IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 22, 2019, accepted September 20, 2019, date of publication September 25, 2019, date of current version October 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2943718

Construction of Certificateless Proxy Signcryption Scheme From CMGs

HUIFANG YU[1] AND ZHICANG WANG[2,3]
[1]School of Cyberspace Security, Xi'an University of Posts and Telecommunications, Xi'an 710121, China
[2]School of Automation, Xi'an University of Posts and Telecommunications, Xi'an 710121, China
[3]School of Computer, Qinghai Normal University, Xining 810008, China

Corresponding author: Huifang Yu (yuhuifang@xupt.edu.cn)

This work was supported in part by the Project of Basic Research of Qinghai Province under Grant 2016-ZJ-776, and in part by the Chunhui Project of Ministry of Education under Grant Z2017052.

ABSTRACT As a cryptography primitive for secure data transmission, certificateless proxy signcryption (CLPS) allows an original signcrypter to entrust his signing authority to a proxy signcrypter for signing specified message on his behalf. In this paper, we combine CLPS with cyclic multiplication groups (CMGs) to construct a new certificateless proxy signcryption scheme from CMGs (CMGs-CLPSS). CMGs-CLPSS will receive significant attention because it simplifies the traditional public key cryptosystem (PKC) and solves the key escrow issue suffered by identity-based public key cryptosystem (IB-PKC). In CMGs-CLPSS, an encrypted message can only be decrypted by a designated receiver who is also responsible for verifying the message; moreover, if a later dispute over repudiation occurs, the designated receiver can readily announce ordinary CLPS for public verification without any extra computation effort. CMGs-CLPSS is proved to have the indistinguishability under adaptive chosen-ciphertext attacks (IND-CCA2 security) and existential unforgeability under adaptive chosen-message attacks (UF-CMA security) in the random oracle model. CMGs-CLPSS outperforms the existing schemes on the basis of computational complexity and is suitable for applications in digital contract signing and online proxy auction, and so on.

INDEX TERMS Certificateless proxy signcryption, cyclic multiplication groups, indistinguishability, existential unforgeability.

I. INTRODUCTION

In traditional PKC, the confidentiality and unforgeability are ensured by first signing the message with a sender's private key and then encrypting message-signature pair using one session key. Subsequently, this session key is encrypted using a receiver's public key before transmission. In decryption and verification phase, the receiver uses his private key to retrieve the session key and then utilizes this session key to decrypt the encrypted message-signature pair. Finally, the receiver confirms the unforgeability of the message by verifying the signature using the public key of the sender. In some occasions for practical applications, simultaneous confidentiality and unforgeability must be satisfied. Signcryption [1]–[9] can simultaneously provide the goals of encrypting and signing the messages, and only allows a designated receiver to recover original message from ciphertext produced by the signer and then to verify the validity of recovered message. Unlike traditional sign-then-encrypt method, signcryption saves computation and bandwidth load.

However, new application demands require the privilege delegation mechanism from time to time to help people entrust their authorities to one person or a group of people in order to accomplish certain work in time, such as online proxy auction, work transfer for deputy or digital contract signing. Conventional PKC cannot satisfy the requirements for new applications in the light of the security robustness and operation efficiency. For satisfying the requirement of applications described above, Mambo *et al.* [10] proposed the first proxy signature which allows an original signer to entrust his signing authority to a proxy signer on his behalf. Only drawback of proxy signature is that it has the authenticity of signature but is unable to assure the confidentiality of message. For this reason, Gamage et al. [11] first introduced the concept of proxy signcryption in 1999 by combining the functionalities of proxy signature and signcryption, in which an original signcrypter entrusts his signcryption rights to a proxy signcrypter, and the latter signcrypts

The associate editor coordinating the review of this manuscript and approving it for publication was Guangjie Han.

specified message on his behalf. Later, Ming *et al.* [12] devised identity-based proxy signcryption scheme in the standard model, its aim is to facilitate confidential transaction with delegation by an authorized proxy. In 2016, Zhou [13] proposed a generalized proxy signcryption from IB-PKC in the standard model, which can resist insider attacks. In 2018, Yu *et al.* [14] proposed identity-based proxy signcryption scheme in universally composable (UC) security framework. Proxy signcryption from IB-PKC overcomes the problems of large amount of computation, communication and storage costs for managing certificates in traditional PKC. However, they still face the key escrow problem in which the private key generator (PKG) creates the private keys of all users and has full supremacy in signing any message or decrypting any ciphertext at the behest of any user.

Fortunately, certificateless public key cryptosystem (CL-PKC) can solve the problem of key escrow in IB-PKC, this is because the key generation center (KGC) in cooperation with user generates the user's full private key. In 2008, Barbosa and Farshim [15] first proposed certificateless signcryption scheme, however, its performance was not acceptable. In 2010, a certificateless signcryption scheme in the standard mode is proposed by Liu *et al.* [16], but it is not secure against malicious KGC attacks [17], [18]. In 2015, certificateless hybrid signcryption [19] and Leakage-free certificateless signcryption [20] are proposed. In 2017, Yu and Yang [21] devised certificateless hybrid signcryption scheme with low complexity. Up to now, there is no proxy signcryption scheme from both CL-PKC and CMGs. Hence, it is an interesting and important research problem how to construct a secure and efficient CMGs-CLPSS suitable for applications in online proxy auction, ubiquitous computing, cloud computing and mobile agents, and so on.

Contributions: In this paper, we provide a construction of certificateless proxy signcryption scheme from CMGs (CMGs-CLPSS). CMGs-CLPSS is IND-CCA2 and UF-CMA secure under the co-CDH and co-DBDH assumptions. By comparison with the previous schemes, we find that CMGs-CLPSS is more efficient in terms of the computation complexity.

Paper Organization: For the remainder of this paper, we organize as follows. Section 2 reviews some hard problems on which the security of CMGs-CLPSS depends. Section 3 provides the descriptions of notations. Section 4 describes the algorithm definition and security models of CMGs-CLPSS. Section 5 constructs a concrete instance of CMGs-CLPSS and verifies its correctness. Section 6 provides the security analysis of confidentiality and unforgeability for CMGs-CLPSS. Section 7 provides a comparison analysis. Finally, we draw the conclusions of this paper in section 8.

II. HARD PROBLEMS

A. BILINEAR PAIRING

Let $< G_1, G_2, G_3 >$ denote three cyclic multiplication groups with prime order p. Let g_1 (resp. g_2) denote a generator of G_1 (resp. G_2), there exists ψ which is an isomorphism from G_2 to G_1, with $\psi(g_2) = g_1$. A map $e : G_1 \times G_2 \rightarrow G_3$ is a bilinear pairing with the properties ① and ② as follows.

① $e\left(g^a, \mathcal{X}^b\right) = e(g, \mathcal{X})^{ab}$ for all $g \in G_1, \mathcal{X} \in G_2$ and $a, b \in Z_p$;

② $e(g_1, g_2) \neq 1$.

B. CO-BDH PROBLEM

Given $< g, g^a, g^b > \in G_1$ and $\mathcal{X} \in G_2$, the co-bilinear Diffie-Hellman (co-BDH) problem is that it is computationally infeasible to determine the value of $e(g, \mathcal{X})^{ab} \in G_3$ for unknown $a, b \in Z_p$.

C. CO-CDH PROBLEM

Given $< g, g^a > \in G_1$ and $\mathcal{X} \in G_2$, the co-computational Diffie-Hellman (co-CDH) problem is that it is computationally infeasible to determine the value of $\mathcal{X}^a \in G_2$ for unknown $a \in Z_p$.

