given a tuple $(P, aP, bP) \in G^3$ for unknown $a, b \in Z_p^*$, to compute $abP \in G$. The advantage of a probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ in solving the CDH problem in G is defined as

$$Adv_{\mathcal{A}}^{CDH}(k) = \Pr[\mathcal{A}(P, aP, bP) = abP].$$

We say that CDH problem in G is hard if $Adv_{\mathcal{A}}^{CDH}(k)$ is negligible for all PPT algorithms $\mathcal{A}$.

Definition 2 [7]. The bilinear Diffie-Hellman (BDH) problem in (G, G_T) is, given a tuple $(P, aP, bP, cP) \in G^4$ for unknown $a, b, c \in Z_p^*$, to compute $e(P, P)^{abc} \in G_T$. The advantage of a PPT algorithm $\mathcal{A}$ in solving the BDH problem in (G, G_T) is defined as

$$Adv_{\mathcal{A}}^{BDH}(k) = \Pr[\mathcal{A}(P, aP, bP, cP) = e(P, P)^{abc}].$$

We say that BDH problem in (G, G_T) is hard if $Adv_{\mathcal{A}}^{BDH}(k)$ is negligible for all PPT algorithms $\mathcal{A}$.

Definition 3 [21]. The collusion attack algorithm with q-traitors (q-CAA) problem in G is, given a tuple $(P, \alpha P, (\omega_1 + \alpha)^{-1}P,\ldots, (\omega_q + \alpha)^{-1}P, \omega_1,\ldots, \omega_q) \in G^{q+2} \times (Z_p^*)^q$ for unknown $\alpha \in Z_p^*$, to compute $(\omega^* + \alpha)^{-1}P$ for some value $\omega^* \notin \{\omega_1,\ldots, \omega_q\}$. The advantage of a PPT algorithm $\mathcal{A}$ in solving the q-CAA problem in G is defined as

$$Adv_{\mathcal{A}}^{q\text{-}CAA}(k) = \Pr[\mathcal{A}(P, \alpha P, (\omega_1 + \alpha)^{-1}P,\ldots, (\omega_q + \alpha)^{-1}P, \omega_1,\ldots, \omega_q) = (\omega^* + \alpha)^{-1}P \mid \alpha, \omega^* \in Z_p^*, \omega^* \notin \{\omega_1,\ldots, \omega_q\}].$$

We say that q-CAA problem in G is hard if $Adv_{\mathcal{A}}^{q\text{-}CAA}(k)$ is negligible for all PPT algorithms $\mathcal{A}$.

Definition 4 [25]. The modified bilinear Diffie-Hellman inversion for q-values (q-mBDHI) problem in G is, given a tuple $(P, \alpha P, (\omega_1 + \alpha)^{-1}P,\ldots, (\omega_q + \alpha)^{-1}P, \omega_1,\ldots, \omega_q) \in G^{q+2} \times (Z_p^*)^q$ for unknown $\alpha \in Z_p^*$, to compute $e(P,P)^{(\omega^*+\alpha)^{-1}}$ for some value $\omega^* \in Z_p^* - \{\omega_1,\ldots, \omega_q\}$. The advantage of a PPT algorithm $\mathcal{A}$ in solving the q-mBDHI problem in (G, G_T) is defined as

$$Adv_{\mathcal{A}}^{q\text{-}mBDHI}(k) = \Pr[\mathcal{A}(P, \alpha P, (\omega_1 + \alpha)^{-1}P,\ldots, (\omega_q + \alpha)^{-1}P, \omega_1,\ldots, \omega_q) = e(P,P)^{(\omega^*+\alpha)^{-1}} \mid \alpha, \omega^* \in Z_p^*, \omega^* \notin \{\omega_1,\ldots, \omega_q\}].$$

We say that q-mBDHI problem in G is hard if $Adv_{\mathcal{A}}^{q\text{-}mBDHI}(k)$ is negligible for all PPT algorithms $\mathcal{A}$.

3. Certificate-Based Signcryption

In this section, we first introduce the definition of CBSC. Then, we present a strengthened security model of CBSC.

3.1 Definition of Certificate-Based Signcryption

In a CBCS system, a CA will first generate the system parameter including a master key and a list of public parameters. The CA will use its master key to issue certificates for users in the system. Users will generate their own public/private key pairs and then contact the CA to obtain the corresponding certificates. A user should use his private key and certificate to decrypt the ciphertext sent to him or generate a signature on a message.

Formally, a CBSC scheme is specified by the following five algorithms:

Setup: This algorithm takes as input a security parameter 1^k and outputs a master key *msk* and a list of public parameters *params* that include the descriptions of a finite message space $\mathcal{M}$ and a finite ciphertext space $\mathcal{C}$. After the algorithm is performed, the CA publishes the public parameters *params* and keeps the master key *msk* secret.

UserKeyGen: This algorithm takes as input *params* and outputs a public/private key pair.

This algorithm is run once by every user independently. Specifically, when a user with identity *id* runs the algorithm, the key pair generated is denoted as (PK_{id}, SK_{id}).

CertGen: This algorithm takes as input *params*, *msk*, an identity *id* and a public key PK_{id}, and outputs a certificate $Cert_{id}$ which is sent to the user *id* through an open channel. This algorithm is performed by a CA.

Signcrypt: This algorithm takes *params*, a message $M \in \mathcal{M}$, a sender's identity id_S, public key PK_{id_S}, private key SK_{id_S} and certificate $Cert_{id_S}$, and a receiver's identity id_R and public key PK_{id_R} as input, and outputs a ciphertext $\sigma \in \mathcal{C}$.

Designcrypt: This algorithm takes *params*, a ciphertext $\sigma \in \mathcal{C}$, the receiver's identity id_R, public key PK_{id_R}, private key SK_{id_R} and certificate $Cert_{id_R}$, and the sender's identity id_S and public key PK_{id_S} as input, and outputs either a message $M \in \mathcal{M}$ or an error symbol $\perp$ if σ is not a valid ciphertext.

For correctness, we require that if $\sigma = \mathsf{Signcrypt}(params, M, id_S, PK_{id_S}, SK_{id_S}, Cert_{id_S}, id_R, PK_{id_R})$, then $M = \mathsf{Designcrypt}(params, \sigma, id_R, PK_{id_R}, SK_{id_R}, Cert_{id_R}, id_S, PK_{id_S})$.

Note that there are six algorithms defined in Luo *et al.*'s definition of CBSC [20]. In their definition, a signcryption of a message should be first decrypted by a Receiver Decrypt algorithm and then be verified by another Receiver Verify algorithm. For simplicity, we replace these two algorithms with a single Designcrypt algorithm. It is easy to see that a CBSC scheme presented in Luo *et al.*'s six-algorithm form can also be presented in our new form.

3.2 Strengthened Security Model of Certificate-Based Signcryption

As introduced in [20], a CBSC scheme should satisfy both confidentiality (indistinguishability against adaptive chosen-ciphertext attacks (IND-CBSC-CCA2)) and unforgeability (existential unforgeability against adaptive chosen-messages attacks (EUF-CBSC-CMA)). To define these security notions, our security model distinguishes two different types of adversaries:

- **Type-I adversary** (denoted by $\mathcal{A}_I$) simulates an uncertified user who can replace public keys of any users but is not allowed to access the master key. In Luo *et al.*'s security model [20], such an adversary is not given the ability to make key replacement attacks. However, as discussed in Section 1, their definition does not reflect the ability of a Type-I adversary in a real CBSC system.
- **Type-II adversary** (denoted by $\mathcal{A}_{II}$) models an honest-but-curious CA who is equipped with the master key but is not allowed to replace any user's public key. It is clear that if a Type-II adversary can replace any user's public key, then it may trivially break the security of a CBCS system using a man-in-the-middle attack.

Below, we give our definitions. Note that we do not consider attacks targeting ciphertexts where the identities of the sender and receiver are the same. That is, we disallow such queries to relevant oracles and do not accept this type of ciphertext as a valid forgery.

3.2.1 Confidentiality

For the confidentiality, we consider two different games "IND-CBSC-CCA2-I" and "IND-CBSC-CCA2-II", in which a Type-I adversary $\mathcal{A}_I$ and a Type-II adversary $\mathcal{A}_{II}$ interact with the game challenger respectively.

IND-CBSC-CCA2-I: This is the game in which $\mathcal{A}_I$ interacts with the challenger.

- **Setup.** The challenger runs the algorithm $\mathsf{Setup}(1^k)$ to generate *msk* and *params*. It then returns *params* to $\mathcal{A}_I$ and keeps *msk* to itself.

– **Phase 1.** In this phase, $\mathcal{A}_I$ adaptively makes requests to the following six oracles.

