Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 295419, 12 pages
http://dx.doi.org/10.1155/2014/295419

Research Article

Efficient Certificate-Based Signcryption Secure against Public Key Replacement Attacks and Insider Attacks

Yang Lu and Jiguo Li

College of Computer and Information Engineering, Hohai University, No. 8, Focheng Xi Road, Jiangning District, Nanjing, Jiangsu 211100, China

Correspondence should be addressed to Yang Lu; luyangnsd@163.com

Received 12 March 2014; Accepted 24 April 2014; Published 12 May 2014

Academic Editor: Tianjie Cao

Signcryption is a useful cryptographic primitive that achieves confidentiality and authentication in an efficient manner. As an extension of signcryption in certificate-based cryptography, certificate-based signcryption preserves the merits of certificate-based cryptography and signcryption simultaneously. In this paper, we present an improved security model of certificate-based signcryption that covers both public key replacement attack and insider security. We show that an existing certificate-based signcryption scheme is insecure in our model. We also propose a new certificate-based signcryption scheme that achieves security against both public key replacement attacks and insider attacks. We prove in the random oracle model that the proposed scheme is chosen-ciphertext secure and existentially unforgeable. Performance analysis shows that the proposed scheme outperforms all the previous certificate-based signcryption schemes in the literature.

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 [1] 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.

To fill the gap between traditional PKC and IBC, Al-Riyami and Paterson [2] proposed a new paradigm called certificateless public key cryptography (CL-PKC) in Asiacrypt 2003. CL-PKC eliminates the key escrow problem inherent in IBC. At the same time, it preserves the advantage of IBC which is the absence of certificates and their heavy management overhead. In CL-PKC, a trusted third party called key generating center (KGC) is involved in the process of issuing a partial secret key for each user. The user independently generates its public/private key pair and combines the partial secret key from the KGC with its private key to generate the actual decryption key. By way of contrast to the PKG in IBC, the KGC does not have access to the user's decryption key. Therefore, CL-PKC solves the key escrow problem. However, as partial secret keys must be sent to the users over secure channels, CL-PKC suffers from the distribution problem.

In Eurocrypt 2003, Gentry [3] introduced the notion of certificate-based cryptography (CBC). CBC provides an implicit certification mechanism for a traditional PKI and allows for a periodical update of certificate status. As in traditional PKC, each user in CBC generates his own public/private key pair and requests a certificate from a trusted

Table 1: Properties of the related public key cryptosystems.

	Do not require trusted third party	Implicit certificates	Key escrow free	Key distribution free
Traditional PKC	×	×	√	√
IBC	×	√	×	×
CL-PKC	×	√	√	×
CBC	×	√	√	√

third party called certifier. The certificate will be pushed only to the owner of the public/private key pair and act as a partial decryption key or a partial signing key. This additional functionality provides an efficient implicit certificate mechanism. For example, in the encryption scenario, a receiver needs both his private key and certificate to decrypt a ciphertext sent to him, while the message sender need not be concerned about the certificate revocation problem. The feature of implicit certification allows us to eliminate third-party queries for the certificate status and simply the public key revocation problem so that CBC does not need infrastructures like CRL and OCSP. Therefore, CBC can be used to construct an efficient PKI requiring fewer infrastructures than the traditional one. Although CBC may be inefficient when a certifier has a large number of users, this problem can be overcome by using *subset covers* [3]. Furthermore, there are no key escrow problem (since the certifier does not know the private keys of users) and key distribution problem (since the certificates need not be kept secret) in CBC.

Table 1 summarizes the comparison of the above cryptosystems.

Since its advent, CBC has attracted great interest in the research community and many schemes have been proposed, including many encryption schemes (e.g., [4–10]) and signature schemes (e.g., [11–16]). As an extension of the signcryption [17] in CBC, Li et al. [18] introduced the concept of certificate-based signcryption (CBSC) that provides the functionalities of encryption and signature simultaneously. As far as we know, there exist three CBSC schemes in the literature so far. In [18], Li et al. proposed the first CBSC scheme based on Chen and Malone-Lee's identity-based signcryption scheme [19]. However, they did not give a formal proof of their security claim. A subsequent paper by Luo et al. [20] proposed the second CBSC scheme alone with a security model of CBSC. Recently, Li et al. [21] proposed a publicly verifiable CBSC scheme that is provably secure in the random oracle model.

Our Motivations and Contributions. In this paper, we focus on the construction of a CBSC scheme that resists both the public key replacement attacks and the insider attacks.

Public key replacement attack was first introduced into CL-PKC by Al-Riyami and Paterson [2]. In this attack, an adversary who can replace a user's public key with a value of its choice dupes any other third parties to encrypt messages or verify signatures using a false public key. It seems that this attack does not have effect on CBC since a certifier is employed for issuing a certificate for each user. Unfortunately, some previous research works [13, 16, 22] have demonstrated that it does. In CBC, the certifier does issue the certificates. However, as introduced above, CBC adopts the implicit certificate mechanism so that only the owner of a certificate needs to check the validity of his certificate and others need not be concerned about the status of his certificate. Thus, a malicious user is able to launch the public key replacement attack against an ill-designed certificate-based cryptographic scheme. We observe that Luo et al.'s CBSC scheme [20] is insecure under this attack. The concrete attack can be found in Section 4 of this paper.

Insider security [23] refers to the security against the attacks made by the insider (i.e., the sender or the receiver). It requires that, even if a sender's private key is compromised, an attacker should not be able to designcrypt the message generated by the sender and, even with a receiver's private key, an attacker should not be able to forge a valid signcryption as if generated by the same sender. In contrast to outsider security [23] that refers to the security against the attacks made by the outsider (i.e., any third party except the sender and the receiver), insider security can provide the stronger security for signcryption schemes [24, 25]. Therefore, it has been accepted as a necessary security requirement for a signcryption scheme to achieve. However, none of the previous constructions of CBSC [18, 20, 21] has considered insider security. The previous security models of CBSC [20, 21] only cover the case where the CBSC scheme is attacked by the outsiders. Actually, the public key replacement attack presented in Section 4 also shows that Luo et al.'s CBSC scheme [20] fails in providing insider security.

The main contributions of this paper are as follows.

(1) We extend previous works by proposing an improved security model for CBSC that accurately models both the public key replacement attacks and the insider attacks. We show that Luo et al.'s CBSC scheme [20] is insecure in our security model.

(2) We develop a new CBSC scheme and formally prove its security in our improved security model. In the random oracle, we prove that the proposed scheme is chosen-ciphertext secure and existentially unforgeable. To the best of our knowledge, it is the first signcryption scheme that achieves security under both the public key replacement attacks and the insider attacks in the certificate-based cryptographic setting. Furthermore, compared with the previous CBSC schemes, our scheme enjoys better performance, especially in the computation efficiency.

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 an improved security model of CBSC. In Section 4, we show that Luo et al.'s CBSC scheme is insecure in our security model. The proposed CBSC scheme is described and analyzed in Section 5. Finally, we draw our conclusions in Section 6.

2. Preliminaries

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

(i) Bilinearity: for all $P_1, P_2 \in G$, and all $a, b \in Z_p^*$, we have $e(aP_1, bP_2) = e(P_1, P_2)^{ab}$.

(ii) Nondegeneracy: $e(P, P) \neq 1$.

(iii) Computability: for all $P_1, P_2 \in G$, $e(P_1, P_2)$ can be efficiently computed.

The security of our CBSC scheme is based on the following hard problems.

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 $abP \in G$.

Definition 2 (see [26]). 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 $e(P, P)^{abc} \in G_T$.

Definition 3 (see [27]). 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 $\omega^* \notin \{\omega_1, \ldots, \omega_q\}$.

Definition 4 (see [28]). 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 $\omega^* \in Z_p^* - \{\omega_1, \ldots, \omega_q\}$.

3. Improved Security Model for CBSC Schemes

In this section, we present an improved security model for CBSC that covers both public key replacement attack and insider security. Below, we first briefly review the definition of CBSC.