D. CO-DBDH PROBLEM

Given $< g, g^a, g^b > \in G_1$, $\mathcal{X} \in G_2$ and $\mathcal{U} \in G_3$ for unknown $a, b \in Z_p$, the co-decisional bilinear Diffie-Hellman (co-DBDH) problem is to decide whether $e(g, \mathcal{X})^{ab} = \mathcal{U}$. $\mathcal{O}_{co-DBDH}$ oracle outputs 1 if it holds and 0 otherwise.

III. DESCRIPTIONS OF NOTATIONS

See Table 1.

IV. FORMAL DEFINITION OF CMGS-CLPSS

A. ALGORITHM MODEL

A CMGs-CLPSS contains six probabilistic polynomial time (PPT) algorithms as shown in Table 2: **Setup, KeyGen**, **Extract**, **PkeyGen**, **PSigncrypt** and **Unsigncrypt**.

Refer to Table 2, we now describe each algorithm of CMGs-CLPSS.

Setup is run by the key generation center (KGC) which takes a security parameter k as input and outputs a system master key x along with a set of public system parameters $\mathcal{L}$.

KeyGen is run by a user which takes $< \mathcal{L}, I_i >$ as input and outputs the public/private key pair $< y_i, x_i >$ of this user.

Extract is run by the KGC which takes $< \mathcal{L}, I_i, y_i, x >$ as input and outputs the partial private key r_i of this user with identity I_i.

PkeyGen is a proxy key generation algorithm which takes $< \mathcal{L}, m_w, I_a, I_p >$ as input and outputs a proxy key x_{ap}.

PSigncryt is a proxy signcryption algorithm which takes $< \mathcal{L}, m_w, m, I_p, I_b, x_{ap}, y_p, y_b, r_p >$ as input and outputs a ciphertext σ to the receiver with identity I_b.

Unsigcrypt is an unsigncryption algorithm which takes $< \mathcal{L}, m_w, \sigma, I_a, I_p, x_b, r_b, y_p, y_b >$ as input and outputs a message m if the verification equality holds and an error symbol $\perp$ otherwise.

B. SECURITY MODELS

A CMGs-CLPSS must satisfy the IND-CCA2 and UF-CMA security. In our security models, we do not consider these queries where the identities of entities are same.

TABLE 1. Notations and their descriptions.

Notations	Descriptions
p	a large prime, where $p>2^{160}$
G_1	cyclic multiplication
G_2	cyclic multiplication
G_3	cyclic multiplication
I_i	the identity of arbitrary user(i= a,p,b)
I_a	the identity of original signcryter
I_p	the identity of proxy signcryter
I_b	the identity of receiver
y_a	the public key of original signcryter
y_p	the public key of proxy signcryter
y_b	the public key of receiver
x_a	the private key of original signcryter
x_p	the private key of proxy signcryter
x_b	the private key of receiver
r_a	the partial private key of original signcryter
r_p	the partial private key of proxy signcryter
r_b	the partial private key of receiver
m	arbitrary message
σ	ciphertext
⊥	the symbol denoting unsigncryption failure
x	the master key of system
$\mathcal{L}$	a set of system parameters

TABLE 2. Algorithm definition.

Algorithms	input	output
Setup	k	$\mathcal{L}, x$
KeyGen	$\mathcal{L}, I_i$	x_i, y_i
Extract	$\mathcal{L}, I_i, y_i, x$	r_i
PKeyGen	$\mathcal{L}, m_w, I_a, I_p$	x_{ap}
PSigncrypt	$\mathcal{L}, m_w, m, I_p, I_b, y_p, y_b, x_{ap}, r_p$	σ
Unsigncrypt	$\mathcal{L}, m_w, \sigma, I_b, I_p, y_p, y_b, x_b, r_b$	m or ⊥

CMGs-CLPSS can resist the attacks of two types of adversaries A_1 and A_2. As an outsider adversary, A_1 cannot corrupt the master key of the KGC but can replace the public key of arbitrary user in an adaptive method. As an insider adversary, A_2 can corrupt the master key of the KGC but cannot replace the public key of arbitrary user.

1) CONFIDENTIALITY

For the confidentiality of CMGs-CLPSS, we may refer to the IND-CCA2 security model in [19].

In the following, we illustrate the IND-CCA2-I security model of CMGs-CLPSS in terms of an interaction game IND-CMGs-CLPSS-CCA2-I between a challenger C and an adversary A_1.

At the start of the game, C calls **Setup(1^k^)** to obtain the master key x and a set of public system parameters $\mathcal{L}$. Finally, C retains the master key x with itself and returns $\mathcal{L}$ to A_1.

Phase 1. In an adaptive way, A_1 makes a polynomially bounded number of queries.

Request public key: A_1 requests a public key for identity I_i of its choice. C calls **KeyGen** to calculate its public key y_i and returns this public key to A_1.

Private key queries: A_1 requests a private key for identity I_i of its choice. C returns a private key x_i as answer if A_1 has not replaced its public key.

Partial private key queries: A_1 requests a partial private key for identity I_i of its choice. C calls **Extract** to calculate its partial private key r_i and returns r_i to A_1.

Replace public key: A_1 may replace current public key y_i of identity I_i with a random number y'_i.

Proxy key queries: A_1 requests a proxy key for two identities I_a and I_p along with an authorization certificate m_w. C returns a proxy key x_{ap} to A_1 by calling **PkeyGen**.

Proxy signcryption queries: A_1 submits a query of proxy signcryption for the quaternion $< I_b, I_p, m, m_w >$. C runs **PSigncrypt** and returns a ciphertext σ to A_1.

Unsigncryption queries: A_1 submits an unsigncryption query for the quaternion $< I_b, I_p, \sigma, m_w >$. C returns m or $\perp$ to A_1 by calling **Unsigncrypt**.

Challenge. As the first phase is over, A_1 outputs equal-length messages m_0 and m_1 along with $< I_b^*, I_p^*, m_w^* >$. In the first phase, A_1 cannot query the partial private key for identity I_b^* and the public key of this identity should not have been replaced. C selects a random number t from $\{0,1\}$ and obtains σ^* relevant to message m_t. Finally, C returns σ^* as a challenge ciphertext.

Phase 2. In an adaptive fashion, A_1 continues to submit a series of queries as **Phase 1**. A_1 cannot submit a query of partial private key for identity I_b^*. A_1 cannot extract the private key for identity I_b^* if its public key has been replaced before challenge phase. In addition, A_1 cannot submit a query to the unsigncryption oracle for σ^* after challenge phase.

At the end of IND-CMGs-CLPSS-CCA2-I, A_1 outputs t^* as a guess of t. A_1 is said to win IND-CMGs-CLPSS-CCA2-I if $t^* = t$. We define the advantage of A_1 as follows:

$$Adv\,(A_1) = |\Pr[t^* = t] - 1/2| \tag{1}$$

In the following, we elaborate the IND-CCA2-II security model of CMGs-CLPSS on the basis of an interaction game IND-CMGs-CLPSS-CCA2-II between an adversary A_2 and its challenger C.

First of all, C runs **Setup(1^k^)** to generate a set of global parameters $\mathcal{L}$ along with the master key x. Finally, C outputs $< \mathcal{L}, x >$ to A_2.

Phase 1. In an adaptive way, A_2 makes a sequence of polynomially bounded number of queries.

Request public key: A_2 queries a public key for identity I_i of its choice. C runs **KeyGen** and returns its public key y_i.

Full private key queries: A_2 queries a full private key for identity I_i of its choice. C returns its full private key $< r_i, x_i >$ to A_2.

Proxy key queries: A_2 queries a proxy key for $< \mathcal{L}, I_a, I_p >$. C runs **PkeyGen** and returns a proxy key x_{ap} to A_2.

Proxy signcryption queries: A_2 issues a query of proxy signcryption for $< I_b, I_p, m, m_w >$. C calls **PSigncrypt** and returns a ciphertext σ to A_2.