- CreateUser: On input an identity *id*, if *id* has already been created, the challenger outputs the current public key PK_{id} associated with the identity *id*. Otherwise, the challenger runs the algorithm UserKeyGen to generate a private/public key pair (SK_{id}, PK_{id}). It then outputs PK_{id} and inserts (id, SK_{id}, PK_{id}) into a list which records the information about a created user's private key and public key. In this case, *id* is said to be created. We assume that other oracles defined below only respond to an identity which has been created.
- ReplacePublicKey: On input an identity *id* and a value PK'_{id}, the challenger replaces the current public key of the identity *id* with PK'_{id}. Note that the current value of an entity's public key is used by the challenger in any computations or responses to $\mathcal{A}_I$'s requests. Such an oracle models the ability of a Type-I adversary to convince a legitimate user to use a false public key and enables our security model to capture the key replacement attacks attempted by the Type-I adversary.
- GenerateCertificate: On input an identity *id*, the challenger responds with the corresponding certificate $Cert_{id}$. If the identity *id* has no associated certificate, then the challenger generates a certificate $Cert_{id}$ for *id* by running the algorithm CertGen.
- ExtractPrivateKey: On input an identity *id*, the challenger responds with the corresponding private key SK_{id}. Here, $\mathcal{A}_I$ is disallowed 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 the challenger to be able to provide a private key of a user for which it does not know the private key.
- Signcryption: On input a message *M*, a sender's identity id_S and a receiver's identity id_R, the challenger responds with σ = Signcrypt($params$, M, id_S, PK_{id_S}, SK_{id_S}, $Cert_{id_S}$, id_R, PK_{id_R}). Note that it is possible that the challenger is not aware of the sender's private key if the associated public key has been replaced. In this case, we require $\mathcal{A}_I$ to provide it. We disallow queries where $id_S = id_R$.
- Designcryption: On input a ciphertext σ, a sender's identity id_S and a receiver's identity id_R, the challenger responds with the result of Designcrypt($params$, σ, id_R, PK_{id_R}, SK_{id_R}, $Cert_{id_R}$, id_S, PK_{id_S}). Note that it is possible that the challenger is not aware of the receiver's private key if the associated public key has been replaced. In this case, we require the adversary to provide it. Again, we disallow queries where $id_S = id_R$.

– **Challenge.** Once $\mathcal{A}_I$ decides that Phase 1 is over, it outputs two equal-length messages (M_0, M_1) and two distinct identities (id_S^*, id_R^*). The challenger picks a random bit *b* and computes the challenge ciphertext as σ^* = Signcrypt($params$, M_b, id_S^*, $PK_{id_S^*}$, $SK_{id_S^*}$, $Cert_{id_S^*}$, id_R^*, $PK_{id_R^*}$). It then returns σ^* to $\mathcal{A}_I$.

– **Phase 2.** In this phase, $\mathcal{A}_I$ continues to issues queries as in Phase 1.

– **Guess.** Finally, $\mathcal{A}_I$ outputs a guess $b' \in \{0, 1\}$. We say that $\mathcal{A}_I$ wins the game if $b = b'$ and the following conditions are simultaneously satisfied: (1) $\mathcal{A}_I$ cannot query GenerateCertificate on the identity id_R^* at any point; (2) $\mathcal{A}_I$ cannot query ExtractPrivateKey on an identity if the corresponding public key has been replaced; (3) In phase 2, $\mathcal{A}_I$ cannot query Designcryption on (σ^*, id_S^*, id_R^*) unless the public key of the sender id_S^* or that of the receiver id_R^* has been replaced after the challenge was issued. We define $\mathcal{A}_I$'s advantage in this game to be

$$Adv_{\mathcal{A}_I}^{IND\text{-}CBSC\text{-}CCA2\text{-}I}(k) = 2|\Pr[b = b'] - 1/2|.$$

IND-CBSC-CCA2-II: This is the game in which $\mathcal{A}_{II}$ interacts with the challenger.

– **Setup.** The challenger runs the algorithm Setup(1^k) to generate *msk* and *params*. It then returns *params* and *msk* to $\mathcal{A}_{II}$.

- **Phase 1.** In this phase, $\mathcal{A}_{II}$ adaptively asks a polynomial bounded number of queries as in the game **IND-CBSC-CCA2-I**. The only restriction is that $\mathcal{A}_{II}$ can not replace public keys of any users. In addition, $\mathcal{A}_{II}$ need not make any GenerateCertificate queries since it can computes certificates of any identities by itself with the master key *msk*.
- **Challenge.** Once $\mathcal{A}_{II}$ decides that Phase 1 is over, it outputs two equal-length messages (M_0, M_1) and two distinct identities (id_S^*, id_R^*). The challenger picks a random bit b and computes the challenge ciphertext as σ^* = Signcrypt($params$, M_b, $id_S^*, PK_{id_S^*}, SK_{id_S^*}, Cert_{id_S^*}, id_R^*, PK_{id_R^*}$). It then returns σ^* to $\mathcal{A}_{II}$.
- **Phase 2.** In this phase, $\mathcal{A}_{II}$ continues to issues queries as in Phase 1.
- **Guess.** Finally, $\mathcal{A}_{II}$ outputs a guess $b' \in \{0, 1\}$. We say that $\mathcal{A}_{II}$ wins the game if $b = b'$ and the following conditions are simultaneously satisfied: (1) $\mathcal{A}_{II}$ cannot query ExtractPrivateKey on the identity id_R^* at any point; (2) $\mathcal{A}_{II}$ cannot query Designcryption on (σ^*, id_S^*, id_R^*) in phase 2. We define $\mathcal{A}_{II}$'s advantage in this game to be

$$Adv_{\mathcal{A}_{II}}^{IND\text{-}CBSC\text{-}CCA2\text{-}II}(k) = 2|\Pr[b = b'] - 1/2|.$$

Definition 5. A CBSC scheme is said to be indistinguishable against adaptive chosen-ciphertext attacks (or IND-CBSC-CCA2 secure) if no PPT adversary has non-negligible advantage in the above two games.

Note that the adversary in the definition of message confidentiality is allowed to be challenged on a ciphertext generated using a corrupted sender's private key and certificate. This condition corresponds to the stringent requirement of insider security for confidentiality of signcryption. This means that our security model ensures that the confidentiality of signcryption is preserved even if a sender's private key is compromised.

3.2.2 Unforgeability

For the unforgeability, we consider two different games "EUF-CBSC-CMA-I" and "EUF-CBSC-CMA-II", in which a Type-I adversary $\mathcal{A}_I$ and a Type-II adversary $\mathcal{A}_{II}$ interact with the challenger respectively.

EUF-CBSC-CMA-I: This is the game in which $\mathcal{A}_I$ interacts with the challenger.

- **Setup.** The challenger runs the algorithm Setup(1^k) to generate *msk* and *params*. It then returns *params* to $\mathcal{A}_I$ and keeps *msk* to itself.
- **Query.** In this phase, $\mathcal{A}_I$ adaptively asks a polynomial bounded number of queries as in the game **IND-CBSC-CCA2-I**.
- **Forge.** Finally, $\mathcal{A}_I$ outputs a forgery (σ^*, id_S^*, id_R^*). We say that $\mathcal{A}_I$ wins the game if the result of Designcrypt($params$, $\sigma^*, id_R^*, PK_{id_R^*}, SK_{id_R^*}, Cert_{id_R^*}, id_S^*, PK_{id_S^*}$) is not the $\perp$ symbol and the following conditions are simultaneously satisfied: (1) $\mathcal{A}_I$ cannot query GenerateCertificate on the identity id_S^* at any point; (2) $\mathcal{A}_I$ cannot query ExtractPrivateKey on an identity if the corresponding public key has been replaced; (3) σ^* is not the output of any Signcryption query on (M^*, id_S^*, id_R^*). We define $\mathcal{A}_I$'s advantage $Adv_{\mathcal{A}_I}^{EUF\text{-}CBSC\text{-}CMA\text{-}I}(k)$ to be the probability that it wins the game.

EUF-CBSC-CMA-II: This is the game in which $\mathcal{A}_{II}$ interacts with the challenger.

- **Setup.** The challenger runs the algorithm Setup(1^k) to generate *msk* and *params*. It then returns *params* and *msk* to $\mathcal{A}_{II}$.

- **Query.** In this phase, $\mathcal{A}_{II}$ can adaptively ask a polynomial bounded number of queries as in the game **IND-CBSC-CCA2-II**.
- **Forge.** Finally, $\mathcal{A}_{II}$ outputs a forgery $(\sigma^*, id_S^*, id_R^*)$. We say that $\mathcal{A}_{II}$ wins the game if the result of Designcrypt($params$, $\sigma^*, id_R^*, PK_{id_R^*}, SK_{id_R^*}, Cert_{id_R^*}, id_S^*, PK_{id_S^*}$) is not the $\perp$ symbol and the following conditions are simultaneously satisfied: (1) $\mathcal{A}_{II}$ cannot query ExtractPrivateKey on the identity id_S^*; (2) σ^* is not the output of any Signcryption query on (M^*, id_S^*, id_R^*). We define $\mathcal{A}_{II}$'s advantage $Adv_{\mathcal{A}_{II}}^{EUF\text{-}CBSC\text{-}CMA\text{-}II}(k)$ to be the probability that it wins the game.

Definition 6. A CBSC scheme is said to be existential unforgeable against adaptive chosen-messages attacks (or EUF-CBSC-CMA secure) if no PPT adversary has non-negligible advantage in the above two games.

Note that the adversary in the definition of signature unforgeability may output a ciphertext generated using a corrupted receiver's private key and certificate. Again, this condition corresponds to the stringent requirement of insider security for unforgeability of signcryption. Hence, our security model also ensures that the unforgeability of signcryption is preserved even if a receiver's private key is compromised.

4. Certificate-Based Signcryption Scheme of Luo *et al.*

In this section, we give the review and attack of the CBSC scheme by Luo *et al.* [20].