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

(i) *Setup*(k): on input a security parameter $k \in Z^+$, this algorithm generates a master key *msk* and a list of public parameters *params*. This algorithm is performed by a certifier. After the algorithm is performed, the certifier publishes *params* and keeps *msk* secret.

(ii) *UserKeyGen*(*params*): on input the public parameters *params*, this algorithm generates a private key and public key pair (SK_U, PK_U) for a user with identity id_U.

(iii) *CertGen*(*params*, *msk*, id_U, PK_U): on input the public parameters *params*, the master key *msk*, a user's identity id_U, and public key PK_U, this algorithm generates a certificate $Cert_U$. This algorithm is performed by a certifier. After this algorithm is performed, the certifier sends the certificate $Cert_U$ to the user id_U via an open channel.

(iv) *Signcrypt*(*params*, M, id_S, PK_S, SK_S, $Cert_S$, id_R, PK_R): on input the public parameters *params*, a sender's identity id_S, public key PK_S, private key SK_S and certificate $Cert_S$, a receiver's identity id_R, and public key PK_R, this algorithm generates a ciphertext σ.

(v) *Designcrypt*(*params*, σ, id_R, PK_R, SK_R, $Cert_R$, id_S, PK_S): on input the public parameters *params*, a ciphertext σ, the receiver's identity id_R, public key PK_R, private key SK_R and certificate $Cert_R$, the sender's identity id_S, and public key PK_S, this algorithm outputs either a plaintext M or a special symbol $\perp$ indicating a designcryption failure.

As introduced in [3], the adversaries against a certificate-based cryptographic scheme should be divided into two types: Type I and Type II. Type I adversary (denoted by A_I) models an uncertified user while Type II adversary (denoted by A_{II}) models an honest-but-curious certifier who is equipped with the master key. In order to capture public key replacement attack, the Type I adversary A_I in our CBSC security model is allowed to replace any user's public key. Note that the Type II adversary A_{II} should not be allowed to make public key replacement attacks; otherwise, it may trivially break the security of a CBSC scheme using a man-in-the-middle attack.

A CBSC scheme should satisfy both confidentiality (i.e., indistinguishability against adaptive chosen-ciphertext attacks (IND-CBSC-CCA2)) and unforgeability (i.e., existential unforgeability against adaptive chosen-messages attacks (EUF-CBSC-CMA)).

The confidentiality security of a CBSC scheme is defined via the following two games: "*IND-CBSC-CCA2 Game-I*" and "*IND-CBSC-CCA2 Game-II*," in which a Type I adversary A_I and a Type II adversary A_{II} interact with a challenger, respectively.

IND-CBSC-CCA2 Game-I. This game is played between A_I and a challenger.

Setup. The challenger runs the algorithm *Setup*(k) to generate *msk* and *params*. It then returns *params* to A_I and keeps *msk* to itself.

Phase 1. In this phase, A_I makes requests to the following oracles adaptively.

(i) $O^{CreateUser}$: on input an identity id_U, if id_U has already been created; the challenger outputs the current public key PK_U associated with id_U. Otherwise, it performs the algorithm *UserKeyGen(params)* to generate a private/public key pair (SK_U, PK_U), inserts (id_U, PK_U, SK_U) into a list, and outputs PK_U. In this case, id_U is said to be created. We assume that other oracles defined below only respond to an identity which has been created.

(ii) $O^{ReplacePublicKey}$: on input an identity id_U and a value PK'_U, the challenger replaces the current public key of the identity id_U with PK'_U. Note that the current value of a user's public key is used by the challenger in any computations or responses to A_I's requests. This oracle models the ability of a Type I adversary to convince a legitimate user to use a false public key and thus enables our security model to capture the public key replacement attacks attempted by the Type I adversary A_I.

(iii) $O^{GenerateCertificate}$: on input an identity id_U, the challenger responds with a certificate $Cert_U$ by running the algorithm *CertGen(params*, *msk*, id_U, PK_U).

(iv) $O^{ExtractPrivateKey}$: on input an identity id_U, the challenger responds with a private key SK_U. Here, A_I is disallowed to query this oracle on any identity for which the 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.

(v) $O^{Signcryption}$: on input a message M, a sender's identity id_S, and a receiver's identity id_R, the challenger responds with $\sigma = Signcrypt(params, M, id_S, PK_S, SK_S, Cert_S, id_R, PK_R)$. 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 A_I to provide it. In addition, we do not consider attacks targeting ciphertexts where the identities of the sender and receiver are the same. So, we disallow queries where $id_S = id_R$.

(vi) $O^{Designcryption}$: 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_R, SK_R, $Cert_R$, id_S, PK_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 A_I to provide it. Again, we disallow queries where $id_S = id_R$.

Challenge. Once 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, computes $\sigma^* = Signcrypt(params, M_b, id^*_S, PK^*_S, SK^*_S, Cert^*_S, id^*_R, PK^*_R)$, and returns σ^* as the challenge ciphertext to A_I.

Phase 2. In this phase, A_I continues to issues queries as in Phase 1.

Guess. Finally, A_I outputs a guess $b' \in \{0, 1\}$. We say that A_I wins the game if $b = b'$ and the following conditions are simultaneously satisfied: (1) A_I cannot query $O^{GenerateCertificate}$ on the identity id^*_R at any point; (2) A_I cannot query $O^{ExtractPrivateKey}$ on an identity if the corresponding public key has been replaced; (3) in Phase 2, A_I cannot query $O^{Designcryption}$ on $(\sigma^*, id^*_S, id^*_R)$ 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 A_I's advantage in this game to be $2|\Pr\{b = b'\} - 1/2|$.

IND-CBSC-CCA2 Game-II. This game is played between A_{II} and a challenger.

Setup. The challenger runs the algorithm *Setup*(k) to generate *msk* and *params*. It then returns *params* and *msk* to A_{II}.

Phase 1. In this phase, A_{II} adaptively asks a polynomial bounded number of queries as in *IND-CBSC-CCA2 Game-I*. The only restriction is that A_{II} cannot replace public keys of any users. In addition, A_{II} need not make any queries to $O^{GenerateCertificate}$ since it can compute the certificates for any identities by itself with the master key *msk*.

Challenge. Once 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, computes $\sigma^* = Signcrypt(params, M_b, id^*_S, PK^*_S, SK^*_S, Cert^*_S, id^*_R, PK^*_R)$, and returns σ^* as the challenge ciphertext to A_{II}.

Phase 2. In this phase, A_{II} continues to issue queries as in Phase 1.

Guess. Finally, A_{II} outputs a guess $b' \in \{0, 1\}$. We say that A_{II} wins the game if $b = b'$ and the following two conditions are both satisfied: (1) A_{II} cannot query $O^{ExtractPrivateKey}$ on the identity id^*_R at any point; (2) A_{II} cannot query $O^{Designcryption}$ on $(\sigma^*, id^*_S, id^*_R)$ in Phase 2. We define A_{II}'s advantage in this game to be $2|\Pr\{b = b'\} - 1/2|$.

Definition 5. A CBSC scheme is said to be IND-CBSC-CCA2 secure if no probabilistic polynomial time (PPT) adversary has nonnegligible advantage in the above two games.

Remark 6. The oracle $O^{ReplacePublicKey}$ defined in the game *IND-CBSC-CCA2 Game-I* models the ability of a Type I adversary to convince a legitimate user to use a false public key. It enables our security model to capture the public key replacement attacks attempted by the Type I adversary A_I.

Remark 7. The adversary in the above 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 [23]. This means that our security model ensures that the confidentiality of signcryption is preserved even if a sender's private key is corrupted.

The unforgeability security of a CBSC scheme is defined via the following two games: "*EUF-CBSC-CMA Game-I*" and "*EUF-CBSC-CMA Game-II,*" in which a Type I adversary A_I and a Type II adversary A_{II} interact with a challenger, respectively.