Unsigncryption queries: A_2 issues an unsigncryption query for $< I_b, I_p, \sigma, m_w >$. C executes **Unsigncrypt** and returns $m/\perp$.

Challenge. At the end of **Phase 1**, A_2 outputs same-length messages m_0 and m_1 along with $< I_b^*, I_p^*, m_w^* >$. A_2 cannot extract the private key of identity I_b^* in **Phase 1**. C chooses a random number $t \in\{0,1\}$ and calculates a challenge ciphertext σ^* of message m_t. Then C delivers σ^* to A_2.

Phase 2. In an adaptive fashion, A_2 submits a series of queries as **Phase 1**. A_2 cannot extract the private key of identity I_b^* in **Phase 2** and should not submit a query to unsigncryption oracle for σ^* after challenge phase.

Finally, A_2 outputs t^* as a guess of t and wins IND-CMGs-CLPSS-CCA2-II if $t^* = t$. The advantage of A_2 is defined to be

$$Adv\,(A_2) = |\Pr[t^* = t] - 1/2| \tag{2}$$

Definition 2 (Confidentiality): A CMGs-CLPSS is said to be IND-CCA2 secure if there is no PPT adversary A_1 (resp. A_2) to win IND-CMGs-CLPSS-CCA2-I (resp. IND-CMGs-CLPSS-CCA2-II) with a non-negligible advantage.

2) UNFORGEABLITY

For the unforgeability of CMGs-CLPSS, we may refer to the UF-CMA security model in [19].

In the following, we show the UF-CMA-I security model of CMGs-CLPSS in terms of an interaction game UF-CMGs-CLPSS-CMA-I between a challenger C and an adversary A_1.

First of all, C runs **Setup(1^k)** to obtain the master key x and a set of system parameters $\mathcal{L}$. Finally, C retains the master key x with itself and delivers $\mathcal{L}$ to A_1.

Queries. In an adaptive way, A_1 makes a sequence of polynomially bounded number of queries as **Phase 1** in IND-CMGs-CLPSS-CCA2-I.

Forgery. As the queries are over, A_1 generates a forgery $< I_b^*, I_p^*, \sigma^*, m_w^* >$ to C. A_1 wins UF-CMGs-CLPSS-CMA-I if the result of unsigncryption is valid and the queries are subject to several restrictions as follows: ① A_1 cannot extract the private key of identity I_a^*; ② I_a^* cannot be an identity for which both the partial private key has been extracted and the public key has been replaced; ③ $< I_b^*, I_p^*, m_w^*, \sigma^* >$ should not be returned by the proxy signcryption oracle.

A_1's advantage is defined as the probability that it wins UF-CMGs-CLPSS-CMG-I.

In the following, we expound the UF-CMA-II security model of CMGs-CLPSS on the basis of an interaction game UF-CMGs-CLPSS-CMA-II between an adversary A_2 and its challenger C.

At the beginning of the game, C runs **Setup(1^k)** to achieve the master key x and a set of system parameters $\mathcal{L}$. Then C outputs $< \mathcal{L}, x >$ to A_2.

Queries. A_2 adaptively submits a polynomially bounded number of queries as **Phase 1** in IND-CMGs-CLPSS-CCA2-II.

Forgery. At the end of queries, A_2 outputs a forgery $< I_b^*, I_p^*, \sigma^*, m_w^* >$ to C. A_2 cannot query the private key of identity I_a^* and $< I_b^*, I_p^*, m_w^*, \sigma^* >$ should not be returned by proxy signcryption oracle. A_2 wins UF-CMGs-CLPSS-CMA-II if the result of unsigncryption is valid.

A_2's advantage is defined as the probability that it wins UF-CMGs-CLPSS-CMA-II.

Definition 2 (Unforgeability): A CMGs-CLPSS is said to be secure with respect to UF-CMA if there is no PPT adversary A_1 (resp. A_2) to win UF-CMGs-CLPSS-CMA-I (resp. UF-CMGs-CLPSS-CMA-II) with a non-negligible advantage.

V. AN EXAMPLE OF CMGS-CLPSS

In this section, we construct a concrete instance of CMGs-CLPSS.

A. SETUP

The KGC chooses a security parameter k and executes this algorithm as follows:

① select three cyclic multiplication groups G_1, G_2 and G_3 with prime order p and a bilinear map $e : G_1 \times G_2 \to G_3$, please keep in mind that g is a generator of G_1;

② select a random number x from Z_p as the system master key and determine the system public key y by the equality (3):

$$y = g^x \in G_1 \tag{3}$$

③ define four collision-resistant hash functions (n_1 is the length of authorization certificate and n_2 is the length of message):

$$H_0 : G_1 \times G_1 \times \{0, 1\}^* \to G_2,$$
$$H_1 : \{0, 1\}^{n_1} \times G_1 \times G_3 \to \{0, 1\}^{n_2},$$
$$H_2 : \{0, 1\}^{n_1+n_2} \times G_1 \times G_1 \times G_1 \times G_3 \to G_2,$$
$$H_3 : \{0, 1\}^{n_1} \times G_1 \to G_2;$$

④ retain the master key x with itself and publish the system parameters named $\mathcal{L}$, where

$$\mathcal{L} =< p, G_1, G_2, G_3, g, y, n_1, n_2, H_0, H_1, H_2, H_3 > \tag{4}$$

B. KEYGEN

In this key generation algorithm, a user with identity I_i ($i = a$, p, b) selects a random number $x_i \in Z_p$ as his private key and determines his public key y_i by the equality (5):

$$y_i = g^{x_i} \in G_1 \tag{5}$$

C. EXTRACT

In this extraction algorithm, the KGC carries out the following steps:

① calculate $\mathcal{H}_i = H_0(y, y_i, I_i) \in G_2$;
② calculate $r_i = \mathcal{H}_i^x \in G_2$;
③ deliver r_i to a user with identity I_i.

After receiving r_i, this user verifies the legality of the partial private key r_i by computing the values at both sides of the equality (6):

$$e(g, r_i) = e(y, \mathcal{H}_i) \tag{6}$$

Please keep in mind that $< r_i, x_i >$ $(i = a, p, b)$ is the full private key of the user with identity I_i $(i = a, p, b)$.

D. PKEYGEN

In this proxy key generation algorithm, an original signcrypter **A** with identity I_a generates an authorization certificate m_w, where m_w includes explicit description of delegation relation and the restriction of usage to a proxy signcrypter. Then **A** calculates V by the equality (7):

$$V = H_3^{x_a}(m_w, y_a) \in G_2 \tag{7}$$

and outputs $< V, m_w >$ to the proxy signcrypter **P** with identity I_p. If the equality $e(g, V) = e\left(y_a, H_3\left(m_w, y_a\right)\right)$ holds, **P** calculates the proxy key x_{ap} by the equality (8):

$$x_{ap} = V\mathcal{H}_p^{x_p} \in G_2 \tag{8}$$

E. PROXY SIGNCRYPTION

In this proxy signcryption algorithm, **P** chooses a random number $\mu \in Z_p$ and calculates r by the equality $r = g^{\mu} \in G_1$. Then **P** continues to carry out the following steps:

① calculate $\rho = e(y_b y, H_b)^{\mu} \in G_3$;
② calculate $c = H_1(m_w, r, \rho) \oplus m$;
③ calculate $\phi = H_2\left(m_w, m, r, y_p, y_b, \rho\right) \in G_2$;
④ calculate $s = x_{ap} r_p \phi^{\mu} \in G_2$;
⑤ output $\sigma =< r, c, s >$ the receiver **B** with identity I_b.