4.1 Overview of the Scheme

Luo *et al.*'s CBSC scheme consists of the following six algorithms:

Setup: Given a security parameter k, the CA performs as follows:

(1) Generate two cyclic groups G and G_T of prime order p such that there exists a bilinear paring map e: $G \times G \to G_T$.
(2) Select a random element $s \in Z_p^*$ as the master key, choose a random generator $P \in G$ and set $P_{pub} = sP$.
(3) Select four hash functions H_1: $\{0,1\}^n \times G \to G$, H_2: $\{0,1\}^n \times G \times G \to G$, H_3: $G \times G \times \{0,1\}^n \to Z_p^*$ and H_4: $G_T \to \{0,1\}^n$.
(4) The public parameters are $params = \{p, G, G_T, e, n, P, P_{pub}, H_1, H_2, H_3, H_4\}$ and the certifier's master key is $msk = s$.

UserKeyGen: Given $params$, a user with identity id chooses a random $x_{id} \in Z_p^*$ as his private key SK_{id}, and then computes the corresponding public key as $PK_{id} = x_{id}P$.

CertGen: To generate a certificate for the user with identity $id \in \{0, 1\}^n$ and public key PK_{id}, the CA computes $Q_{id} = H_1(id, PK_{id})$ and outputs $Cert_{id} = sQ_{id}$ as the certificate.

Sender Signcrypt: To send a message $M \in \{0, 1\}^n$ to the receiver id_R, the sender id_S does the following:

(1) Randomly choose $x \in Z_p^*$ and compute $R = xP$ and $S = H_2(id_S, PK_{id_S}, R)$.
(2) Compute $h = H_3(R, S, M)$ and $V = x^{-1}(Cert_{id_S} + SK_{id_S}S + hP_{pub})$.
(3) Compute $W = e(PK_{id_S}, PK_{id_R})^x$ and then $C = M \oplus H_4(W)$.
(4) Output $\sigma = (C, R, V)$ as the ciphertext.

Receiver Decrypt: When receiving a ciphertext $\sigma = (C, R, V)$ from the sender id_S, the receiver id_R does the following:

(1) Compute $W = e(R, SK_{id_R}PK_{id_S})$ and then $M = C \oplus H_4(W)$.
(2) Forward the message M and signature (R, V) to the Receiver Verifry algorithm.

Receiver Verifry: To verify the sender id_S's signature (R, V) on the message M, the receiver id_R does the following:

(1) Compute $S = H_2(id_S, PK_{id_S}, R)$ and $h = H_3(R, S, M)$.

(2) Check whether $e(R,V) = e(P_{pub}, Q_{id_S})e(PK_{id_S}, S)e(P, P_{pub})^h$. If the check holds, output M, otherwise output $\perp$.

4.2 Key Replacement Attack on the Scheme

The Type-I adversary $\mathcal{A}_I$ who is capable of replacing any user's public key and is not allowed to know the master key can forge a valid signcryption on any message M from id_S to id_R by performing the following steps:

(1) Randomly choose $x^{'} \in Z_p^*$ and compute $R^{'} = x^{'}P_{pub}$.

(2) Randomly choose $y \in Z_p^*$ and then replace the public key of the user id_S with $PK^{'}_{id_S} = yP_{pub}$.

(3) Compute $S^{'} = H_2(id_S, PK^{'}_{id_S}, R^{'})$.

(4) Choose a message M and compute $h^{'} = H_3(R^{'}, S^{'}, M)$.

(5) Compute $Q_{id_S} = H_1(id_S, PK^{'}_{id_S})$ and then $V^{'} = x^{'-1}(Q_{id_S} + yS^{'} + h^{'}P)$.

(6) Randomly choose $C^{'} \in \{0, 1\}^n$ and outputs the signcryption on message M as $\sigma^{'} = (C^{'}, R^{'}, V^{'})$. If the adversary has corrupted the private key SK_{id_R} of the receiver id_R, then it can set $C^{'} = M \oplus H_4(W^{'})$, where $W^{'} = e(R^{'}, SK_{id_R} PK^{'}_{id_S})$.

The ciphertext $\sigma^{'} = (C^{'}, R^{'}, V^{'})$ passes the verification test as shown blow.

$$\begin{aligned}
& e(P_{pub}, Q_{id_S})e(PK^{'}_{id_S}, S^{'})e(P, P_{pub})^{h^{'}} \\
&= e(P_{pub}, Q_{id_S})e(yP_{pub}, S^{'})e(P, P_{pub})^{h^{'}} \\
&= e(P_{pub}, Q_{id_S} + yS^{'} + h^{'}P) \\
&= e(x^{'}P_{pub}, x^{'-1}(Q_{id_S} + yS^{'} + h^{'}P)) \\
&= e(R^{'}, V^{'}) .
\end{aligned}$$

This proves that the forgery generated is valid. In addition, we can see that Luo *et al.*'s CBSC scheme doses not achieve insider security under the key replacement attack, since the adversary can forge a valid signcryption using the corrupted receiver's private key.

5. The Proposed Certificate-Based Signcryption Scheme

In this section, we propose a new CBSC scheme and prove its security in our strengthened security model. We also compare it with other existing CBSC schemes in the literature.

5.1 Concrete Construction

Our scheme is constructed from the CBE scheme proposed by Lu *et al.* [19]. It consists of the following algorithms:

Setup: Given a security parameter k, the CA performs the following to set up the system:

(1) Generate two cyclic groups G and G_T of prime order p such that there exists a bilinear paring map e: $G \times G \to G_T$.

(2) Choose two random generators $P, Q \in G$ and set $g = e(P, Q)$.

(3) Choose a random element $\alpha \in Z_p^*$ and set $P_{pub} = \alpha P$.

(4) Select three hash functions H_1: $\{0,1\}^* \times G_T \to Z_p^*$, H_2: $G_T \times G_T \to \{0,1\}^n$ and H_3: $\{0,1\}^* \to Z_p^*$, where n is the bit-length of the message to be signcrypted.

(5) The public parameters are $params = \{p, G, G_T, e, n, P, Q, P_{pub}, g, H_1, H_2, H_3\}$ and the certifier's master key is $msk = \alpha$.

UserKeyGen: Given *params,* a user with identity *id* chooses a random $x_{id} \in Z_p^*$ as his private key SK_{id}, and then computes the corresponding public key as $PK_{id} = g^{x_{id}} \in G_T$.

CertGen: To generate a certificate for a user with identity *id* and public key PK_{id}, the CA computes $Cert_{id} = (H_1(id, PK_{id}) + \alpha)^{-1}Q$ as the certificate for the user id. The user id can check the validness of $Cert_{id}$ by verifying whether $e(H_1(id, PK_{id})P + P_{pub}, Cert_{id}) = g$.

Signcrypt: To send a message $m \in \{0, 1\}^n$ to the receiver id_R, the sender id_S does the following:

(1) Randomly choose $r \in Z_p^*$ and compute $R_1 = g^r$ and $R_2 = (PK_{id_R})^r$.

(2) Compute $U = r(H_1(id_{id_R}, PK_{id_R})P + P_{pub})$ and $C = M \oplus H_2(R_1, R_2)$.

(3) Compute $h = H_3(M, U, R_1, R_2, id_S, PK_{id_S}, id_R, PK_{id_R})$ and $V = (hSK_{id_S} + r) \cdot Cert_{id_S}$.

(4) Set $\sigma = (C, U, V)$ as the ciphertext.

Designcrypt: To designcrypt a ciphertext $\sigma = (C, U, V)$ from the sender id_S, the receiver id_R does the following:

(1) Compute $R_1 = e(U, Cert_{id_R})$ and $R_2 = e(U, Cert_{id_R})^{SK_{id_R}}$.

(2) Compute $M = C \oplus H_2(R_1, R_2)$.

(3) Compute $h = H_3(M, U, R_1, R_2, id_S, PK_{id_S}, id_R, PK_{id_R})$.

(4) Check whether $e(H_1(id_S, PK_{id_S})P + P_{pub}, V)(PK_{id_S})^{-h} = R_1$. If the check holds, output M, otherwise output $\perp$.

The consistency of the above scheme can be easily verified by the following equalities

$$e(U, Cert_{id_R}) = e(r(H_1(id_R, PK_{id_R})P + P_{pub}), (H_1(id_R, PK_{id_R}) + \alpha)^{-1}Q) = e(P, Q)^r = g^r.$$

$$e(U, Cert_{id_R})^{SK_{id_R}} = e(P, Q)^{SK_{id_R} \cdot r} = (PK_{id_R})^r.$$

$$\begin{aligned} & e(H_1(id_S, PK_{id_S})P + P_{pub}, V) \cdot (PK_{id_S})^{-h} \\ &= e((H_1(id_S, PK_{id_S}) + \alpha)P, (hSK_{id_S} + r)(H_1(id_R, PK_{id_R}) + \alpha)^{-1}Q) \cdot e(P, Q)^{-hSK_{id_S}} \\ &= e(P, Q)^r = R_1. \end{aligned}$$

5.2 Security Proof

Theorem 1. *The CBSC scheme above is IND-CBSC-CCA2 secure under the hardness of the q-mBDHI and BDH problems in the random oracle model.*

This theorem can be proved by combining the following two lemmas.