EUF-CBSC-CMA Game-I. This game is played between A_I and a challenger.

Setup. The challenger runs the algorithm $Setup(k)$ to generate msk and $params$. It then returns $params$ to A_I and keeps msk to itself.

Query. In this phase, A_I can adaptively ask a polynomial bounded number of queries as in the game *IND-CBSC-CCA2 Game-I*.

Forge. Finally, A_I outputs a forgery $(\sigma^*, id_S^*, id_R^*)$. We say that A_I wins the game if the result of $Designcrypt(params, \sigma^*, id_R^*, PK_R^*, SK_R^*, Cert_R^*, id_S^*, PK_S^*)$ is not the $\perp$ symbol and the following conditions are simultaneously satisfied: (1) A_I cannot query $O^{GenerateCertificate}$ on the identity id_S^* at any point; (2) A_I cannot query $O^{ExtractPrivateKey}$ on an identity if the corresponding public key has been replaced; (3) σ^* is not the output of any $O^{Signcryption}$ query on (M^*, id_S^*, id_R^*), where M^* is a message. We define A_I's advantage in this game to be the probability that it wins the game.

EUF-CBSC-CMA Game-II. This game is played between A_{II} and a challenger.

Setup. The challenger runs the algorithm $Setup(k)$ to generate msk and $params$. It then returns $params$ and msk to A_{II}.

Query. In this phase, A_{II} can adaptively ask a polynomial bounded number of queries as in the game *IND-CBSC-CCA2 Game-II*.

Forge. Finally, A_{II} outputs a forgery $(\sigma^*, id_S^*, id_R^*)$. We say that A_{II} wins the game if the result of $Designcrypt(params, \sigma^*, id_R^*, PK_R^*, SK_R^*, Cert_R^*, id_S^*, PK_S^*)$ is not the $\perp$ symbol and the following conditions are simultaneously satisfied: (1) A_{II} cannot query $O^{ExtractPrivateKey}$ on the identity id_S^*; (2) σ^* is not the output of any $O^{Signcryption}$ query on (M^*, id_S^*, id_R^*), where M^* is a message. We define A_{II}'s advantage in this game to be the probability that it wins the game.

Definition 8. A CBSC scheme is said to be EUF-CBSC-CMA secure if no PPT adversary has nonnegligible advantage in the above two games.

Remark 9. The adversary in the above 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 [23]. Hence, our security model also ensures that the unforgeability of signcryption is preserved even if a receiver's private key is corrupted.

4. Cryptanalysis of Luo et al.'s CBSC Scheme

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

4.1. Review of Luo et al.'s CBSC Scheme. Luo et al.'s CBSC scheme consists of the following six algorithms.

(i) *Setup:* given a security parameter k, the certifier performs as follows: generate two cyclic groups G and G_T of prime order p such that there exists a bilinear pairing map $e : G \times G \to G_T$; select a random element $s \in Z_p^*$ and a random generator $P \in G$, and compute $P_{\text{pub}} = sP$; 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 $H_4 : G_T \to \{0,1\}^n$; set the public parameters $params = \{p, G, G_T, e, n, P, P_{\text{pub}}, H_1, H_2, H_3, H_4\}$ and the master key $msk = s$.

(ii) *UserKeyGen:* given $params$, a user with identity $id_U \in \{0,1\}^n$ chooses a random $x_U \in Z_p^*$ as his private key SK_U and then computes his public key $PK_U = x_U P$.

(iii) *CertGen:* to generate a certificate for the user with identity id_U and public key PK_U, the certifier computes $Q_U = H_1(id_U, PK_U)$ and outputs the certificate $Cert_U = sQ_U$.

(iv) *Sender Signcrypt:* to send a message $M \in \{0,1\}^n$ to the receiver id_R, the sender id_S does the following: randomly choose $r \in Z_p^*$ and compute $R = rP$ and $T = H_2(id_S, PK_S, R)$; compute $h = H_3(R, S, M)$ and $V = r^{-1}(Cert_S + SK_S \cdot T + h \cdot P_{\text{pub}})$; compute $W = e(PK_S, PK_R)^r$ and then $C = M \oplus H_4(W)$; set the ciphertext $\sigma = (C, R, V)$.

(v) *Receiver Decrypt:* when receiving a ciphertext $\sigma = (C, R, V)$ from the sender id_S, the receiver id_R does the following: compute $M = C \oplus H_4(W)$ where $W = e(R, SK_R \cdot PK_S)$; forward the message M and signature (R, V) to the algorithm *Receiver Verify*.

(vi) *Receiver Verify:* to verify the sender id_S's signature (R, V) on the message M, the receiver id_R does the following: compute $S = H_2(id_S, PK_S, R)$ and $h = H_3(R, S, M)$; check whether $e(R, V) = e(P_{\text{pub}}, Q_S)e(PK_S, S)e(P, P_{\text{pub}})^h$. If the check holds, output M; otherwise, output $\perp$.

4.2. Attack on Luo et al.'s CBSC Scheme. A Type I adversary who is capable of replacing any user's public key can forge a valid signcryption on any message M from id_S to id_R by performing the following steps.

(1) Replace the sender id_S's public key with $PK'_S = x'_S P_{\text{pub}}$, where x'_S is a random value chosen from Z_p^*.

(2) Choose a random value $r' \in Z_p^*$ and compute $R' = r'P_{\text{pub}}$ and $T' = H_2(id_S, PK'_S, R')$.

(3) Choose a random message $M \in \{0,1\}^n$ and compute $V' = r'^{-1}(Q_S + x'_S T' + h'P)$, where $Q_S = H_1(id_S, PK'_S)$ and $h' = H_3(R', T', M)$.

(4) Randomly choose $C' \in \{0,1\}^n$ and set $\sigma' = (C', R', V')$ as the signcryption of the message M. Note that if the adversary has corrupted the receiver id_R's private key SK_R, it can compute $C' = M \oplus H_4(W')$, where $W' = e(R', SK_R, PK'_S)$.

The ciphertext $\sigma' = (C', R', V')$ passes the verification test as shown below:

$$\begin{aligned} & e\left(P_{\text{pub}}, Q_S\right) e\left(PK'_S, T'\right) e\left(P, P_{\text{pub}}\right)^{h'} \\ &= e\left(P_{\text{pub}}, Q_S + x'_S T' + h'P\right) \\ &= e\left(r'P_{\text{pub}}, r'^{-1}\left(Q_S + x'_S T' + h'P\right)\right) \\ &= e\left(R', V'\right). \end{aligned} \tag{1}$$

This proves that the forged signcryption is valid.

Note that Luo et al.'s scheme also doses not resist insider attacks since the adversary can forge a valid signcryption using the corrupted receiver id_R's private key in the step (4).

5. Our Proposed CBSC Scheme

5.1. Description of the Scheme. Our CBSC scheme is constructed from the certificate-based encryption scheme proposed by Lu et al. [8]. It consists of the following five algorithms.

(i) *Setup*(k): given a security parameter k, the certifier performs the following: generate two cyclic groups G and G_T of a k-bit prime order p such that there exists a bilinear pairing map $e : G \times G \to G_T$; choose two random generators $P, Q \in G$ and compute $g = e(P,Q)$; choose a random element $\alpha \in Z_p^*$ and set $P_{\text{pub}} = \alpha P$; 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; set the public parameters $params = \{p, G, G_T, e, n, P, Q, P_{\text{pub}}, g, H_1, H_2, H_3\}$ and the master key $msk = \alpha$.

(ii) *UserKeyGen*($params$): given $params$, a user with identity $id_U \in \{0,1\}^*$ chooses a random $x_U \in Z_p^*$ as his private key SK_U and then computes his public key $PK_U = g^{x_U}$.