F. UNSIGNCRYPTION

After receiving $\sigma =< r, c, s >$, **B** computes $< \rho, m, \phi >$ by the equalities (9)~(11):

$$\rho = e\left(r, r_b H_b^{x_b}\right) \in G_3 \tag{9}$$
$$m = H_1(m_w, r, \rho) \oplus c \tag{10}$$
$$\phi = H_2\left(m_w, m, r, y_p, y_b, \rho\right) \in G_2 \tag{11}$$

Then **B** checks whether the verification equality (12) holds as follows:

$$e(g, s) = e(y_a, H_3(m_w, y_a))\, e\left(yy_p, \mathcal{H}_p\right) e(r, \phi) \tag{12}$$

If the equality (12) holds, **B** accepts the message m and rejects it otherwise.

Correctness of CMGs-CLPSS can be proved by the equalities (13) and (14):

$$\begin{aligned}\rho &= e(yy_b, \mathcal{H}_b)^{\mu}\\ &= e\left(y, \mathcal{H}_b^{\mu}\right) e\left(y_b, \mathcal{H}_b^{\mu}\right)\\ &= e\left(r, \mathcal{H}_b^{x}\right) e\left(r, \mathcal{H}_b^{x_b}\right)\\ &= e\left(r, r_b \mathcal{H}_b^{x_b}\right)\end{aligned} \tag{13}$$

$$\begin{aligned}e(g, s) &= e\left(g, x_{ap} r_p \phi^{\mu}\right)\\ &= e\left(g, x_{ap} r_p\right) e\left(g, \phi^{\mu}\right)\\ &= e\left(g, V\mathcal{H}_p^{x_p}\mathcal{H}_p^{x}\right) e\left(g^{\mu}, \phi\right)\\ &= e(y_a, H_3(m_w, y_a))\, e\left(yy_p, \mathcal{H}_p\right) e(r, \phi)\end{aligned} \tag{14}$$

VI. SECURITY PROOF

A. CONFIDENTIALITY

Theorem 1: If a PPT adversary A_1 can break the IND-CCA2-I security of CMGs-CLPSS with an advantage $\mathcal{E}$ in the random oracle model by making l_i queries to the H_i $(i = 0, 1, 2, 3)$ oracle, $l_{p'}$ queries to the partial private key oracle, l_p queries to the private key oracle, l_r queries to the public key replacement oracle, and l_{ap} queries to the proxy key oracle, there exists an algorithm C which can solve the co-DBDH problem with an advantage

$$\varepsilon' \geq \frac{\varepsilon}{l_1 e\left(l_p + l_{p'} + l_{ap} + l_r\right)} \tag{15}$$

where e is the base of natural logarithm.

Proof: Assume C receives a co-DBDH problem instance $< g, C_1 = g^a, C_2 = g^b, \mathcal{X}, \mathcal{U} >$ and its aim is to calculate $\mathcal{U} = e(g, \mathcal{X})^{ab}$. For this aim, C runs the adversary A_1 as a subroutine and plays the role of its challenger in the whole game.

C maintains six lists $< L_0, L_1, L_2, L_3, L_k, L_{ap} >$ which are initially empty, and these lists are made use of tracing the H_0 oracle, H_1 oracle, H_2 oracle, H_3 oracle, public key oracle and proxy key oracle, respectively. C chooses an integer $j \in \{1, 2, \ldots, l_0\}$ and considers I_j as the challenge identity. Let δ denote the probability of $I_i = I_j$ and the value of δ will be determined later.

First of all, C calls **Setup(1^k)** to obtain a set of system parameters $\mathcal{L}$ with $y = C_1$ and then outputs $\mathcal{L}$ to A_1.

Phase 1. In an adaptive way, A_1 issues a series of polynomially bounded number of queries.

H_0 queries: A_1 submits an H_0 oracle query. C outputs H_i to A_1 if there is a relevant tuple in the list L_0; otherwise, C considers two cases to deal with this H_0 query:

Case 1: If it is the jth query, C first sets $\mathcal{H}_i = \mathcal{X}$. Then C outputs $\mathcal{H}_i$ to A_1 and stores $< y, y_i, I_i, \mathcal{H}_i, - >$ into the list L_0.

Case 2: If it is not the jth query, C calculates $\mathcal{H}_i = g^{\lambda}$ by using a random number $\lambda \in Z_p$ of its choice. Then C stores $< y, y_i, I_i, \mathcal{H}_i, \lambda >$ into the list L_0 and delivers $\mathcal{H}_i$ to A_1.

H_1 queries: A_1 submits an H_1 oracle query. C outputs f to A_1 if there is a matching tuple $< m_w, r, \rho, \upsilon >$ in the list L_1; otherwise, C returns $\upsilon \in \{0,1\}^{n_2}$ of its choice and adds $< m_w, r, \rho, \upsilon >$ into the list L_1.

H_2 queries: A_1 submits an H_2 oracle query. C outputs ϕ to A_1 if there is a related tuple in the list L_2; otherwise, it is necessary for C to consider two cases in response to this query.

Case 1: If it is the jth query, C outputs $\phi = \mathcal{X}$ to A_1 and stores $< m_w, m, r, y_p, y_b, \rho, \phi >$ into the list L_2.

Case 2: If it is not the jth query, C first sets $\phi = \mathcal{H}_p = g^{\lambda}$. Then C stores $< m_w, m, r, y_p, y_b, \rho, \phi >$ into the list L_2 and returns ϕ to A_1.

H_3 queries: A_1 submits a query to the H_3 oracle. C delivers ζ to A_1 if there exists a matching tuple in the list L_3; otherwise, C returns a random number $\zeta \in G_2$ of its choice and stores $< m_w, y_a, \zeta >$ to the list L_3.

Request public key: A_1 submits a public key query for identity I_i of its choice. C outputs the public key y_i to A_1 if it exists in the list L_k; otherwise, C chooses a random number $x_i \in Z_p$ to calculate $y_i = g^{x_i}$. Then C outputs the public key y_i to A_1 and stores $< I_i, x_i, y_i, - >$ into the list L_k.

Partial private key queries: A_1 submits a partial private key query for identity I_i of its choice. C fails and stops this game if it is the jth query; otherwise, C first calls the H_0 oracle and public key oracle, then it updates the list L_k with $< I_i, x_i, y_i, r_i = y^{\lambda} >$ and returns a partial private key r_i to A_1. A_1 can verify the legality of the partial private key by the equality (16):

$$e\left(g, r_i\right) = e\left(y, \mathcal{H}_i\right) \tag{16}$$

Private key queries: A_1 submits a private key query for identity I_i of its choice. C fails and stops this game if it is the jth query; otherwise, C calls the public key oracle along with partial private key oracle and then returns the private key $< x_i, r_i >$ to A_1.

Replace public key: A_1 wants to replace the public key y_i of identity I_i with a random y_i' of its choice. If it is the jth query, C fails and stops this game; otherwise, C replaces the public key y_i with y_i' and updates the list L_k with $< I_i, -, y_i, r_i >$.

Proxy key queries: A_1 submits a proxy key query for $< I_a, I_p, m_w >$. If $I_a = I_j$, C fails and stops this game; otherwise, C first calls the H_0 oracle along with public key oracle and calculates $V = \zeta^{x_a}$. Then C verifies whether the equality $e\left(g, V\right) = e\left(y_a, \zeta\right)$ holds. If it holds, C calculates the proxy signcryption key x_{ap} by the equality (17):

$$x_{ap} = V\mathcal{H}_p^{x_p} \in G_2 \tag{17}$$

Finally, C outputs x_{ap} to A_1 and stores $< V, x_{ap}, m_w >$ into the list L_{ap}.

Proxy signcryption queries: A_1 submits a query of proxy signcryption for the quaternion $< I_b, I_p, m_w, m >$. C first calls the H_0 and H_3 oracles along with the public key oracle and (partial) private key oracle, then it considers two cases in response to this proxy signcryption query.

Case 1: If $I_p \neq I_j$, C outputs a ciphertext σ to A_1 by a call to actual proxy signcryption algorithm.