Lemma 1. *If an IND-CBSC-CCA2-I adversary $\mathcal{A}_I$ has advantage ε against our CBSC scheme when running in time τ, asking at most q_{cu} CreateUser queries, q_{sc} Signcryption queries, q_{dsc} Designcryption queries and q_i queries to random oracles H_i (i = 1, 2, 3), then there exists an algorithm $\mathcal{B}$ to solve the q-mBDHI problem for $q = q_1 - 1$, with advantage*

$$\varepsilon' \geq \frac{\varepsilon}{q_1(q_2 + 2q_3 + 2q_{sc})}(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k})$$

and with time $\tau' \leq \tau + O(q_1 + q_{sc} + q_{dsc})\cdot\tau_m + O(q_{cu} + q_{sc} + q_{dsc})\cdot\tau_e + O(q_{sc} + q_{dsc})\cdot\tau_p$ where τ_m, τ_e, τ_p denote the time for computing a scalar multiplication in G, an exponentiation in G_T, and a pairing respectively.

Proof. We show how to construct an algorithm $\mathcal{B}$ to solve the q-mBDHI problem from $\mathcal{A}_I$. Assume that $\mathcal{B}$ is given a random q-mBDHI instance $(P, \alpha P, (\omega_1 + \alpha)^{-1}P, \ldots, (\omega_q + \alpha)^{-1}P, \omega_1, \ldots, \omega_q)$ where α is a random element from Z_p^*. $\mathcal{B}$ interacts with $\mathcal{A}_I$ as follows:

In the setup phase, $\mathcal{B}$ randomly chooses $t \in Z_p^*$, and sets $P_{pub} = \alpha P$, $Q = tP$ and $g = e(P, Q)$. Furthermore, it randomly chooses a value $\omega^* \in Z_p^*$ such that $\omega^* \notin \{\omega_1, \ldots, \omega_q\}$ and an index $\theta \in [1, q_1]$. Then, $\mathcal{B}$ starts the game IND-CBSC-CCA2-I by supplying $\mathcal{A}_I$ with the public parameters $params = \{p, G, G_T, e, n, P, Q, P_{pub}, g, H_1, H_2, H_3\}$ where H_1~H_3 are random oracles controlled by

$\mathcal{B}$. $\mathcal{A}_I$ can make queries on these random oracles at any time during the game. Note that the corresponding master key is $msk = \alpha$ which is unknown to $\mathcal{B}$.

Now, $\mathcal{B}$ starts to respond various queries as follows:

H_1 Queries: We assume that q_1 queries to H_1 are distinct. $\mathcal{B}$ maintains a list H_1List of tuples $< id_i, PK_{id_i}, h_{1,i}, Cert_{id_i} >$ which is initially empty. On receiving such a query on (id_i, PK_{id_i}), $\mathcal{B}$ does the following:

– If (id_i, PK_{id_i}) already appears on H_1List in a tuple $< id_i, PK_{id_i}, h_{1,i}, Cert_{id_i} >$, then $\mathcal{B}$ returns $h_{1,i}$ to $\mathcal{A}_I$.

– Else if the query is on the θ-th distinct $(id_\theta, PK_{id_\theta})$, then $\mathcal{B}$ inserts $< id_\theta, PK_{id_\theta}, \omega^*, \perp >$ into H_1List and returns $h_{1,\theta} = \omega^*$ to $\mathcal{A}_I$. Note that the certificate for the identity id_θ is $Cert_{id_\theta} = t(\omega^* + \alpha)^{-1}P$ which is unknown to $\mathcal{B}$.

– Else $\mathcal{B}$ sets $h_{1,i}$ to be ω_j ($j \in [1, q]$) which has not been used and computes $Cert_{id_i} = t(\omega_j + \alpha)^{-1}P$. It then inserts $< id_i, PK_{id_i}, h_{1,i}, Cert_{id_i} >$ into H_1List and returns $h_{1,i}$ to $\mathcal{A}_I$.

H_2 Queries: $\mathcal{B}$ maintains a list H_2List of tuples $<R_1, R_2, h_2>$ which is initially empty. On receiving such a query on (R_1, R_2), $\mathcal{B}$ does the following:

– If (R_1, R_2) already appears on H_2List in a tuple $<R_1, R_2, h_2>$, then $\mathcal{B}$ returns h_2 to $\mathcal{A}_I$.

– Otherwise, it returns a random $h_2 \in \{0, 1\}^n$ to $\mathcal{A}_I$ and inserts $<R_1, R_2, h_2>$ into H_2List.

H_3 Queries: $\mathcal{B}$ maintains a list H_3List of tuples $<M, U, R_1, R_2, R_3, id_S, PK_{id_S}, id_R, PK_{id_R}, h_3, C>$ which is initially empty. On receiving such a query on $(M, U, R_1, R_2, R_3, id_S, PK_{id_S}, id_R, PK_{id_R})$, $\mathcal{B}$ does the following:

– If $(M, U, R_1, R_2, R_3, id_S, PK_{id_S}, id_R, PK_{id_R})$ already appears on H_3List in a tuple $<M, U, R_1, R_2, R_3, id_S, PK_{id_S}, id_R, PK_{id_R}, h_3, C>$, $\mathcal{B}$ returns h_3 to $\mathcal{A}_I$.

– Otherwise, it returns a random $h_3 \in Z_p^*$ to $\mathcal{A}_I$. To anticipate possible subsequent Designcryption queries, it additionally simulates random oracle H_2 on its own to obtain $h_2 = H_2(R_1, R_2)$ and then inserts $<M, U, R_1, R_2, R_3, id_S, PK_{id_S}, id_R, PK_{id_R}, h_3, C = M \oplus h_2>$ into H_3List.

CreateUser: $\mathcal{B}$ maintains a list KeyList of tuples $<id, PK_{id}, SK_{id}, flag>$ which is initially empty. On receiving such a query on id, $\mathcal{B}$ does the following:

– If id already appears on KeyList in a tuple $<id, PK_{id}, SK_{id}, flag>$, $\mathcal{B}$ returns PK_{id} to $\mathcal{A}_I$.

– Otherwise, $\mathcal{B}$ randomly chooses $x_{id} \in Z_p^*$ as the private key SK_{id} for the identity id and computes the corresponding public key as $PK_{id} = g^{x_{id}}$. It then inserts $<id, PK_{id}, SK_{id}, 0>$ into KeyList and returns PK_{id} to $\mathcal{A}_I$.

ReplacePublicKey: On receiving such a query on $(id, PK_{id}^{'})$, $\mathcal{B}$ searches id in KeyList to find a tuple $<id, PK_{id}, SK_{id}, flag>$ and updates the resulting tuple with $<id, PK_{id}^{'}, SK_{id}, 1>$.

ExtractPrivateKey: On receiving such a query on id, $\mathcal{B}$ searches id in KeyList to find a tuple $<id, PK_{id}, SK_{id}, flag>$. If $flag = 0$, it returns SK_{id} to $\mathcal{A}_I$; otherwise, it rejects this query.

GenerateCertificate: On receiving such a query on id_i, $\mathcal{B}$ does the following:

– If $(id_i, PK_{id_i}) = (id_\theta, PK_{id_\theta})$, then $\mathcal{B}$ aborts.

– Otherwise, $\mathcal{B}$ searches id_i in H_1List to find a tuple $< id_i, PK_{id_i}, h_{1,i}, Cert_{id_i} >$ and returns $Cert_{id_i}$ to $\mathcal{A}_I$. If H1List does not contain such a tuple, $\mathcal{B}$ queries H_1 on (id_i, PK_{id_i}) first.

Signcryption: On receiving such a query on (M, id_S, id_R), $\mathcal{B}$ does the following:

- If $(id_S, PK_{id_S}) \neq (id_\theta, PK_{id_\theta})$, $\mathcal{B}$ can answer the query according to the specification of the algorithm Signcrypt since it knows the sender id_S's private key and certificate.
- Otherwise, $\mathcal{B}$ randomly chooses r, $h_3 \in Z_p^*$, $h_2 \in \{0, 1\}^n$, and sets $U = r(H_1(id_\theta, PK_{id_\theta})P + P_{pub}) - h_3 SK_{id_\theta}(H_1(id_R, PK_{id_R})P + P_{pub})$, $V = rCert_{id_R}$, $C = M \oplus h_2$, $R_1 = e(U, Cert_{id_R})$ and $R_2 = e(U, Cert_{id_R})^{SK_{id_R}}$ respectively. It is easy to verify that $e(H_1(id_\theta, PK_{id_\theta})P + P_{pub}, V) \cdot (PK_{id_\theta})^{-h_3} = e(U, Cert_{id_R})$. Then, $\mathcal{B}$ inserts $<R_1, R_2, h_2>$ and $<M, U, R_1, R_2, id_\theta, PK_{id_\theta}, id_R, PK_{id_R}, h_3, C>$ into H2List and H3List respectively and returns the ciphertext $\sigma = (C, U, V)$ to $\mathcal{A}_I$. Note that $\mathcal{B}$ fails if H2List or H3List is already defined in the corresponding value but this only happens with probability smaller than $(q_2 + 2q_3 + 2q_{sc})/2^k$.

Designcryption: On receiving such a query on $(\sigma = (C, U, V), id_S, id_R)$, $\mathcal{B}$ does the following:

- If $(id_R, PK_{id_R}) \neq (id_\theta, PK_{id_\theta})$, $\mathcal{B}$ can answer the query according to the specification of the algorithm Designcrypt since it knows the receiver id_R's private key and certificate.
- Otherwise, $\mathcal{B}$ searches in H3List for all tuples of the form $<M, U, R_1, R_2, id_S, PK_{id_S}, id_\theta, PK_{id_\theta}, h_3, C>$. If no such tuple is found, then σ is rejected. Otherwise, each one of them is further examined. For a tuple $<M, U, R_1, R_2, id_S, PK_{id_S}, id_\theta, PK_{id_\theta}, h_3, C>$, $\mathcal{B}$ first checks whether $e(H_1(id_S, PK_{id_S})P + P_{pub}, V) \cdot (PK_{id_S})^{-h_3} = R_1$. If the tuple passes the verification, then $\mathcal{B}$ returns M in this tuple to $\mathcal{A}_I$. If no such tuple is found, σ is rejected. Note that a valid ciphertext is rejected with probability smaller than $q_{dsc}/2^k$ across the whole game.