(iii) *CertGen*($params, msk, id_U, PK_U$): to generate a certificate for a user with identity id_U and public key PK_U, the certifier computes $Cert_U = (H_1(id_U, PK_U) + \alpha)^{-1} Q$. The user id_U can check the validness of the certificate $Cert_U$ by verifying whether $e(H_1(id_U, PK_U)P + P_{\text{pub}}, Cert_U) = g$.

(iv) *Signcrypt*($params, M, id_S, PK_S, SK_S, Cert_S, id_R, PK_R$): to send a message $M \in \{0,1\}^n$ to the receiver id_R, the sender id_S does the following: randomly choose $r \in Z_p^*$ and compute $R_1 = g^r$ and $R_2 = (PK_R)^r$; compute $U = r(H_1(id_R, PK_R)P + P_{\text{pub}})$ and $C = M \oplus H_2(R_1, R_2)$; compute $V = (h \cdot SK_S + r) \cdot Cert_S$, where $h = H_3(M, U, R_1, R_2, id_S, PK_S, id_R, PK_R)$; set the ciphertext $\sigma = (C, U, V)$.

(v) *Designcrypt*($params$, σ, id_R, PK_R, SK_R, $Cert_R$, id_S, PK_S): to designcrypt a ciphertext $\sigma = (C, U, V)$ from the sender id_S, the receiver id_R does the following: compute $R_1 = e(U, Cert_R)$ and $R_2 = e(U, Cert_R)^{SK_R}$; compute $M = C \oplus H_2(R_1, R_2)$ and then check whether $e(H_1(id_S, PK_S)P + P_{\text{pub}}, V)(PK_S)^{-h} = R_1$, where $h = H_3(M, U, R_1, R_2, id_S, PK_S, id_R, PK_R)$. If the check holds, output M; otherwise, output $\perp$.

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

(1) $e(U, Cert_R) = e(r(H_1(id_R, PK_R)P + P_{\text{pub}}), (H_1(id_R, PK_R) + \alpha)^{-1} Q) = e(P,Q)^r = g^r$;

(2) $e(U, Cert_R)^{SK_R} = (g^r)^{SK_R} = (PK_R)^r$;

(3)

$$\begin{aligned} & e\left(H_1\left(id_S, PK_S\right)P + P_{\text{pub}}, V\right)\left(PK_S\right)^{-h} \\ &= e\left(\left(H_1\left(id_S, PK_S\right) + \alpha\right)^{-1} P, \left(h \cdot SK_S + r\right) \cdot Cert_S\right) \\ &\quad \cdot \left(PK_S\right)^{-h} \\ &= e\left(P, \left(h \cdot SK_S + r\right) Q\right) \cdot \left(PK_S\right)^{-h} \\ &= e(P,Q)^r = R_1. \end{aligned} \tag{2}$$

5.2. Security Proof

Theorem 10. *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 11. *If a Type I adversary A_I has advantage ε against our CBSC scheme when asking at most q_{cu} to $O^{CreateUser}$ queries, q_{sc} queries to $O^{Signcryption}$, q_{dsc} queries to $O^{Designcryption}$, and q_i queries to random oracles H_1~H_3, then there exists an algorithm B to solve the $(q_1 - 1)$-mBDHI problem with advantage*

$$\begin{aligned} \varepsilon' \geq{}& \frac{\varepsilon}{q_1\left(q_2 + 2q_3 + 2q_{sc}\right)}\left(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k}\right) \\ & \times \left(1 - \frac{q_{dsc}}{2^k}\right). \end{aligned} \tag{3}$$

Proof. Assume that 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 $q = q_1 - 1$. B interacts with A_I as follows.

In the setup phase, B randomly chooses $t \in Z_p^*$ and sets $P_{\text{pub}} = \alpha P$, $Q = tP$, 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, B starts *IND-CBSC-CCA2 Game-I* by supplying A_I with $params = \{p, G, G_T, e, n, P, Q, P_{\text{pub}}, g, H_1, H_2, H_3\}$, where $H_1 \sim H_3$ are random oracles controlled by B. 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 B.

Now, B starts to respond to various queries as follows:

H_1 Queries. We assume that q_1 queries to H_1 are distinct. B maintains a list $\mathsf{H_1List}$ of tuples $\langle id_i, PK_i, h_{1,i}, Cert_i\rangle$. On input (id_i, PK_i), B does the following.

(1) If (id_i, PK_i) already appears on $\mathsf{H_1List}$ in a tuple $\langle id_i, PK_i, h_{1,i}, Cert_i\rangle$, then B returns $h_{1,i}$ to A_I.

(2) Else if the query is on the θth distinct (id_θ, PK_θ), then B inserts $\langle id_\theta, PK_\theta, \omega^*, \perp\rangle$ into $\mathsf{H_1List}$ and returns $h_{1,\theta} = \omega^*$ to A_I. Note that the certificate for the identity id_θ is $Cert_\theta = t(\omega^* + \alpha)^{-1}P$ which is unknown to B.

(3) Else B sets $h_{1,i}$ to be ω_j $(j \in [1, q])$ which has not been used and computes $Cert_i = t(\omega_j + \alpha)^{-1}P$. It then inserts $\langle id_i, PK_i, h_{1,i}, Cert_i\rangle$ into $\mathsf{H_1List}$ and returns $h_{1,i}$.

H_2 Queries. B maintains a list $\mathsf{H_2List}$ of tuples $\langle R_1, R_2, h_2\rangle$. On input (R_1, R_2), B does the following.

(1) If (R_1, R_2) already appears on $\mathsf{H_2List}$ in a tuple $\langle R_1, R_2, h_2\rangle$, B returns h_2 to A_I.

(2) Otherwise, it returns a random $h_2 \in \{0, 1\}^n$ and inserts $\langle R_1, R_2, h_2\rangle$ into $\mathsf{H_2List}$.

H_3 Queries. B maintains a list $\mathsf{H_3List}$ of tuples $\langle M, U, R_1, R_2, id_S, PK_S, id_R, PK_R, h_3, C\rangle$. On input $(M, U, R_1, R_2, id_S, PK_S, id_R, PK_R)$, B does the following.

(1) If $(M, U, R_1, R_2, id_S, PK_S, id_R, PK_R)$ already appears on $\mathsf{H_3List}$ in a tuple $\langle M, U, R_1, R_2, id_S, PK_S, id_R, PK_R, h_3, C\rangle$, B returns h_3 to A_I.

(2) Otherwise, it returns a random $h_3 \in Z_p^*$ to A_I. To anticipate possible subsequent queries to $O^{\text{Designcryption}}$, it additionally simulates the random oracle H_2 on its own to obtain $h_2 = H_2(R_1, R_2)$ and then inserts $\langle M, U, R_1, R_2, id_S, PK_S, id_R, PK_R, h_3, C = M \oplus h_2\rangle$ into $\mathsf{H_3List}$.

$O^{CreateUser}$ Queries. B maintains a list $\mathsf{KeyList}$ of tuples $\langle id_i, PK_i, SK_i, \text{flag}_i\rangle$ which is initially empty. On input (id_i), B does the following.

(1) If id_i already appears on $\mathsf{KeyList}$ in a tuple $\langle id_i, PK_i, SK_i, \text{flag}_i\rangle$, B returns PK_i to A_I directly.

(2) Otherwise, B randomly chooses $x_i \in Z_p^*$ as the private key SK_i for the identity id_i and computes the corresponding public key as $PK_i = g^{x_i}$. It then inserts $\langle id_i, PK_i, SK_i, 0\rangle$ into $\mathsf{KeyList}$ and returns PK_i to A_I.

$O^{ReplacePublicKeyr}$ Queries. On input (id_i, PK_i'), B searches id_i in $\mathsf{KeyList}$ to find a tuple $\langle id_i, PK_i, SK_i, \text{flag}_i\rangle$ and updates the tuple with $\langle id_i, PK_i', SK_i, 1\rangle$.