Case 2: If $I_p = I_j$, C chooses a random number $\mu \in Z_p$ and sets $r = g^{\mu}\left(yy_p\right)^{-1}$. Then C continues to calculate $< \rho, c, \phi, s >$ by the equalities (18)~(21).

$$\rho = e\left(r, r_b\mathcal{H}_b^{x_b}\right) \in G_3 \tag{18}$$
$$c = \upsilon \oplus m \in \{0,1\}^{n_2} \tag{19}$$
$$\phi = \mathcal{X} \in G_2 \tag{20}$$
$$s = \zeta^{x_a}\mathcal{H}_p\phi^{\mu} \tag{21}$$

Finally, C stores $< m_w, m, r, y_a, y_p, \rho, \phi >$ into the list L_2 and outputs $\sigma =< r, c, s >$ to A_1.

A_1 can readily verify the validity of $\sigma =< r, c, s >$ by the equality (22):

$$\begin{aligned} e\left(y_a, H_3\left(m_w, y_a\right)\right) e\left(yy_p, \mathcal{H}_p\right) e\left(r, \phi\right) &= e\left(y_a, \zeta\right) e\left(yy_p, \mathcal{H}_p\right) e\left(r, \phi\right) \\ &= e\left(y_a, \zeta\right) e\left(g, \mathcal{H}_p\phi^{\mu}\right) \\ &= e\left(g, \zeta^{x_a}\mathcal{H}_p\phi^{\mu}\right) \\ &= e\left(g, s\right) \end{aligned} \tag{22}$$

Unsigncryption queries: A_1 submits an unsigncryption query for the quaternion $< I_b, I_p, m_w, \sigma >$. Assume A_1 has queried various oracles before unsigncryption query. It is needful for C to consider two cases in response to this unsigncryption query.

Case 1: If $I_b \neq I_j$, C outputs a result to A_1 by a call to actual unsigncryption algorithm.

Case 2: If $I_b = I_j$, C checks the list L_1 to seek a tuple $< m_w, r, \rho, \upsilon >$ such that $\mathcal{O}_{co\text{-}DBDH}$ returns 1 when A_1 queried for $< y, r, \mathcal{X}, \mathcal{U} >$. If this case occurs, C calculates $< \rho, m >$ by the equalities (23) and (24):

$$\rho = e\left(r, \mathcal{X}^{x_b}\right) \cdot \mathcal{U} \tag{23}$$
$$m = \upsilon \oplus c \in \{0,1\}^{n_2} \tag{24}$$

Then C calculates $\phi = H_2\left(m_w, m, r, y_a, y_p, \rho\right)$ and checks whether $e\left(g, s\right) = e\left(y_a, \zeta\right) e\left(yy_p, \mathcal{H}_p\right) e\left(r, \phi\right)$ holds. C outputs m to A_1 if the verification is true and $\perp$ otherwise.

Challenge. After a sequence of polynomially bounded number of the queries, A_1 outputs $< m_0, m_1 >\in \{0, 1\}^{n_2}$ along with $< I_b^*, I_p^*, m_w^* >$ to C. In **Phase 1**, A_1 cannot extract the private key of identity I_b. In addition, I_b^* should not be this identity for which the public key has been replaced and the partial private key has been extracted. C has queried the H_0 and H_3 oracles along with the public key oracle and (partial) private key oracles. C has to consider two cases in response to this challenge query.

Case 1: If $I_b^* \neq I_j$, C fails and stops this game.

Case 2: If $I_b^* = I_j$, C randomly picks t from {0,1} and $\mathcal{U}^* \in G_3$. Then C sets $r^* = C_2$ and continues to calculate $< \rho^*, \upsilon^*, c^* >$ by the equalities (25)~(27):

$$\rho^* = e\left(r^*, \mathcal{X}^{x_b*}\right) \cdot \mathcal{U} \tag{25}$$
$$\upsilon^* = H_1\left(m_w^*, r^*, \rho^*\right) \tag{26}$$
$$c^* = \upsilon^* \oplus m_t \in \{0,1\}^{n_2} \tag{27}$$

where x_b^* is from either the adversary A_1 or the list L_k. C stores $< m_w^*, r^*, \rho^*, \upsilon^* >$ into the list L_1 and then calculates $< \phi^*, s^* >$ by the equalities (28) and (29):

$$\phi^* = H_2\left(m_w^*, m_t, r^*, y_a^*, y_p^*, \rho^*\right) \tag{28}$$

$$s^* = r_a^*\left(r^* y_a^*\right)^{\lambda} \tag{29}$$

Finally, C stores $< m_w^*, m_t, r^*, y_a^*, y_p^*, \rho^*, \phi^* >$ into the list L_3 and outputs challenge ciphertext $\sigma^* =< r^*, c^*, s^* >$.

Phase 2. In an adaptive method, A_1 continues to submit a polynomially bounded number of queries as those in **Phase 1**. A_1 cannot query the private key of identity I_b^* in **Phase 2**. A_1 cannot extract the partial private key of identity I_b^* if its public key has been replaced before challenge phase. In addition, A_1 cannot submit a query to the unsigncryption oracle for $\sigma^* =< r^*, c^*, s^* >$ after challenge phase.

Guess. At the end of the game, A_1 outputs a guess t^* of t. If $t^* = t$, C outputs the solution of co-DBDH problem instance by the equality (30):

$$\mathcal{U} = e\left(C_2, \mathcal{X}\right)^a = e\left(y, \mathcal{X}\right)^b = e\left(g, \mathcal{X}\right)^{ab} \tag{30}$$

In other words, C makes use of the adversary A_1 to solve the co-DBDH problem.

Probability analysis. As described above, l_p is the times of querying the private key oracle, $l_{p'}$ is the times of querying the partial private key oracle, l_{ap} is the times of querying the proxy key oracle, and l_r is the times of the public key replacement. As shown in [8], the probability of C not failing in **Phase** 1 **or 2** is $\delta^{l_p+l_{p'}+l_{ap}+l_r}$, and the probability of C not failing in challenge phase is $1-\delta$. Hence, the probability of C not stopping the execution of game is $\delta^{l_p+l_{p'}+l_{ap}+l_r}(1-\delta)$, this value is maximized at

$$\delta = 1 - \frac{1}{1 + l_p + l_{p'} + l_{ap} + l_r} \tag{31}$$

Thus, the probability of C not failing in the whole game is at least $1/e(l_p + l_{p'} + l_{ap} + l_r)$. In addition, the probability of C uniformly selecting $\mathcal{U}^*$ from the list L_1 is $1/l_1$. Hence, the probability ε' of C in solving the co-DBDH problem is at least $\varepsilon/l_1 e(l_p + l_{p'} + l_{ap} + l_r)$.

Theorem 2: If a PPT adversary A_2 can break the IND-CCA2-II security of CMGs-CLPSS with an advantage ε by asking l_i queries to the H_i $(i = 0, 1, 2, 3)$ oracle, l_p queries to the private key oracle, and l_{ap} queries to the proxy key oracle, there exists an algorithm C which can solve the co-DBDH problem with an advantage

$$\varepsilon' \geq \frac{\varepsilon}{e l_1\left(l_p + l_{ap}\right)} \tag{32}$$

where e is the base of natural logarithm.

Proof: As a challenger, C takes as input a co-DBDH problem instance $< g, C_1 = g^a, C_2 = g^b, \mathcal{X}, \mathcal{U} >$ and its aim is to determine the value of $\mathcal{U} = e(g, \mathcal{X})^{ab}$. For this purpose, C runs the adversary A_2 as a subroutine in the interactive game.

C maintains six lists $< L_0, L_1, L_2, L_3, L_k, L_{ap} >$ which are empty in the beginning, and these lists are made use of tracing the H_0 oracle, H_1 oracle, H_2 oracle, H_3 oracle, public key oracle and proxy key oracle, respectively. C chooses an integer j from $\{1,2,\ldots, l_0\}$ and considers the identity I_j as challenge identity. Let δ be the probability of $I_i = I_j$, and the value of δ will be determined later.