In the challenge phase, $\mathcal{A}_I$ outputs $(M_0, M_1, id_S^*, id_R^*)$, on which it wants to be challenged. If $(id_R^*, PK_{id_R^*}) \neq (id_\theta, PK_{id_\theta})$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ randomly chooses $C^* \in \{0, 1\}^n$, $r^* \in Z_p^*$ and $V^* \in G_1^*$, sets $U^* = r^*P$, and returns $\sigma^* = (C^*, U^*, V^*)$ to $\mathcal{A}_I$ as the challenge ciphertext. Observe that the decryption of C^* is $C^* \oplus H_2(e(U^*, Cert_{id_\theta}), e(U^*, Cert_{id_\theta})^{SK_{id_\theta}})$.

In the guess phase, $\mathcal{A}_I$ outputs a bit, which is ignored by $\mathcal{B}$. Note that $\mathcal{A}_I$ cannot recognize that σ^* is not a valid ciphertext unless it queries H_2 on $(e(U^*, Cert_{id_\theta}), e(U^*, Cert_{id_\theta})^{SK_{id_\theta}})$ or H_3 on $(M_b, U^*, e(U^*, Cert_{id_\theta}), e(U^*, Cert_{id_\theta})^{SK_{id_\theta}}, id_S^*, PK_{id_S^*}, id_\theta, PK_{id_\theta})$, where $b \in \{0, 1\}$. Standard arguments can show that a successful $\mathcal{A}_I$ is very likely to query H_2 on $(e(U^*, Cert_{id_\theta}), e(U^*, Cert_{id_\theta})^{SK_{id_\theta}})$ or H_3 on $(M_b, U^*, e(U^*, Cert_{id_\theta}), e(U^*, Cert_{id_\theta})^{SK_{id_\theta}}, id_S^*, PK_{id_S^*}, id_\theta, PK_{id_\theta})$ if the simulation is indistinguishable from a real attack environment. To produce a result, $\mathcal{B}$ picks a random tuple $<R_1, R_2, h_2>$ or $<M, U, R_1, R_2, id_S, PK_{id_S}, id_R, PK_{id_R}, h_3, C>$ from H2List or H3List. With probability $1/(q_2 + 2q_3 + 2q_{sc})$ (as H2List, H3List contain at most $q_2 + q_3 + q_{sc}$, $q_3 + q_{sc}$ tuples respectively), the chosen tuple will contain the value $R_1 = e(U^*, Cert_{id_R^*}) = e(r^*P, t(\omega^* + \alpha)^{-1}P) = e(P, P)^{tr^*(\omega^* + \alpha)^{-1}}$. $\mathcal{B}$ then returns $T = R_1^{(tr^*)^{-1}}$ as the solution to the given q-mBDHI problem.

We now derive the advantage of $\mathcal{B}$ in solving the q-mBDHI problem. From the above construction, the simulation fails if any of the following events occurs:

- E_1: $(id_R^*, PK_{id_R^*}) \neq (id_\theta, PK_{id_\theta})$.

– E_2: A GenerateCertificate query is made on $(id_\theta, PK_{id_\theta})$.

– E_3: $\mathcal{B}$ aborts in answer $\mathcal{A}_I$'s Signcryption query because of a collision on H_2 or H_3.

– E_4: $\mathcal{B}$ rejects a valid ciphertext at some point of the game.

We clearly have that $\Pr[\neg E_1] = 1/q_1$ and $\neg E_1$ implies $\neg E_2$. We also already observed that $\Pr[E_3] \le (q_2 + 2q_3 + 2q_{sc})/2^k$ and $\Pr[E_4] \le q_{dsc}/2^k$. Thus, we have that

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

Since $\mathcal{B}$ selects the correct tuple from H$_2$List or H$_3$List with probability $1/(q_2 + 2q_3 + 2q_{sc})$, the advantage of $\mathcal{B}$ in solving the q-mBDHI problem is

$$\varepsilon' \ge \frac{\varepsilon}{q_1(q_2 + 2q_3 + 2q_{sc})}(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k}) .$$

The time complexity of the algorithm $\mathcal{B}$ is dominated by the pairings, exponentiations and scalar multiplications performed in queries. From the above simulation, it is easy to see that the time complexity of $\mathcal{B}$ is bound by $\tau' \le \tau + O(q_1 + q_{sc} + q_{dsc})\cdot \tau_m + O(q_{cu} + q_{sc} + q_{dsc})\cdot \tau_e + O(q_{sc} + q_{dsc})\cdot \tau_p$. □

Lemma 2. *If an IND-CBSC-CCA2-II adversary $\mathcal{A}_{II}$ has advantage ε against our CBSC scheme when running in time τ, asking at most q_{cu} CreateUser queries, q_{sc} Signcryption queries, q_{dsc} Designcryption queries and q_i queries to random oracles H_i* (i = 1, 2, 3), *then there exists an algorithm $\mathcal{B}$ to solve the BDH problem with advantage*

$$\varepsilon' \ge \frac{\varepsilon}{q_{cu}(q_2 + 2q_3 + 2q_{sc})}(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k})$$

and with time $\tau' \le \tau + O(q_{sc} + q_{dsc})\cdot \tau_m + O(q_{cu} + q_{sc} + q_{dsc})\cdot \tau_e + O(q_{sc} + q_{dsc})\cdot \tau_p$ where τ_m, τ_e, τ_p denote the time for computing a scalar multiplication in G, an exponentiation in G_T, and a pairing respectively.

Proof. We show how to construct an algorithm $\mathcal{B}$ to solve the BDH problem from $\mathcal{A}_{II}$. Assume that $\mathcal{B}$ is given a random BDH instance (P, aP, bP, cP), where a, b, c are three random elements from Z_p^*. $\mathcal{B}$ interacts with $\mathcal{A}_{II}$ as follows:

In the setup phase, $\mathcal{B}$ randomly chooses a random $\alpha \in Z_p^*$ as the master key, sets $Q = aP$, and computes $P_{pub} = \alpha P$ and $g = e(P, Q)$. Furthermore, it randomly chooses an index θ with $1 \le \theta \le q_{cu}$. Then, $\mathcal{B}$ starts the game IND-CBSC-CCA2-II by supplying $\mathcal{A}_{II}$ with the master key $msk = \alpha$ and the public parameters $params = \{p, G, G_T, e, n, P, Q, P_{pub}, g, H_1, H_2, H_3\}$, where H_1~H_3 are random oracles controlled by $\mathcal{B}$. $\mathcal{A}_{II}$ can make queries on these random oracles at any time during the game.

Now, $\mathcal{B}$ starts to respond various queries as follows:

H$_1$ Queries: $\mathcal{B}$ maintains a list H$_1$List of tuples $<id, PK_{id}, h_1>$ which is initially empty. On receiving such a query on (id, PK_{id}), $\mathcal{B}$ does the following:

– If (id, PK_{id}) already appears on H$_1$List in a tuple $<id, PK_{id}, h_1>$, then $\mathcal{B}$ returns h_1 to $\mathcal{A}_{II}$.

– Otherwise, it returns a random $h_1 \in Z_p^*$ to $\mathcal{A}_{II}$ and inserts $<id, PK_{id}, h_1>$ into H$_1$List.

H$_2$ Queries: $\mathcal{B}$ responds as in the proof of Lemma 1.

H$_3$ Queries: $\mathcal{B}$ responds as in the proof of Lemma 1.

CreateUser: $\mathcal{B}$ maintains a list KeyList of tuples $< id_i, PK_{id_i}, SK_{id_i} >$ which is initially empty. On receiving such a query on id_i, $\mathcal{B}$ does the following:

- If id_i already appears on KeyList in a tuple $< id_i, PK_{id_i}, SK_{id_i} >$, $\mathcal{B}$ returns PK_{id_i} to $\mathcal{A}_{II}$.
- Else if $id_i = id_\theta$, $\mathcal{B}$ returns $PK_{id_\theta} = e(bP, Q) = e(bP, aP)$ to $\mathcal{A}_{II}$ and inserts $< id_\theta, PK_{id_\theta}, \perp >$ into KeyList. Note that the private key for the identity id_θ is b which is unknown to $\mathcal{B}$.
- Else $\mathcal{B}$ randomly chooses $x_{id_i} \in Z_p^*$ as the private key SK_{id_i} for the identity id_i and computes the corresponding public key as $PK_{id_i} = g^{x_{id_i}}$. It then inserts $< id_i, PK_{id_i}, SK_{id_i} >$ into KeyList and returns PK_{id_i} to $\mathcal{A}_{II}$.

ExtractPrivateKey: On receiving such a query on id_i, $\mathcal{B}$ does the following:

- If $id_i = id_\theta$, then $\mathcal{B}$ aborts.
- Otherwise, $\mathcal{B}$ searches id_i in KeyList to find the tuple $< id_i, PK_{id_i}, SK_{id_i} >$ and returns SK_{id_i} to $\mathcal{A}_{II}$.