$O^{ExtractPrivateKey}$ Queries. On input (id_i), B searches id_i in $\mathsf{KeyList}$ to find a tuple $\langle id_i, PK_i, SK_i, \text{flag}_i\rangle$. If $\text{flag}_i = 0$, it returns SK_i to A_I; otherwise, it rejects this query.

$O^{GenerateCertificate}$ Queries. On input (id_i), B does the following.

(1) If $(id_i, PK_i) = (id_\theta, PK_\theta)$, then B aborts.

(2) Otherwise, B searches id_i in $\mathsf{H_1List}$ to find a tuple $\langle id_i, PK_i, h_{1,i}, Cert_i\rangle$ and then returns $Cert_i$ to A_I. If $\mathsf{H_1List}$ does not contain such a tuple, B queries H_1 on (id_i, PK_i) first.

$O^{Signcryption}$ Queries. On input (M, id_S, id_R), B performs as follows.

(1) If $(id_S, PK_S) \neq (id_\theta, PK_\theta)$, 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.

(2) Otherwise, B randomly chooses r, $h_3 \in Z_p^*$, $h_2 \in \{0, 1\}^n$ and sets $U = r(H_1(id_\theta, PK_\theta)P + P_{\text{pub}}) - h_3 SK_\theta(H_1(id_R, PK_R)P + P_{\text{pub}})$, $V = rCert_R$, $C = M \oplus h_2$, $R_1 = e(U, Cert_R)$, and $R_2 = e(U, Cert_R)^{SK_R}$, respectively. It is easy to verify that $e(H_1(id_\theta, PK_\theta)P + P_{\text{pub}}, V) \cdot (PK_\theta)^{-h_3} = e(U, Cert_R)$. Then, B inserts $\langle R_1, R_2, h_2\rangle$ and $\langle M, U, R_1, R_2, id_\theta, PK_\theta, id_R, PK_R, h_3, C\rangle$ into $\mathsf{H_2List}$ and $\mathsf{H_3List}$ respectively, and returns the ciphertext $\sigma = (C, U, V)$ to A_I. Note that B fails if $\mathsf{H_2List}$ or $\mathsf{H_3List}$ is already defined in the corresponding value, but this only happens with probability smaller than $(q_2 + 2q_3 + 2q_{sc})/2^k$.

$O^{Designcryption}$ Queries. On input $(\sigma = (C, U, V), id_S, id_R)$, B does the following.

(1) If $(id_R, PK_R) \neq (id_\theta, PK_\theta)$, 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.

(2) Otherwise, B searches in $\mathsf{H_3List}$ for all tuples of the form $\langle M, U, R_1, R_2, id_S, PK_S, id_\theta, PK_\theta, h_3, C\rangle$. If no such tuple is found, then σ is rejected. Otherwise,

each one of them is further examined. For a tuple $\langle M, U, R_1, R_2, id_S, PK_S, id_\theta, PK_\theta, h_3, C\rangle$, B first checks whether $e(H_1(id_S, PK_S)P + P_{\text{pub}}, V)\cdot(PK_S)^{-h_3} = R_1$. If the tuple passes the verification, then B returns M in this tuple to 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, A_I outputs $(M_0, M_1, id_S^*, id_R^*)$, on which it wants to be challenged. If $(id_R^*, PK_R^*) \neq (id_\theta, PK_\theta)$, then B aborts. Otherwise, B randomly chooses $C^* \in \{0,1\}^n$, $r^* \in Z_p^*$, and $V^* \in G$, computes $U^* = r^*P$, and returns $\sigma^* = (C^*, U^*, V^*)$ to A_I as the challenge ciphertext. Observe that the decryption of C^* is $C^* \oplus H_2(e(U^*, Cert_\theta), e(U^*, Cert_\theta)^{SK_\theta})$.

In the guess phase, A_I outputs a bit which is ignored by B. Note that A_I cannot recognize that σ^* is not a valid ciphertext unless it queries H_2 on $(e(U^*, Cert_\theta), e(U^*, Cert_\theta)^{SK_\theta})$ or H_3 on $(M_b, U^*, (e(U^*, Cert_\theta), e(U^*, Cert_\theta)^{SK_\theta}), id_S^*, PK_S^*, id_\theta, PK_\theta)$, where $b \in \{0,1\}$. Standard arguments can show that a successful A_I is very likely to query H_2 on $(e(U^*, Cert_\theta), e(U^*, Cert_\theta)^{SK_\theta})$ or H_3 on $(M_b, U^*, (e(U^*, Cert_\theta), e(U^*, Cert_\theta)^{SK_\theta}), id_S^*, PK_S^*, id_\theta, PK_\theta)$ if the simulation is indistinguishable from a real attack environment. To produce a result, B picks a random tuple $\langle R_1, R_2, h_2\rangle$ or $\langle M, U, R_1, R_2, id_S, PK_S, id_R, PK_R, h_3, C\rangle$ from $\mathsf{H_2List}$ or $\mathsf{H_3List}$. With probability $1/(q_2 + 2q_3 + 2q_{sc})$ (as $\mathsf{H_2List}$, $\mathsf{H_3List}$ contain at most $q_2 + q_3 + q_{sc}, q_3 + q_{sc}$ tuples, resp.), the chosen tuple will contain the value $R_1 = e(U^*, Cert_\theta)$. Because $e(U^*, Cert_\theta) = e(r^*P, t(\omega^* + \alpha)^{-1}P) = e(P,P)^{tr^*(\omega^*+\alpha)^{-1}}$, B returns $T = R_1^{(tr^*)^{-1}}$ as the solution to the given q-mBDHI problem.

We now derive B's advantage in solving the q-mBDHI problem. From the above construction, the simulation fails if any of the following events occurs: (1) E_1: in the challenge phase, B aborts because $(id_R^*, PK_R^*) \neq (id_\theta, PK_\theta)$; (2) E_2: A_I makes an $O^{GenerateCertificate}$ query on (id_θ, PK_θ); (3) E_3: B aborts in answer one of A_I's $O^{Signcryption}$ queries because of a collision on H_2 or H_3; (4) E_4: 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 $\Pr[E_3] \leq (q_2 + 2q_3 + 2q_{sc})/2^k$ and $\Pr[E_4] \leq q_{dsc}/2^k$. Thus, we have that

$$\Pr\left[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4\right] \geq \frac{1}{q_1}\left(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k}\right) \times \left(1 - \frac{q_{dsc}}{2^k}\right). \tag{4}$$

Since B selects the correct tuple from $\mathsf{H_2List}$ or $\mathsf{H_3List}$ with probability $1/(q_2 + 2q_3 + 2q_{sc})$, we obtain the announced bound on B's advantage in solving the q-mBDHI problem. □

Lemma 12. *If a Type II adversary A_{II} has advantage ε against our CBSC scheme when asking at most q_{cu} queries to $O^{CreateUser}$, q_{sc} queries to $O^{Signcryption}$, q_{dsc} queries to $O^{Designcryption}$, and q_i queries to random oracles $H_1 \sim H_3$, then there exists an algorithm B to solve the BDH problem with advantage*

$$\varepsilon' \geq \frac{\varepsilon}{q_{cu}(q_2 + 2q_3 + 2q_{sc})}\left(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k}\right) \times \left(1 - \frac{q_{dsc}}{2^k}\right). \tag{5}$$

Proof. Assume that B is given a BDH instance (P, aP, bP, cP), where a, b, c are three random elements from Z_p^*. B interacts with A_{II} as follows.

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

Now, B starts to respond various queries as follows.

H_1 Queries. B maintains a list $\mathsf{H_1List}$ of tuples $\langle id_i, PK_i, h_{1,i}\rangle$. On input (id_i, PK_i), B does the following: if (id_i, PK_i) already appears on $\mathsf{H_1List}$ in a tuple $\langle id_i, PK_i, h_{1,i}\rangle$, then B returns $h_{1,i}$ to A_{II}; otherwise, it returns a random $h_{1,i} \in Z_p^*$ and inserts $\langle id_i, PK_i, h_{1,i}\rangle$ into $\mathsf{H_1List}$.