First of all, C runs **Setup($\mathbf{1^k}$)** to obtain a set of system parameters $\mathcal{L}$ with $y = g^x$ and outputs $< \mathcal{L}, x >$ to A_2.

Phase 1. In an adaptive way, A_2 submits a series of polynomially bounded number of queries to C. Queries and answers to the $H_0 \sim H_3$ oracles are as those in **Phase 1** in **Theorem 1**. Other queries and answers are described as follows.

Request public key: A_2 queries a public key relevant to this identity I_i of its choice. It is necessary for C to consider two cases in response to this query:

Case 1: If it is the jth query, C sets $y_i = C_1$ and outputs the public key y_i to A_2. Then C stores $< I_i, -, y_i, - >$ into the list L_k.

Case 2: If it is not the jth query, C selects a random number x_i from Z_p and calculates $y_i = g^{x_i}$. Then C stores $< I_i, x_i, y_i, - >$ into the list L_k and outputs the public key y_i to A_2.

Private key queries: A_2 submits a private key query for identity I_i of its choice. C fails and terminates the game if this query is the jth query; otherwise, C calculates the partial private key $r_i = g^{\lambda}$. C first calls the public key oracle along with H_0 oracle. Then C outputs $< x_i, r_i >$ to A_2 and updates the list L_k with $< I_i, x_i, y_i, r_i >$. A_2 can verify the validity of the partial private key r_i by the equality (33):

$$e\left(g, r_i\right) = e\left(y, \mathcal{H}_i\right) \tag{33}$$

Proxy key queries: A_2 submits a proxy key query for $< I_a, I_p, m_w >$. If $I_a = I_j$, C fails and terminates this game; otherwise, C calculates $V = \zeta^{x_a}$ by calling the public and private key oracles along with the H_3 oracle. Then C calculates the proxy signcryption key $x_{ap} = V\mathcal{H}_p^{x_p}$ if the equality $e(g, V) = e(y_a, \zeta)$ holds. Finally, C returns x_{ap} to A_2 and stores $< m_w, x_{ap}, V >$ into the list L_{ap}.

Proxy signcryption queries: A_2 issues a query of proxy signcryption for $< I_p, I_b, m_w, m >$. C calls the H_0 and H_3 oracles along with the public key oracle and (private) private key oracles. It is needful for C to consider two cases to deal with this query.

Case 1: If $I_p \neq I_j$, C outputs σ to A_2 by running actual proxy signcryption algorithm.

Case 2: If $I_p = I_j$, C chooses a random number μ from Z_p and calculates $r = g^{\mu}\left(y y_p\right)^{-1}$. C continues to calculate $< \rho, c, \phi, s >$ by the equalities (34)~(37):

$$\rho = e\left(r, r_b\mathcal{H}_b^{x_b}\right) \in G_3 \tag{34}$$

$$c = \upsilon \oplus m \in \{0,1\}^{n_2} \tag{35}$$

$$\phi = \mathcal{X} \in G_2 \tag{36}$$

$$s = \zeta^{x_a}\mathcal{H}_p\phi^{\mu} \tag{37}$$

Finally, C stores $< m_w, m, r, y_a, y_p, \rho, \phi >$ into the list L_2 and outputs $\sigma =< r, c, s >$ into the list L_2. A_2 can easily verify the validity of σ by the equality (38).

$$\begin{aligned} e\left(y_a, H_3\left(m_w, y_a\right)\right) e\left(yy_p, \mathcal{H}_p\right) e\left(r, \phi\right) &= e\left(y_a, \zeta\right) e\left(yy_p, \mathcal{H}_p\right) e\left(r, \phi\right) \\ &= e\left(y_a, \zeta\right) e\left(g, \mathcal{H}_p \phi^{\mu}\right) \\ &= e\left(g, \zeta^{x_a} \mathcal{H}_p \phi^{\mu}\right) \\ &= e\left(g, s\right) \end{aligned} \tag{38}$$

Unsigncryption queries: A_2 issues an unsigncryption query for $< I_p, I_b, m_w, \sigma >$. C calls various oracles to deal with this unsigncryption query as follows.

Case 1: If $I_b \neq I_j$, C outputs a result to A_2 by running actual unsigncryption algorithm.

Case 2: If $I_b = I_j$, C checks the list L_1 to look through a tuple $< m_w, r, \rho, \upsilon >$ such that $\mathcal{O}_{\text{co-D}BDH}$ returns 1 when A_2 queried for $< y_b, r, \mathcal{X}, \mathcal{U} >$. If there is this case, C calculates $< \rho, m >$ by the equalities (39) and (40):

$$\rho = e\left(r, \mathcal{X}^{x_b}\right) \cdot \mathcal{U} \tag{39}$$

$$m = \upsilon \oplus c \in \{0,1\}^{n_2} \tag{40}$$

Then C calculates $\phi = H_2\left(m_w, m, r, y_a, y_p, \rho\right)$ and checks whether $e\left(g, s\right) = e\left(y_a, \zeta\right) e\left(yy_p, \mathcal{H}_p\right) e\left(r, \phi\right)$ holds. C outputs m to A_2 if the verification equality holds and $\perp$ otherwise.

Challenge. At the end of the first phase, A_2 outputs $< m_0, m_1 > \in \{0, 1\}^{n_2}$ along with $< I_b^*, I_p^*, m_w^* >$ to C. A_2 cannot request the private key for identity I_b in **Phase 1**. Assume A_2 has queried the public key oracle and (partial) private key oracles along with the H_0 and H_3 oracles before challenge query. It is needful for C to consider two cases to handle this query.

Case 1: If $I_b^* \neq I_j$, C fails and terminates this game.

Case 2: If $I_b^* = I_j$, C randomly chooses $\mathcal{U}^* \in G_3$ and $t \in \{0, 1\}$. C sets $r^* = C_2$ and continues to calculate $< \rho^*, \upsilon^*, c^* >$ by the equalities (41)~(43):

$$\rho^* = e\left(r^*, \mathcal{X}^x\right) \cdot \mathcal{U} \tag{41}$$

$$\upsilon^* = H_1\left(m_w^*, r^*, \rho^*\right) \tag{42}$$

$$c^* = \upsilon^* \oplus m_t \in \{0,1\}^{n_2} \tag{43}$$

C stores $< m_w^*, r^*, \rho^*, \upsilon^* >$ into the list L_1 and then calculates $< \phi^*, s^* >$ by the equalities (44) and (45):

$$\phi^* = H_2\left(m_w^*, m_t, r^*, y_a^*, y_p^*, \rho^*\right) \tag{44}$$

$$s^* = r_a^*\left(r^* y_a^*\right)^{\lambda} \tag{45}$$

Finally, C stores $< m_w^*, m_t, r^*, y_a^*, y_p^*, \rho^*, \phi^* >$ into the list L_3 and outputs $\sigma^* =< r^*, c^*, s^* >$ as a challenge ciphertext.

Phase 2. A_2 continues to submit a sequence of queries to C as those in **Phase 1**. A_2 cannot request the private key query for identity I_b^* in **Phase 2**. A_2 cannot make an unsigncryption query for $\sigma^* =< r^*, c^*, s^* >$ after challenge phase.