Signcryption: On receiving such a query on (M, id_S, id_R), $\mathcal{B}$ does the following:

- If $id_S \neq id_\theta$, $\mathcal{B}$ can answer the query according to the specification of the Signcrypt algorithm since it knows the sender id_S's private key and certificate.
- Otherwise, $\mathcal{B}$ randomly chooses r, $h_3 \in Z_p^*$, $h_2 \in \{0,1\}^n$, and sets $U = r(H_1(id_\theta, PK_{id_\theta})P + P_{pub}) - h_3(H_1(id_R, PK_{id_R})bP + \alpha bP)$, $V = rCert_{id_R}$, $C = M \oplus h_2$ $R_1 = e(U, Cert_{id_R})$ and $R_2 = e(U, Cert_{id_R})^{SK_{id_R}}$ respectively. It is easy to verify that $e(H_1(id_\theta, PK_{id_\theta})P + P_{pub}, V) \cdot (PK_{id_\theta})^{-h_3} = e(U, Cert_{id_R})$. It then inserts $<R_1, R_2, h_2>$ and $<M, U, R_1, R_2, id_\theta, PK_{id_\theta}, id_R, PK_{id_R}, h_3, C>$ into H_2List and H_3List respectively, and returns the ciphertext $\sigma = (C, U, V)$ to $\mathcal{A}_{II}$. Note that $\mathcal{B}$ fails if H_2List or H_3List is already defined in the corresponding value but this only happens with probability smaller than $(q_2 + 2q_3 + 2q_{sc})/2^k$.

Designcryption: $\mathcal{B}$ responds as in the proof of Lemma 1.

In the challenge phase, $\mathcal{A}_{II}$ outputs $(M_0, M_1, id_S^*, id_R^*)$, on which it wants to be challenged. If $id_R^* \neq id_\theta$, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ randomly chooses $C^* \in \{0, 1\}^n$, $V^* \in G_1^*$, sets $U^* = (H_1(id_\theta, PK_{id_\theta}) + \alpha)cP$, and returns $\sigma^* = (C^*, U^*, V^*)$ to $\mathcal{A}_{II}$ as the challenge ciphertext. Observe that the decryption of C^* is $C^* \oplus H_2(e(U^*, Cert_{id_\theta}), U^{*SK_{id_\theta}})$.

In the guess phase, $\mathcal{A}_{II}$ outputs a bit, which is ignored by $\mathcal{B}$. Note that $\mathcal{A}_{II}$ cannot recognize that σ^* is not a valid ciphertext unless it queries H_2 on $(e(U^*, Cert_{id_\theta}), e(U^*, Cert_{id_\theta})^{SK_{id_\theta}})$ or H_3 on $(M_\beta, U^*, e(U^*, Cert_{id_\theta}), e(U^*, Cert_{id_\theta})^{SK_{id_\theta}}, id_S^*, PK_{id_S^*}, id_\theta, PK_{id_\theta})$, where $\beta \in \{0,1\}$. Standard arguments can show that a successful $\mathcal{A}_{II}$ is very likely to query H_2 on $(e(U^*, Cert_{id_\theta}), e(U^*, Cert_{id_\theta})^{SK_{id_\theta}})$ or H_3 on $(M_\beta, U^*, e(U^*, Cert_{id_\theta}), e(U^*, Cert_{id_\theta})^{SK_{id_\theta}}, id_S^*, PK_{id_S^*}, id_\theta, PK_{id_\theta})$ if the simulation is indistinguishable from a real attack environment. To produce a result, $\mathcal{B}$ picks a random tuple $<R_1, R_2, h_2>$ or $<M, U, R_1, R_2, id_S, PK_{id_S}, id_R, PK_{id_R}, h_3, C>$ from H_2List or H_3List. With probability $1/(q_2 + 2q_3 + 2q_{sc})$ (as H_2List, H_3List contain at most $q_2 + q_3 + q_{sc}$, $q_3 + q_{sc}$ tuples respectively), the chosen tuple will contain the right element $R_2 = e(U^*, Cert_{id_\theta})^{SK_{id_\theta}} = e(P, P)^{abc}$. $\mathcal{B}$ then returns R_2 as the solution to the given BDH problem.

We now derive the advantage of $\mathcal{B}$ in solving the BDH problem using the adversary $\mathcal{A}_{II}$. From the above construction, the simulation fails if any of the following events occurs:

- E_1: The challenge receiver's identity id_R^* chosen by $\mathcal{A}_{II}$ does not equal to id_θ.

– E_2: An ExtractPrivateKey query is made on id_θ.

– E_3: $\mathcal{B}$ aborts in answer $\mathcal{A}_{II}$'s Signcryption query because of a collision on H_2 or H_3.

– E_4: $\mathcal{B}$ rejects a valid ciphertext at some point of the game.

We clearly have that $\Pr[\neg E_1] = 1/q_{cu}$ and $\neg E_1$ implies $\neg E_2$. We also already observed that $\Pr[E_3] \le (q_2 + 2q_3 + 2q_{sc})/2^k$ and $\Pr[E_4] \le q_{dsc}/2^k$. Thus, we have that

$$\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4] \ge \frac{1}{q_{cu}}(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k}) .$$

Since $\mathcal{B}$ selects the correct tuple from H2List or H3List with probability $1/(q_2 + 2q_3 + 2q_{sc})$, the advantage of $\mathcal{B}$ in solving the BDH problem is

$$\varepsilon' \ge \frac{\varepsilon}{q_{cu}(q_2 + 2q_3 + 2q_{sc})}(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k}) .$$

From the above simulation, it is easy to see that the time complexity of $\mathcal{B}$ is bound by $\tau' \le \tau + O(q_{sc} + q_{dsc})\cdot\tau_m + O(q_{cu} + q_{sc} + q_{dsc})\cdot\tau_e + O(q_{sc} + q_{dsc})\cdot\tau_p$. □

Theorem 2. *The CBSC scheme above is EUF-CBSC-CMA secure under the hardness of the q-CAA and CDH problems in the random oracle model.*

This theorem can be proved by combining the following two lemmas.

Lemma 3. *Assume that $\mathcal{A}_I$ is an EUF-CBSC-CMA-I adversary that asks at most q_{cu} CreateUser queries, q_{sc} Signcryption queries, q_{dsc} Designcryption queries and q_i queries to random oracles H_i (i = 1, 2, 3). Assume also that $\mathcal{A}_I$ produces a forgery with probability $\varepsilon \ge 10(q_{sc}+1)(q_{sc}+q_3)/2^k$ within a time τ. Then there exists an algorithm $\mathcal{B}$ to solve the q-CAA problem for $q = q_1 - 1$, with advantage*

$$\varepsilon' \ge \frac{\varepsilon}{9q_1}(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k})$$

and with time $\tau' \le 84480q_1q_3[\tau + O(q_1 + q_{sc} + q_{dsc})\cdot\tau_m + O(q_{cu} + q_{sc} + q_{dsc})\cdot\tau_e + O(q_{sc} + q_{dsc})\cdot\tau_p]/[\varepsilon(1 - q_{dsc}/2^k)(1 - q_{sc}(q_2 + 2q_3 + 2q_{sc})/2^k)]$ where τ_m, τ_e, τ_p denote the time for computing a scalar multiplication in G, an exponentiation in G_T, and a pairing respectively.

Proof. We show how to construct an algorithm $\mathcal{B}$ to solve the q-CAA problem from $\mathcal{A}_I$. Assume that $\mathcal{B}$ is given a random q-CAA instance $(P, \alpha P, (\omega_1 + \alpha)^{-1}P, \ldots, (\omega_q + \alpha)^{-1}P, \omega_1, \ldots, \omega_q)$ where α is a random element from Z_p^*. $\mathcal{B}$ interactsg with $\mathcal{A}_I$ as follows:

In the setup phase, $\mathcal{B}$ randomly chooses $t \in Z_p^*$, and sets $P_{pub} = \alpha P$, $Q = tP$ and $g = e(P, Q)$. Furthermore, it randomly chooses a value $\omega^* \in Z_p^*$ such that $\omega^* \notin \{\omega_1, \ldots, \omega_q\}$ and an index $\theta \in [1, q_1]$. Then, $\mathcal{B}$ starts the game EUF-CBSC-CMA-I by supplying $\mathcal{A}_I$ with the public parameters $params = \{p, G, G_T, e, n, P, Q, P_{pub}, g, H_1, H_2, H_3\}$ where H_1~H_3 are random oracles controlled by $\mathcal{B}$. Note that the corresponding master key is $msk = \alpha$ which is unknown to $\mathcal{B}$.

In the query phase, $\mathcal{B}$ responds various oracle queries as in the proof of **Lemma 1**.

Finally, $\mathcal{A}_I$ outputs a valid ciphertext $\sigma^* = (C^*, U^*, V^*)$ from id_S^* to id_R^*. If $(id_S^*, PK_{id_S^*}) \ne (id_\theta, PK_{id_\theta})$, then $\mathcal{B}$ aborts. Otherwise, having the knowledge of $SK_{id_R^*}$ and $Cert_{id_R^*}$, $\mathcal{B}$ queries the oracle Designcryption on $(\sigma^*, id_\theta, PK_{id_\theta}, id_R^*, SK_{id_R^*}, Cert_{id_R^*})$ to obtain M^*, and then queries H_3 on $(M^*, U^*, e(U^*, Cert_{id_R^*}), e(U^*, Cert_{id_R^*})^{SK_{id_R^*}}, id_\theta, PK_{id_\theta}, id_R^*, PK_{id_R^*})$ to obtain h_3^*. Using the oracle replay technique [23], $\mathcal{B}$ replays $\mathcal{A}_I$ with the same random tape but with the different hash value $h_3^{*'} (\ne h_3^*)$ to generates one more valid ciphertext $\sigma^{*'} = (C^*, U^*, V^{*'})$ such that $V^{*'} \ne V^*$.