H_2 Queries. B responds as in the proof of Lemma 11.

H_3 Queries. B responds as in the proof of Lemma 11.

$O^{CreateUser}$ Queries. B maintains a list $\mathsf{KeyList}$ of tuples $\langle id_i, PK_i, SK_i\rangle$. On input (id_i), B does the following: (1) if id_i already appears on $\mathsf{KeyList}$ in a tuple $\langle id_i, PK_i, SK_i\rangle$, B returns PK_i to A_{II}. (2) Else if $id_i = id_\theta$, B returns $PK_\theta = e(bP, Q) = e(bP, aP)$ to A_{II} and inserts $\langle id_\theta, PK_\theta, \perp\rangle$ into $\mathsf{KeyList}$. Note that the private key for the identity id_θ is b which is unknown to B. (3) Else B randomly chooses $x_i \in Z_p^*$ as the private key SK_i for the identity id_i and computes the corresponding public key as $PK_i = g^{x_i}$. It then inserts $\langle id_i, PK_i, SK_i\rangle$ into $\mathsf{KeyList}$ and returns PK_i to A_{II}.

$O^{ExtractPrivateKey}$ Queries. On receiving such a query on id_i, B does the following: if $id_i = id_\theta$, then B aborts; otherwise, B searches id_i in $\mathsf{KeyList}$ to find the tuple $\langle id_i, PK_i, SK_i\rangle$ and returns SK_i to A_{II}.

$O^{Signcryption}$ Queries. On input (M, id_S, id_R), B does the following: if $id_S \neq id_\theta$, B can answer the query according to the specification of the $\mathsf{Signcrypt}$ algorithm since it knows the sender id_S's private key and certificate. Otherwise, B randomly chooses $r, h_3 \in Z_p^*$, $h_2 \in \{0,1\}^n$ and computes $U = r(H_1(id_\theta, PK_\theta)P + P_{\text{pub}}) - h_3(H_1(id_R, PK_R)bP + \alpha bP)$, $V = rCert_R$, $C = M \oplus h_2$, $R_1 = e(U, Cert_R)$, and $R_2 = e(U, Cert_R)^{SK_R}$, respectively. It is easy to verify that $e(H_1(id_\theta, PK_\theta)P + P_{\text{pub}}, V)\cdot(PK_\theta)^{-h_3} = e(U, Cert_R)$. It then inserts $\langle R_1, R_2, h_2\rangle$ and $\langle M, U, R_1, R_2, id_\theta, PK_\theta, id_R, PK_R, h_3, C\rangle$

into $\mathsf{H}_2\mathsf{List}$ and $\mathsf{H}_3\mathsf{List}$ respectively, and returns the ciphertext $\sigma = (C, U, V)$ to A_{II}. Note that B fails if $\mathsf{H}_2\mathsf{List}$ or $\mathsf{H}_3\mathsf{List}$ is already defined in the corresponding value, but this only happens with probability smaller than $(q_2 + 2q_3 + 2q_{sc})/2^k$.

$O^{Designcryption}$ *Queries*. B responds as in the proof of Lemma 11.

In the challenge phase, 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 B aborts. Otherwise, B randomly chooses $C^* \in \{0,1\}^n$, $V^* \in G$, computes $U^* = (H_1(id_\theta, PK_\theta) + \alpha)cP$, and returns $\sigma^* = (C^*, U^*, V^*)$ to A_{II} as the challenge ciphertext. Observe that the decryption of C^* is $C^* \oplus H_2(e(U^*, Cert_\theta), e(U^*, Cert_\theta)^{SK_\theta})$.

In the guess phase, A_{II} outputs a bit, which is ignored by B. Note that A_{II} cannot recognize that σ^* is not a valid ciphertext unless it queries H_2 on $(e(U^*, Cert_\theta), e(U^*, Cert_\theta)^{SK_\theta})$ or H_3 on $(M_\beta, U^*, e(U^*, Cert_\theta), e(U^*, Cert_\theta)^{SK_\theta}, id_S^*, PK_S^*, id_\theta, PK_\theta)$, where $\beta \in \{0,1\}$. Standard arguments can show that a successful A_{II} is very likely to query H_2 on $(e(U^*, Cert_\theta), e(U^*, Cert_\theta)^{SK_\theta})$ or H_3 on $(M_\beta, U^*, e(U^*, Cert_\theta), e(U^*, Cert_\theta)^{SK_\theta}, id_S^*, PK_S^*, id_\theta, PK_\theta)$ if the simulation is indistinguishable from a real attack environment. To produce a result, B picks a random tuple $\langle R_1, R_2, h_2\rangle$ or $\langle M, U, R_1, R_2, id_S, PK_S, id_R, PK_R, h_3, C\rangle$ from $\mathsf{H}_2\mathsf{List}$ or $\mathsf{H}_3\mathsf{List}$. With probability $1/(q_2 + 2q_3 + 2q_{sc})$ (as $\mathsf{H}_2\mathsf{List}$, $\mathsf{H}_3\mathsf{List}$ contain at most $q_2 + q_3 + q_{sc}$, $q_3 + q_{sc}$ tuples, resp.), the chosen tuple will contain the right element $R_2 = e(U^*, Cert_\theta)^{SK_\theta} = e(P,P)^{abc}$. B then returns R_2 as the solution to the given BDH problem.

We now derive B's advantage in solving the BDH problem. From the above construction, the simulation fails if any of the following events occurs: (1) E_1: in the challenge phase, B aborts because $id_R^* \neq id_\theta$; (2) E_2: A_{II} makes an $O^{ExtractPrivateKey}$ query on id_θ; (3) E_3: B aborts in answer A_{II}'s $O^{Signcryption}$ query because of a collision on H_2 or H_3; (4) E_4: 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 $\Pr[E_3] \leq (q_2 + 2q_3 + 2q_{sc})/2^k$ and $\Pr[E_4] \leq q_{dsc}/2^k$. Thus, we have that

$$\Pr\left[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4\right] \geq \frac{1}{q_{cu}}\left(1 - q_{sc}\frac{q_2 + 2q_3 + 2q_{sc}}{2^k}\right)\left(1 - \frac{q_{dsc}}{2^k}\right). \quad (6)$$

Since B selects the correct tuple from $\mathsf{H}_2\mathsf{List}$ or $\mathsf{H}_3\mathsf{List}$ with probability $1/(q_2 + 2q_3 + 2q_{sc})$, we obtain the announced bound on B's advantage in solving the BDH problem. □

Theorem 13. *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 14. *If a Type I adversary A_I asks at most q_{cu} queries to $O^{CreateUser}$, q_{sc} queries to $O^{Signcryption}$, q_{dsc} queries to $O^{Designcryption}$, and q_i queries to random oracles H_1~H_3 and produces a valid forgery with probability $\varepsilon \geq 10(q_{sc} + 1)(q_{sc} + q_3)/2^k$, then there exists an algorithm B to solve the $(q_1 - 1)$-CAA problem with advantage $\varepsilon' \geq 1/(9q_1)$.*

Proof. Assume that B is given a q-CAA instance $(P, \alpha P, (\omega_1 + \alpha)^{-1}P, \ldots, (\omega_q + \alpha)^{-1}P, \omega_1, \ldots, \omega_q)$, where $q = q_1 - 1$. B interacts with A_I as follows.

In the setup phase, B randomly chooses $t \in Z_p^*$, sets $P_{\text{pub}} = \alpha P$, and computes $Q = tP$ 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, B starts *EUF-CBSC-CMA Game-I* by supplying A_I with $params = \{p, G, G_T, e, n, P, Q, P_{\text{pub}}, g, H_1, H_2, H_3\}$, where $H_1 \sim H_3$ are random oracles controlled by B. Note that the corresponding master key is $msk = \alpha$ which is unknown to B.

In the query phase, B responds to various oracle queries as in the proof of Lemma 11.