At the end of the game, A_1 outputs a guess t^* of t. If $t^* = t$, C outputs the solution of co-DBDH problem instance by the equality (46):

$$\mathcal{U}^* = e\left(C_2, \mathcal{X}^*\right)^a = e\left(y_b^*, \mathcal{X}^*\right)^b = e\left(g, \mathcal{X}^*\right)^{ab} \tag{46}$$

Probability analysis. As described above, l_p is the times of querying the private key oracle and l_{ap} is the times of querying the proxy key oracle. As shown in [8], the probability of C not failing in the first or second phase is $\delta^{l_p+l_{ap}}$, and the probability of C not failing in challenge phase is $1 - \delta$. Thus, the probability of C not failing in the whole game is $\delta^{l_p+l_{ap}}(1 - \delta)$, this value is maximized at (please see the equality (31))

$$\delta = 1 - \frac{1}{1 + l_p + l_{ap}} \tag{47}$$

Then the probability of C not failing in the whole game is at least $1/e(l_p+l_{ap})$. Since the probability of C uniformly selecting $\mathcal{U}^*$ from the list L_1 is $1/l_1$. Therefore, the probability ε' of C in solving the co-DBDH problem is at least $/l_1 e(l_p + l_{ap})$.

Theorem 3: If a UF-CMGs-CLPSS-CMA-I adversary A_1 can break the UF-CMA-I security of CMGs-CLPSS with an advantage ε by issuing l_i queries to the H_i $(i = 0, 1, 2, 3)$ oracle, l_p queries to the private key oracle, $l_{p'}$ queries to the partial private key oracle, l_{ap} queries to the proxy key oracle, and l_r queries to the public key replacement oracle, there exists an algorithm C which can solve the co-CDH problem with an advantage ε', where

$$\varepsilon' \geq \frac{\varepsilon}{e\left(l_p + l_{p'} + l_{ap} + l_r\right)} \tag{48}$$

where e is the base of natural logarithm.

Proof: C takes a random instance $< g, g^a, \mathcal{X} >$ of co-CDH problem as input and the aim of C is to attempt to determine the value of $\mathcal{X}^a \in G_2$. For this aim, C acts as the role of A_1's challenger of and runs A_1 as a subroutine in the interactive game.

First of all, C calls **Setup(1^k)** to obtain a set of the system parameters $\mathcal{L}$ with $y = g^a$ and outputs $\mathcal{L}$ to A_1.

Queries. In an adaptive way, A_1 submits a sequence of queries including various hash queries, public key queries, partial private key queries, private key queries, public key replacement queries, proxy key queries, proxy signcryption queries, and unsigncryption queries. Moreover, C answers these queries in the same method as those in **Phase 1** in **Theorem 1**.

Forgery. At the end of the queries, A_1 outputs a forgery $< I_b^*, I_p^*, m_w^*, \sigma^* >$ to C. In queries, A_1 cannot extract the private key and partial private key of identity I_p^* and the public key of this identity should not be replaced. In addition, A_1 cannot request the private key for identity I_a^* and $< I_b^*, I_p^*, m_w^*, \sigma^* >$ should not be returned by the signcryption oracle.

C fails and stops the game if this query is not the jth query; otherwise, C calls the H_0 and H_3 oracles along with public key oracle. Then C obtains the solution of co-CDH problem instance:

$$\mathcal{X}^a = \frac{s^*}{\zeta^{x_a^*}\mathcal{X}^{x_p^*}\phi^{\mu}} \tag{49}$$

If C succeeds in the interactive game, this shows that the equality (50) holds:

$$\begin{aligned} e\left(g, s^*\right) &= e\left(y_a^*, H_3\left(m_w, y_a\right)\right) e\left(yy_p^*, \mathcal{H}_p\right) e\left(r^*, \phi\right) \\ &= e\left(y_a^*, \zeta\right) e\left(g, \mathcal{H}_p^{a+x_p*}\right) e\left(g, \phi^{\mu}\right) \\ &= e\left(g, \zeta^{x_a*}\mathcal{H}_p^{a+x_p*}\phi^{\mu}\right) \end{aligned} \tag{50}$$

Probability analysis. Refer to the analysis of probability in **Theorem 1**, the probability of C not aborting the simulation is at least $1/e(l_p + l_{p'} + +l_{ap} + l_r)$. Therefore, the probability ε' of C in solving the co-CDH problem is at least $\varepsilon/e(l_p + l_{p'} + l_{ap} + l_r)$.

Theorem 4: If a UF-CMGs-CLPSS-CMA-II adversary A_2 can break the UF-CMA-II security of CMGs-CLPSS with an advantage $\mathcal{E}$ by requesting l_i queries to the H_i $(i = 0, 1, 2, 3)$ oracle, l_p queries to the private key oracle and l_{ap} queries to the proxy key oracle, there is an algorithm C that can utilize A_2 to solve the co-CDH problem with an advantage ε', where

$$\varepsilon' \geq \frac{\varepsilon}{e\left(l_p + l_{ap}\right)} \tag{51}$$

where e is the base of natural logarithm.

Proof: C receives a random instance $< g, g^a, \mathcal{X} >$ of co-CDH problem and the purpose of C is to determine the value of $\mathcal{X}^a \in G_2$. For this purpose, C makes use of A_2 as a subroutine and acts as the role of its challenger in the whole game.

First of all, C runs **Setup($1^{\mathbf{k}}$)** to obtain a set of system parameters $\mathcal{L}$ with $y = g^x$ and delivers $< \mathcal{L}, x >$ to A_2.

Queries. In an adaptive way, A_2 issues a series of queries to C. Queries and answers are identical to those in **Phase 1** in **Theorem 2**.

Forge. At the end of the queries, A_2 outputs a forgery $< I_b^*, I_p^*, m_w^*, \sigma^* >$. In queries, A_2 cannot query the private key for identity I_a^* and $< I_b^*, I_p^*, m_w^*, \sigma^* >$ should not be returned by the signcryption oracle.

C fails and stops the game if it is not the jth query; otherwise, C calls various hash oracles along with public key oracle. Then C outputs the solution of co-CDH problem instance:

$$\mathcal{X}^a = \frac{s^*}{\zeta^{x_a^*}\mathcal{X}^{x}\phi^{\mu}} \tag{52}$$

If C succeeds in the interactive game, this implies that the equality (53) must hold.

$$\begin{aligned} e\left(g, s^*\right) &= e\left(y_a^*, H_3\left(m_w, y_a\right)\right) e\left(yy_p^*, \mathcal{H}_p\right) e\left(r^*, \phi\right) \\ &= e\left(y_a^*, \zeta\right) e\left(g, \mathcal{H}_p^{x+a}\right) e\left(g, \phi^{\mu}\right) \\ &= e\left(g, \zeta^{x_a*}\mathcal{H}_p^{x+a}\phi^{\mu}\right) \end{aligned} \tag{53}$$

TABLE 3. Comparison of time complexity.

Schemes	Time complexity	Security	
		Con	Unf
Literature[13]	$7t_E$+$9t_P$+$4t_H$	Yes	Yes
Literature[22]	$1t_E$+$7t_P$+$8t_H$	Yes	Yes
Literature[23]	$5t_E$+$7t_P$+$4t_H$	Yes	Yes
CMGs-CLPSS	$4t_E$+$6t_P$+$4t_H$	Yes	Yes

Probability analysis. Refer to the probability analysis in **Theorem 2**, the probability of C not terminating the simulation is at least $1/e(l_p + l_{ap})$. Hence, the probability ε' of C in solving the co-CDH problem is at least $\varepsilon/e(l_p + l_{ap})$.

VII. PERFORMANCE ANALYSIS

In this section, we compare CMGs-CLPSS with similar schemes [13], [22], [23] from pairings in terms of computation complexity and security. Here we consider computational complexity in the proxy signcryption and unsigncryption phases.

In Table 3, t_H is the time complexity of running a one-way hash function,t_P is the time complexity of running a bilinear pairing operation in multiplicative group, and t_E is the time complexity of running an exponent operation in multiplicative group. Moreover, **Unf** denotes unforgeability and **Con** denotes confidentiality.