Since $\sigma^* = (C^*, U^*, V^*)$ and $\sigma^{*'} = (C^*, U^*, V^{*'})$ are both valid ciphertexts for the same message M^* and randomness r^*, we obtain the following relations

$$V^* - V^{*'} = (h_3^* SK_{id_\theta} + r^*)Cert_{id_\theta} - (h_3^{*'} SK_{id_\theta} + r^*)Cert_{id_\theta} = (h_3^* - h_3^{*'})SK_{id_\theta} Cert_{id_\theta} .$$

Because $Cert_{id_\theta} = t(\omega^* + \alpha)^{-1}P$, $\mathcal{B}$ can compute $(\omega^* + \alpha)^{-1}P = [(h_3^* - h_3^{*'})SK_{id_\theta} t]^{-1}(V^* - V^{*'})$ as the solution to the given q-CAA problem.

We now derive the advantage of $\mathcal{B}$ in solving the q-CAA problem. From the above construction, the simulation fails if any of the following events occurs:

- E_1: $(id_S^*, PK_{id_S^*}) \neq (id_\theta, PK_{id_\theta})$.
- E_2: A GenerateCertificate query is made on $(id_\theta, PK_{id_\theta})$.
- E_3: $\mathcal{B}$ aborts in answer $\mathcal{A}_I$'s Signcryption query because of a collision on H_2 or H_3.
- E_4: $\mathcal{B}$ rejects a valid ciphertext at some point of the game.
- E_5: $\mathcal{B}$ fails in using the oracle replay technique to generate one more valid ciphertext.

We clearly have that $\Pr[\neg E_1] = 1/q_1$ and $\neg E_1$ implies $\neg E_2$. We also already observed that $\Pr[E_3] \leq (q_2 + 2q_3 + 2q_{sc})/2^k$ and $\Pr[E_4] \leq q_{dsc}/2^k$. From the forking lemma [23], we know that $\Pr[\neg E_5] \geq 1/9$. Thus, we have that

$$\varepsilon' = \varepsilon \Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4 \wedge \neg E_5] \geq \frac{\varepsilon}{9q_1}(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k}) .$$

Also from the forking lemma [23], if $\varepsilon \geq 10(q_{sc}+1)(q_{sc}+q_3)/2^k$, then the time complexity of $\mathcal{B}$ in solving the given q-CAA problem is bound by $\tau' \leq 84480 q_1 q_3[\tau + O(q_1 + q_{sc} + q_{dsc})\cdot\tau_m + O(q_{cu} + q_{sc} + q_{dsc})\cdot\tau_e + O(q_{sc} + q_{dsc})\cdot\tau_p]/[\varepsilon(1 - q_{dsc}/2^k)(1 - q_{sc}(q_2 + 2q_3 + 2q_{sc})/2^k)]$. □

Lemma 4. *Assume that $\mathcal{A}_{II}$ is an EUF-CBSC-CMA-II adversary that asks at most q_{cu}* CreateUser *queries, q_{sc}* Signcryption *queries, q_{dsc}* Designcryption *queries and q_i queries to random oracles H_i (i = 1, 2, 3). Assume also that $\mathcal{A}_{II}$ produces a forgery with probability $\varepsilon \geq 10(q_{sc}+1)(q_{sc}+q_3)/2^k$ within a time τ. Then there exists an algorithm $\mathcal{B}$ to solve the CDH problem with advantage*

$$\varepsilon' \geq \frac{\varepsilon}{9q_{cu}}(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k})(1 - \frac{q_{dsc}}{2^k})$$

and with time $\tau' \leq 84480 q_{cu} q_3[\tau + O(q_{sc} + q_{dsc})\cdot\tau_m + O(q_{cu} + q_{sc} + q_{dsc})\cdot\tau_e + O(q_{sc} + q_{dsc})\cdot\tau_p]/[\varepsilon(1 - q_{dsc}/2^k)(1 - q_{sc}(q_2 + 2q_3 + 2q_{sc})/2^k)]$ *where τ_m, τ_e, τ_p denote the time for computing a scalar multiplication in G, an exponentiation in G_T, and a pairing respectively.*

Proof. We show how to construct an algorithm $\mathcal{B}$ to solve the BDH problem from $\mathcal{A}_{II}$. Assume that $\mathcal{B}$ is given a random CDH instance (P, aP, bP) where a, b are two random elements from Z_p^*. $\mathcal{B}$ interacts with $\mathcal{A}_{II}$ as follows:

In the setup phase, $\mathcal{B}$ randomly chooses a random $\alpha \in Z_p^*$ as the master key, sets $Q = aP$, and computes $P_{pub} = \alpha P$ and $g = e(P, Q)$. Furthermore, it randomly chooses an index θ with $1 \leq \theta \leq q_{cu}$. Then, $\mathcal{B}$ starts the game EUF-CBSC-CMA-II by supplying $\mathcal{A}_{II}$ with the master key $msk = \alpha$ and the public parameters $params = \{p, G, G_T, e, n, P, Q, P_{pub}, g, H_1, H_2, H_3\}$, where H_1~H_3 are random oracles controlled by $\mathcal{B}$.

In the query phase, $\mathcal{B}$ responds various oracle queries as in the proof of Lemma 2.

Finally, $\mathcal{A}_{II}$ outputs a valid ciphertext $\sigma^* = (C^*, U^*, V^*)$ from id_S^* to id_R^*. If $id_S^* \neq id_\theta$, then $\mathcal{B}$ aborts. Otherwise, having the knowledge of $SK_{id_R^*}$ and $Cert_{id_R^*}$, $\mathcal{B}$ queries the oracle Designcryption on $(\sigma^*, id_\theta, PK_{id_\theta}, id_R^*, SK_{id_R^*}, Cert_{id_R^*})$ to obtain M^*, and then queries H_3 on $(M^*$,

$U^*, e(U^*, Cert_{id_R^*})$, $e(U^*, Cert_{id_R^*})^{SK_{id_R^*}}$, id_θ, PK_{id_θ}, id_R^*, $PK_{id_R^*}$) to obtain h_3^*. Using the oracle replay technique [23], $\mathcal{B}$ replays $\mathcal{A}_{II}$ with the same random tape but with the different hash value $h_3^{*'}$ $(\neq h_3^*)$ to generates one more valid ciphertext $\sigma^{*'} = (C^*, U^*, V^{*'})$ such that $V^{*'} \neq V^*$. Since $\sigma^* = (C^*, U^*, V^*)$ and $\sigma^{*'} = (C^*, U^*, V^{*'})$ are both valid ciphertext for the same message M^* and randomness r^*, we obtain the following relations

$$V^* - V^{*'} = (h_3^* SK_{id_\theta} + r^*)Cert_{id_\theta} - (h_3^{*'} SK_{id_\theta} + r^*)Cert_{id_\theta} = (h_3^* - h_3^{*'})SK_{id_\theta}(H_1(id_\theta, PK_{id_\theta}) + \alpha)^{-1}Q .$$

Then, we have the following relations

$$\begin{aligned} & e(H_1(id_\theta, PK_{id_\theta})P + \alpha P, V^* - V^{*'}) \\ &= e(H_1(id_\theta, PK_{id_\theta})P + \alpha P, (h_3^* - h_3^{*'})SK_{id_\theta}(H_1(id_\theta, PK_{id_\theta}) + \alpha)^{-1}Q) \\ &= e(P, (h_3^* - h_3^{*'})SK_{id_\theta}Q) . \end{aligned}$$

Because $Q = aP$ and $SK_{id_\theta} = b$, $\mathcal{B}$ can compute $abP = SK_{id_\theta}Q = (h_3^* - h_3^{*'})^{-1}(H_1(id_\theta, PK_{id_\theta}) + \alpha)(V^* - V^{*'})$ as the solution to the given CDH problem.

We now derive the advantage of $\mathcal{B}$ in solving the CDH problem. From the above construction, the simulation fails if any of the following events occurs:

– E_1: $\mathcal{A}_{II}$ does not choose id_θ as the challenge sender's identity.
– E_2: An ExtractPrivateKey query is made on id_θ.
– E_3: $\mathcal{B}$ aborts in answer $\mathcal{A}_{II}$'s Signcryption query because of a collision on H_2 or H_3.
– E_4: $\mathcal{B}$ rejects a valid ciphertext at some point of the game.
– E_5: $\mathcal{B}$ fails in using the oracle replay technique to generate one more valid ciphertext.