Finally, in the forge phase A_I outputs a valid forgery $(\sigma^* = (C^*, U^*, V^*), id_S^*, id_R^*)$ with probability $\varepsilon \geq 10(q_{sc} + 1)(q_{sc} + q_3)/2^k$ [29]. If $(id_S^*, PK_S^*) \neq (id_\theta, PK_\theta)$, B aborts. Otherwise, having the knowledge of SK_R^* and $Cert_R^*$, B runs the algorithm $Designcrypt(params, \sigma^*, id_R^*, PK_R^*, SK_R^*, Cert_R^*, id_\theta, PK_\theta)$ to obtain the message M^* and then simulates the random oracle H_3 on its own to obtain $h_3^* = H_3(M^*, U^*, e(U^*, Cert_R^*), e(U^*, Cert_R^*)^{SK_R^*}, id_\theta, PK_\theta, id_R^*, PK_R^*)$. Using the oracle replay technique [29], B replays A_I with the same random tape but with the different hash value $h_3^{*\prime} (\neq h_3^*)$ to generate one more valid ciphertext $\sigma^{*\prime} = (C^*, U^*, V^{*\prime})$ such that $V^{*\prime} \neq V^*$. Since $\sigma^* = (C^*, U^*, V^*)$ and $\sigma^{*\prime} = (C^*, U^*, V^{*\prime})$ are both valid ciphertexts for the same message M^* and the randomness r^*, we obtain the following relations:

$$\begin{aligned} V^* - V^{*\prime} &= \left(h_3^* SK_\theta + r^*\right) Cert_\theta - \left(h_3^{*\prime} SK_\theta + r^*\right) Cert_\theta \\ &= \left(h_3^* - h_3^{*\prime}\right) SK_\theta Cert_\theta. \end{aligned} \quad (7)$$

Because $Cert_\theta = t(\omega^* + \alpha)^{-1}P$, B can compute $(\omega^* + \alpha)^{-1}P = [t(h_3^* - h_3^{*\prime})SK_\theta]^{-1}(V^* - V^{*\prime})$ as the solution to the given q-CAA problem.

We now derive B's advantage in solving the q-CAA problem. From the above construction, the simulation fails after A_I outputs a valid forgery if any of the following events occurs: (1) E_1: in the forge phase, B aborts because $(id_S^*, PK_S^*) \neq (id_\theta, PK_\theta)$; (2) E_2: B fails in using the oracle replay technique to generate one more valid ciphertext.

Clearly, $\Pr[\neg E_1] = 1/q_1$. Moreover, from the forking lemma [29], we know that $\Pr[\neg E_2] \geq 1/9$. Thus, we have that if A_I produces a forgery, then B will succeed in solving the q-CAA problem with probability $\varepsilon' = \Pr[\neg E_1 \wedge \neg E_2] \geq 1/(9q_1)$. □

Lemma 15. *If a Type II adversary A_{II} asks at most q_{cu} queries to $O^{CreateUser}$, q_{sc} queries to $O^{Signcryption}$, q_{dsc} queries to $O^{Designcryption}$, and q_i queries to random oracles H_1~H_3 and*

Table 2: Performance of the CBSC schemes.

Schemes	Signcryption cost	Designcryption cost	Ciphertext overhead
Ours	$2e + 3m + 3h$	$2p + 2e + 1m + 3h$	$2\|G\|$
[18]	$1p + 1e + 4m + 3h$	$3p + 1e + 1m + 3h$	$2\|G\|$
[20]	$1p + 5m + 4h$	$4p + 2m + 3h$	$3\|G\| + \|\mathrm{id}\|$
[21]	$2p + 4e + 3m + 3h$	$3p + 4e + 3h$	$2\|G\| + 2\|Z_p\|$

produces a valid forgery with probability $\varepsilon \geq 10(q_{sc} + 1)(q_{sc} + q_3)/2^k$*, then there exists an algorithm B to solve the CDH problem with advantage* $\varepsilon' \geq 1/(9q_{cu})$.

Proof. Assume that B is given a random CDH instance (P, aP, bP) where a, b are two random elements from Z_p^*. B interacts with A_{II} as follows.

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

In the query phase, B responds to various oracle queries as in the proof of Lemma 12.

Finally, in the forge phase A_I outputs a valid forgery $(\sigma^* = (C^*, U^*, V^*), id_S^*, id_R^*)$ with probability $\varepsilon \geq 10(q_{sc} + 1)(q_{sc} + q_3)/2^k$ [29]. If $id_S^* \neq id_\theta$, then B aborts. Otherwise, having the knowledge of SK_R^* and $Cert_R^*$, B runs the algorithm $Designcrypt(params, \sigma^*, id_R^*, PK_R^*, SK_R^*, Cert_R^*, id_\theta, PK_\theta)$ to obtain the message M^* and then simulates the random oracle H_3 on its own to obtain $h_3^* = H_3(M^*, U^*, e(U^*, Cert_R^*), e(U^*, Cert_R^*)^{SK_R^*}, id_\theta, PK_\theta, id_R^*, PK_R^*)$. Using the oracle replay technique [29], B replays A_{II} with the same random tape but with the different hash value $h_3^{*\prime} (\neq h_3^*)$ to generate one more valid ciphertext $\sigma^{*\prime} = (C^*, U^*, V^{*\prime})$ such that $V^{*\prime} \neq V^*$. Since $\sigma^* = (C^*, U^*, V^*)$ and $\sigma^{*\prime} = (C^*, U^*, V^{*\prime})$ are both valid ciphertexts for the same message M^* and randomness r^*, we obtain the following relations:

$$\begin{aligned} V^* - V^{*\prime} &= \left(h_3^* SK_\theta + r^*\right) Cert_\theta - \left(h_3^{*\prime} SK_\theta + r^*\right) Cert_\theta \\ &= \left(h_3^* - h_3^{*\prime}\right) SK_\theta \left(H_1\left(id_\theta, PK_\theta\right) + \alpha\right)^{-1} Q. \end{aligned} \tag{8}$$

Then, we have the following relations:

$$\begin{aligned} &e\left(H_1\left(id_\theta, PK_\theta\right) P + \alpha P, V^* - V^{*\prime}\right) \\ &\quad = e\left(P, \left(h_3^* - h_3^{*\prime}\right) SK_\theta Q\right). \end{aligned} \tag{9}$$

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

We now derive B's advantage in solving the CDH problem. From the above construction, the simulation fails if any of the following events occurs: (1) E_1: in the forge phase, B aborts because $id_S^* \neq id_\theta$; (2) E_2: B fails in using the oracle replay technique to generate one more valid ciphertext. Clearly, $\Pr[\neg E_1] = 1/q_{cu}$. From the forking lemma [29], we know that $\Pr[\neg E_2] \geq 1/9$. Thus, we have that if A_{II} produces a valid forgery, then B will succeed in solving the CDH problem with probability $\varepsilon' = \Pr[\neg E_1 \wedge \neg E_2] \geq 1/(9q_{cu})$. □

Table 3: Timings needed to perform atomic operations and representation of group elements in bits.

Curves	Relative timings (1 unit = 1 scalar multiplication in G)			Representation sizes (bits)	
	m	e	p	$\|G\|$	$\|G_T\|$
MNT/80	1	36	150	171	1026
SS/80	1	4	20	512	1024

5.3. Performance. To evaluate the performance of our new CBSC scheme, we compare our scheme with the previous CBSC schemes in terms of the computational cost and the communicational cost.

In the computational cost comparison, we consider four major operations: pairing, exponentiation in G_T, scalar multiplication in G, and hash. Among these operations, the pairing is considered as the heaviest time-consuming one in spite of the recent advances in the implementation technique. For simplicity, we denote these operations by p, e, m, and h, respectively. In the communicational 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 $|Z_p|$ denote the bit-length of an element in G and Z_p, respectively. Without considering precomputation, the performances of the compared CBSC schemes are listed in Table 2.