As shown in [24], $t_P \approx 1440 t_H$ and $t_E \approx 21 t_H$. From Table 3, we can readily know the time cost of CMGs-CLPSS is lower than the schemes in [13], [22], [23]. Seen from Table 3, CMGs-CLPSS is an efficient and secure certificateless proxy signcryption scheme.

VIII. SUMMARY

For more complicated business flow processes, secure privilege delegation mechanism has become a necessary function for enterprises, organizations and even every modern citizen. For this reason, we construct a new CMGs-CLPSS, in which the entrustment of signing rights to a proxy signcrypter at the behest of an original signcrypter imparts its utility in various fields such as online proxy auction, mobile agents, cloud computing and ubiquitous computing. CMGs-CLPSS needs no secure channel and is IND-CCA2 and UF-CMA secure in the random oracle model. CMGs-CLPSS can realize secure data transmission and will receive significant attention because it simplifies the traditional PKC and solves the key escrow problem suffered by IB-PKC. In the future work, we are going to construct secure and efficient cryptographic algorithms from CL-PKC using the techniques in [25]–[27].

REFERENCES

[1] M. Chen and F. Wang, "Resistance to misuse ciphertext of signcryption scheme," *J. Electron. Inf. Technol.*, vol. 41, no. 4, pp. 1010–1016, 2019.

[2] C.-F. Wang, C. Liu, Y.-H. Li, S.-F. Niu, and Y.-L. Zhang, "Two-way and anonymous heterogeneous signcryption scheme between PKI and IBC," *J. Commun.*, vol. 38, no. 10, pp. 10–17, 2017.

[3] J. M. Li, H. F. Yu, and Y. Xie, "ElGamal broadcasting multi-signcryption protocol with UC security," *J. Comput. Res. Develop.*, vol. 56, no. 5, pp. 1101–1111, 2019.

[4] L. H. Zhang and R. Sun, "Mutual signcryption schemes under heterogeneous systems," *J. Electron. Inf. Technol.*, vol. 38, no. 11, pp. 2948–2953, 2016.

[5] X.-H. Lu, Q.-Y. Wen, and L.-C. Wang, "A lattice-based heterogeneous signcryption," *J. Univ. Electron. Sci. Technol. China*, vol. 45, no. 3, pp. 458–462, 2016.

[6] C. Wang, Y. Li, S. Niu, and Y. Zhang, "Efficient heterogeneous signcryption scheme in the standard model," *J. Electron. Inf. Technol.*, vol. 39, no. 4, pp. 881–886, 2017.

[7] F. Li, B. Liu, and J. Hong, "An efficient signcryption for data access control in cloud computing," *Computing*, vol. 99, no. 5, pp. 465–479, 2017.

[8] X. H. Lu, Q. Y. Wen, and L. C. Wang, "Non-trapped gate-based signcryption scheme," *J. Electron. Inf. Technol.*, vol. 38, no. 9, pp. 2287–2293, 2016.

[9] H.-F. Yu and B. Yang, "Identity-based hybrid signcryption scheme using ECC," *J. Softw.*, vol. 26, no. 12, pp. 3174–3182, 2015.

[10] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures for delegating signing operation," in *Proc. 3rd ACM Conf. Comput. Commun. Secur.*, New York, NY, USA: ACM Press, 1996, pp. 48–57.

[11] C. Gamage, J. Leiwo, and Y. Zheng, "An efficient scheme for secure message transmission using proxy-signcryption," in *Proc. 22nd Australasim Comput. Sci. Conf.*, Auckland, New Zealand: Springer-Verlag, 1999, pp. 420–431.

[12] Y. Ming, J. Feng, and Q. J. Hu, "Secure identity-based proxy signcryption scheme in standard model," *J. Comput. Appl.*, vol. 34, no. 10, pp. 2834–2839, 2014.

[13] C. X. Zhou, "Identity-based generalized proxy signcryption in the standard model," *J. Cryptologic Res.*, vol. 3, no. 3, pp. 307–320, 2016.

[14] H. Yu, Z. Wang, J. Li, and X. Gao, "Identity-based proxy signcryption protocol with universal composability," *Secur. Commun. Netw.*, vol. 2018, pp. 1–11, Dec. 2018.

[15] M. Barbosa and P. Farshim, "Certificateless signcryption," in *Proc. ACM Symp. Inf., Comput. Commun.*, New York, NY, USA, 2008, pp. 369–372.

[16] Z. Liu, Y. Hu, X. Zhang, and H. Ma, "Certificateless signcryption scheme in the standard model," *Inf. Sci.*, vol. 180, no. 3, pp. 452–464, 2010.

[17] J. Weng, G. Yao, R. H. Deng, M.-R. Chen, and X. Li, "Cryptanalysis of a certificateless signcryption scheme in the standard model," *Inf. Sci.*, vol. 181, no. 3, pp. 661–667, 2001.

[18] S. Miao, F. Zhang, S. Li, and Y. Mu, "On security of a certificateless signcryption scheme," *Inf. Sci.*, vol. 232, pp. 475–481, May 2013.

[19] H. F. Yu and B. Yang, "Provably secure certificateless hybrid signcryption," *Chin. J. Comput.*, vol. 38, no. 4, pp. 804–813, 2015.

[20] S. K. H. Islam and F. Li, "Leakage-free and provably secure certificateless signcryption scheme using bilinear pairings," *Comput. J.*, vol. 58, no. 10, pp. 2636–2648, 2015.

[21] H.-F. Yu and B. Yang, "Low-computation certificateless hybrid signcryption scheme," *Frontier Inf. Technol. Electron. Eng.*, vol. 18, no. 7, pp. 928–940, Jul. 2017.

[22] G. Yu and W. B. Han, "Certificateless signcryption scheme with proxy unsigncryption," *Chin. J. Comput.*, vol. 34, no. 7, pp. 1291–1299, 2011.

[23] Y. Ming and Y. Wang, "Proxy signcryption scheme in the standard model," *Secur. Commun. Netw.*, vol. 8, no. 8, pp. 1431–1446, 2015.

[24] C.-I. Fan, W.-Z. Sun, and V. S.-M. Huang, "Provably secure randomized blind signature scheme based on bilinear pairing," *Comput. Math. Appl.*, vol. 60, no. 2, pp. 285–293, 2010.

[25] Y.-L. Liu, Y.-P. Wang, X.-F. Wang, Z. Xia, and J.-F. Xu, "Privacy-preserving raw data collection without a trusted authority for IoT," *Comput. Netw.*, vol. 148, pp. 340–348, Jan. 2019.

[26] Y. Liu and Q. Zhao, "E-voting scheme using secret sharing and K-anonymity," *World Wide Web*, vol. 22, no. 4, pp. 1657–1667, 2019.

[27] J. Song, Y. Liu, J. Shao, and C. Tang, "A dynamic membership data aggregation (DMDA) protocol for smart grid," *IEEE Syst. J.*, to be published. doi: 10.1109/JSYST.2019.2912415.

HUIFANG YU was born in Qinghai. She received the M.S. degree in information security from Northwest Normal University and the Ph.D. degree in cryptography from Shaanxi Normal University. She is currently a Professor and a Master Supervisor with the Xi'an University of Posts and Telecommunications. She is also a Senior Member of CACR. She has completed more than ten research projects, including a 973 basic research projects. She is the PI of several research projects, including the National Natural Science Foundation of China. She has published two books and more than 50 articles. She has ten national invention patents. Her main research interests include cryptography and information security.

ZHICANG WANG was born in Qinghai. He is currently pursuing the Ph.D. degree with Qinghai Normal University. He is a Professor with the Xi'an University of Posts and Telecommunications. He has finished more than ten research projects, including the National Natural Science Foundation of China. He has published two books and more than 20 articles. He has three national invention patents. His main research interests include information security, neural networks, and computational intelligence. He is a member of CCF.