We clearly have that $\Pr[\neg E_1] = 1/q_{cu}$ and $\neg E_1$ implies $\neg E_2$. We also already observed that $\Pr[E_3] \leq (q_2 + 2q_3 + 2q_{sc})/2^k$ and $\Pr[E_4] \leq q_{dsc}/2^k$. From the forking lemma [23], we know that $\Pr[\neg E_5] \geq 1/9$. Thus, we have that

$$\varepsilon' = \varepsilon \Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4 \wedge \neg E_5] \geq \frac{\varepsilon}{9q_{cu}}\left(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k}\right)\left(1 - \frac{q_{dsc}}{2^k}\right) .$$

Also from the forking lemma [23], if $\varepsilon \geq 10(q_{sc} + 1)(q_{sc} + q_3)/2^k$, then the time complexity of $\mathcal{B}$ in solving the given CDH problem is bound by $\tau' \leq 84480 q_{cu} q_3 [\tau + O(q_{sc} + q_{dsc}) \cdot \tau_m + O(q_{cu} + q_{sc} + q_{dsc}) \cdot \tau_e + O(q_{sc} + q_{dsc}) \cdot \tau_p] / [\varepsilon(1 - q_{dsc}/2^k)(1 - q_{sc}(q_2 + 2q_3 + 2q_{sc})/2^k)]$. □

5.3 Performance Comparison

We next compare our scheme with other existing CBSC schemes in the literature. In the computation cost comparison, we consider four major operations: pairing, exponentiation in G_T, scalar multiplication in G and hash. For simplicity, we denote these operations by p, e, m and h respectively. In the communication cost comparison, ciphertext overhead represents the difference (in bits) between the ciphertext length and the message length, $|id|$ denotes the bit-length of user's identity, and $|G|$ and $|G_T|$ denote the bit length of an element in G and G_T respectively. The performances of the compared CBSC schemes are listed in Table 1.

From the table, we can see that our scheme has better computation efficiency compared with the previous CBSC schemes. Actually, the computation performance of our scheme can be further optimized when $H_1(id, PK_{id})P + P_{pub}$ can be pre-computed. Such a pre-computation enables us to additionally reduce one scalar multiplication computation in G and one hash computation in both the Signcrypt algorithm and the Designcrypt algorithm. Regarding the public key sizes, users' public keys of our scheme lie in G_T and thus have a long representation (typically 1024 bits without optimizations). However, the paring compression techniques due to Barreto and Scott [4] can be used to compress them to a third (say 342 bits) of their original length on supersingular curves in characteristic 3 or even to 1/6 of their original length on BN cures [5]. Those paring compression techniques also can increase the speed of exponentiation computation in G_T. In addition and most importantly, our CBCS scheme is the first signcryption scheme in the certificate-based setting that explicitly achieves insider security and resists key replacement attacks.

Table 1. Performance of our CBSC scheme

Scheme	Insider security?	Secure against key replacement attacks?	Computation cost		Communication cost	
			Signcryption	Designcryption	Ciphertext overhead	Public key size
Ours	Yes		$2e + 3m + 3h$	$2p + 2e + 1m + 3h$	$2\|G\|$	$\|G_T\|$
[20]	No		$1p + 1e + 4m + 3h$	$3p + 1e + 1m + 3h$	$2\|G\|$	$\|G\|$
[14]	Unknown (No Proof)		$1p + 5m + 4h$	$4p + 2m + 3h$	$3\|G\| + \|id\|$	$\|G\|$

6. Conclusions

In this paper, we first introduced a strengthened security model of CBSC that accurately models the key replacement attack and the insider security. Our analysis showed that the CBSC scheme proposed by Luo *et al.* [20] is insecure in our strengthened security model. Furthermore, we proposed a new CBSC scheme that resists key replacement attacks and reaches insider security. Compared with the previous CBSC schemes in the literature, the proposed scheme enjoys better performance, especially in the computation efficiency. However, the security of our scheme is only proved in the random oracle model. Therefore, how to construct a secure CBSC scheme in the standard model becomes an interesting open problem.

References

[1] S.S. Al-Riyami, K.G. Paterson, CBE from CL-PKE: A generic construction and efficient schemes, in: Public Key Cryptography - PKC 2005, LNCS, vol. 3386, Springer-Verlag, 2005, pp. 398-415.

[2] J. An, Y. Dodis, T. Rabin, On the security of joint signature and encryption, in: Advances in Cryptology - Eurocrypt 2002, LNCS, vol. 2332, Springer-Verlag, 2002, pp. 83-107.

[3] M.H. Au, J.K. Liu, W. Susilo, T.H. Yuen, Certificate based (linkable) ring signature, in: Proceedings of the 3rd Information Security Practice and Experience Conference, LNCS, vol. 4464, Springer-Verlag, 2007, pp.79-92.

[4] P.S.L.M. Barreto, M. Scott, Compressed pairings, in: Advances in Cryptology - Crypto 2004, LNCS, vol. 3152, Springer-Verlag, 2004, pp. 140-156.

[5] P.S.L.M. Barreto, M. Naehrig, Pairing-friendly elliptic curves of prime order, in: Proceedings of Selected Areas in Cryptography, 12th International Workshop, LNCS, vol. 3897, Springer-Verlag, 2006, pp. 319-331.

[6] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in: Proceedings of the 1st ACM Conference on Communications and Computer Security, ACM, 1993, pp. 62-73.

[7] D. Boneh, M. Franklin, Identity-based encryption from the Weil pairing, in: Advances in Cryptology - Crypto 2001, LNCS, vol. 2139, Springer-Verlag, 2001, pp.213-229.

[8] R. Canetti, O. Goldreich, and S. Halevi, The random oracle methodology, revisited, Journal of ACM 51(4) (2004) 209-218.

[9] D. Galindo, P. Morillo, C. Ràfols, Improved certificate-based encryption in the standard model, Journal of Systems and Software, 81(7) (2008) 1218-1226.

[10] C. Gentry, Certificate-based encryption and the certificate revocation problem, in: Advances in Cryptology - Eurocrypt 2003, LNCS, vol. 2656, Springer-Verlag, 2003, pp. 272-293.

[11] B.G. Kang, J.H. Park, S.G. Hahn, A certificate-based signature scheme, in: Topics in Cryptology - CT-RSA 2004, LNCS, vol. 2964, Springer-Verlag, 2004, pp. 99-111.

[12] J. Li, X. Huang, Y. Mu, W. Susilo, Q. Wu, Certificate-based signature: security model and efficient construction, in: Proceedings of the 4th European PKI Workshop Theory and Practice, LNCS, vol. 4582, Springer-Verlag, 2007, pp. 110-125.

[13] J. Li, X. Huang, Y. Mu, W. Susilo, Q. Wu, Constructions of certificate-based signature secure against key replacement attacks, Journal of Computer Security, 18(3) (2010) 421-449.

[14] F. Li, X. Xin, Y. Hu. Efficient certificate-based signcryption scheme from bilinear pairings, International Journal of Computers and Applications, 30(2): 129-133, 2008.

[15] J. Li, L. Xu, Y. Zhang, Provably secure certificate-based proxy signature schemes, Journal of Computers, 4(6) (2009) 444-452.

[16] J. K. Liu, J. Zhou, Efficient certificate-based encryption in the standard model, in: Proceedings of the 6th International Conference on Security and Cryptography for Networks, LNCS, vol. 5229, Springer-Verlag, 2008, pp.144-155.

[17] J.K. Liu, J. Baek, W. Susilo, J. Zhou, Certificate based signature schemes without pairings or random oracles,

in: Proceedings of the 11th International conference on Information Security, LNCS, vol. 5222, Springer-Verlag, 2008, pp. 285-297.

[18] Y. Lu, J. Li, J. Xiao, Generic construction of certificate-based encryption, in: Proceedings of the 9th International Conference for Young Computer Scientists, IEEE CS, 2008, pp.1518-1594.

[19] Y. Lu, J. Li, J. Xiao, Constructing efficient certificate-based encryption with paring, Journal of Computers, 4(1) (2009) 19-26.

[20] M. Luo, Y. Wen, H. Zhao, A certificate-based signcryption scheme, in: Proceedings of 2008 International Conference on Computer Science and Information Technology, IEEE CS, 2008, pp. 17-23.

[21] S. Mitsunari, R. Sakai and M. Kasahara, A new traitor tracing, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E85-A(2) (2002) 481-484.

[22] P. Morillo, C. Ràfols, Certificate-based encryption without random oracles, Cryptology ePrint Archive, Report 2006/12. Available at http://eprint.iacr.org/2006/012.pdf.

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

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

[25] S. Sharmila, D. Selvi, S. Sree Vivek, Deepanshu Shukla, Pandu Rangan Chandrasekaran, Efficient and provably secure certificateless multi-receiver signcryption. in: Proceedings of Provable Security 2008, LNCS Vol. 5324, Springer-Verlag, 2008, pp. 52-67.

[26] C. Sur, C.D. Jung, and K.H. Rhee, Multi-receiver certificate-based encryption and application to public key broadcast encryption, in: Proceedings of 2007 ECSIS Symposium on Bio-inspired, Learning, and Intelligent Systems for Security, IEEE CS, 2007, pp. 35-40.

[27] L. Wang, J. Shao, Z. Cao, et al, A certificate-based proxy cryptosystem with revocable proxy decryption power, in: Proceedings of the 8th International Cryptology Conference in India, LNCS, vol. 4859, Springer-Verlag, 2001, pp. 297-311.

[28] W. Wu, Y. Mu, W. Susilo, X. Huang, Certificate-based signatures, revisited, Journal of Universal Computer Science, 15(8) (2009) 1659-1684.

[29] D.H. Yum, P.J. Lee, Separable implicit certificate revocation, In: Proceedings of 2004 International Conference on Information Security and Cryptology, LNCS, vol. 3506, Springer-Verlag, 2005, pp. 121-136.

[30] Y. Zheng, Digital signcryption or how to achieve cost (signature & encryption) << cost (signature) + cost (encryption). In: Advances in Cryptology - Crypto 1997, LNCS, vol. 1294, Springer-Verlag, 1997, pp. 165-179.