The efficiency of a pairing-based cryptosystem always depends on the chosen curve. Boyen [30] computes estimated relative timings for all atomic asymmetric operations (exponentiations and pairings) and representation sizes for group elements when instantiated in supersingular curves with 80-bit security (SS/80) and MNT curves with 80-bit security (MNT/80). In Table 3, we recall the data from [30].

To make a much clearer comparison, Table 4 gives the concrete values of the computational cost and the communicational cost for the compared CBE schemes according to the data in Table 3. As the hash operation is much more efficient than the multiplication in the group G, the costs of the hash operations are ignored.

In our proposed CBSC scheme, the *Signcrypt* algorithm does not require computing any time-consuming pairings. It only needs to compute two exponentiations in G_T, three scalar multiplications in G and three hashes in each signcryption operation. The *Designcrypt* algorithm needs to compute two pairings, two exponentiations in G_T, one

Table 4: Performance comparison of the CBSC schemes.

Schemes	Signcryption cost	Designcryption cost	Ciphertext overhead
MNT/80			
Ours	75	373	342
[18]	187	487	342
[20]	155	602	513 + |id|
[21]	447	594	2390
SS/80			
Ours	11	49	1024
[18]	28	65	1024
[20]	25	82	1536 + |id|
[21]	59	76	3072

scalar multiplication in G, and three hashes to designcrypt a ciphertext. From Tables 2 and 4, we can see that our scheme is more efficient than the previous CBSC scheme, especially in the computational efficiency. Actually, the computational performance of our scheme can be further optimized when $H_1(id_U, PK_U)P + P_{\text{pub}}$ can be precomputed. Such a precomputation enables us to additionally reduce one scalar multiplication computation in G and one hash computation in both the *Signcrypt* algorithm and the *Designcrypt* algorithm. In addition and most importantly, it is believed that our scheme is the first signcryption scheme in the certificate-based cryptographic setting that achieves security against both the public key replacement attacks and the insider attacks.

6. Conclusions

In this paper, we have introduced an improved security model of CBSC that captures both public key replacement attack and insider security. Our cryptanalysis has shown that Luo et al.'s CBSC scheme [20] is insecure in our security model. We have proposed a new CBSC scheme that resists both the key replacement attacks and the insider attacks. Compared with the previous CBSC schemes in the literature, the proposed scheme enjoys better performance, especially in the computation efficiency. However, a limitation of our schemes is that its security can only be achieved in the random oracle model [31]. Therefore, it would be interesting to construct a secure CBSC scheme without random oracles.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the anonymous referees for their helpful comments. This work is supported by the National Natural Science Foundation of China (Grant no. 61272542).

References

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of the Advances in Cryptology (CRYPTO '84)*, pp. 47–53, 1984.

[2] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proceedings of the 9th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT '03)*, pp. 452–473, 2003.

[3] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '03)*, pp. 272–293, 2003.

[4] S. S. Al-Riyami and K. G. Paterson, "CBE from CL-PKE: a generic construction and efficient schemes," in *Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC '05)*, pp. 398–415, January 2005.

[5] P. Morillo and C. Ràfols, "Certificate-based encryption without random oracles," Tech. Rep. 2006/12, Cryptology ePrint Archive, http://eprint.iacr.org/2006/012.pdf.

[6] D. Galindo, P. Morillo, and C. Ràfols, "Improved certificate-based encryption in the standard model," *Journal of Systems and Software*, vol. 81, no. 7, pp. 1218–1226, 2008.

[7] J. K. Liu and J. Zhou, "Efficient certificate-based encryption in the standard model," in *Proceedings of the 6th International Conference on Security and Cryptography for Networks*, pp. 144–155, 2008.

[8] Y. Lu, J. Li, and J. Xiao, "Constructing efficient certificate-based encryption with paring," *Journal of Computers*, vol. 4, no. 1, pp. 19–26, 2009.

[9] Z. Shao, "Enhanced certificate-based encryption from pairings," *Computers and Electrical Engineering*, vol. 37, no. 2, pp. 136–146, 2011.

[10] Y. Lu and J. Li, "Constructing certificate-based encryption secure against key replacement attacks," *ICIC Express Letters B: Applications*, vol. 3, no. 1, pp. 195–200, 2012.

[11] B. G. Kang, J. H. Park, and S. G. Hahn, "A certificate-based signature scheme," in *Proceedings of the Cryptographers' Track at the RSA Conference (CT-RSA '04)*, pp. 99–111, 2004.

[12] M. H. Au, J. K. Liu, W. Susilo, and T. H. Yuen, "Certificate based (linkable) ring signature," in *Proceedings of the 3rd Information Security Practice and Experience Conference*, pp. 79–92, 2007.

[13] J. Li, X. Huang, Y. Mu, W. Susilo, and Q. Wu, "Certificate-based signature: security model and efficient construction," in *Proceedings of the 4th European PKI Workshop*, pp. 110–125, 2007.

[14] J. K. Liu, J. Baek, W. Susilo, and J. Zhou, "Certificate based signature schemes without pairings or random oracles," in *Proceedings of the 11th International conference on Information Security*, pp. 285–297, 2008.

[15] W. Wu, Y. Mu, W. Susilo, and X. Huang, "Certificate-based signatures revisited," *Journal of Universal Computer Science*, vol. 15, no. 8, pp. 1659–1684, 2009.

[16] J. Li, X. Huang, Y. Mu, W. Susilo, and Q. Wu, "Constructions of certificate-based signature secure against key replacement attacks," *Journal of Computer Security*, vol. 18, no. 3, pp. 421–449, 2010.

[17] Y. Zheng, "Digital signcryption or how to achieve cost(signature & encryption) ≪ cost(signature) + cost(encryption)," in *Proceedings of the 17th Annual International Cryptology Conference (CRYPTO '97)*, pp. 165–179, 1997.

[18] F. Li, X. Xin, and Y. Hu, "Efficient certificate-based signcryption scheme from bilinear pairings," *International Journal of Computers and Applications*, vol. 30, no. 2, pp. 129–133, 2008.

[19] L. Chen and J. Malone-Lee, "Improved identity-based signcryption," in *Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC '05)*, pp. 362–379, January 2005.

[20] M. Luo, Y. Wen, and H. Zhao, "A certificate-based signcryption scheme," in *Proceedings of the International Conference on Computer Science and Information Technology (ICCSIT '08)*, pp. 17–23, September 2008.

[21] J. Li, X. Huang, M. Hong, and Y. Zhang, "Certificate-based signcryption with enhanced security features," *Computers and Mathematics with Applications*, 2012.

[22] J. H. Park and D. H. Lee, "On the security of status certificate-based encryption scheme," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E90-A, no. 1, pp. 303–304, 2007.

[23] J. An, Y. Dodis, and T. Rabin, "On the security of joint signature and encryption," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '02)*, pp. 83–107, 2002.

[24] J. Baek, R. Steinfeld, and Y. Zheng, "Formal proofs for the security of signcryption," *Journal of Cryptology*, vol. 20, no. 2, pp. 203–235, 2007.

[25] A. W. Dent, "Hybrid signcryption schemes with insider security," in *Proceedings of the 10th Australasian Conference on Information Security and Privacy (ACISP '05)*, pp. 253–266, July 2005.

[26] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[27] S. Mitsunari, R. Sakai, and M. Kasahara, "A new traitor tracing," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E85-A, no. 2, pp. 481–484, 2002.

[28] S. D. Selvi, S. S. Vivek, D. Shukla, and P. R. Chandrasekaran, "Efficient and provably secure certificateless multi-receiver signcryption," in *Proceedings of the 2nd International Provable Security (ProvSec '08)*, pp. 52–67, 2008.

[29] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.

[30] X. Boyen, "The BB_1 identity-based cryptosystem: a standard for encryption and key encapsulation," IEEE 1363.3, 2006, http://grouper.ieee.org/groups/1363/IBC/submissions/Boyen-bb1_ieee.pdf.

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