Certificateless Public Key Encryption Secure against Malicious KGC Attacks in the Standard Model

Yong Ho Hwang[*] Joseph K. Liu[†]

Abstract

We introduce the first secure Certificateless Public Key Encryption (CL-PKE) scheme against a malicious Key Generation Center (KGC) in the standard model. Recently, Au *et al.* [3] pointed out that the previous security models for CL-PKE schemes cannot guarantee the security against a malicious KGC. They also showed that although some schemes are secure against malicious KGC, they require the random oracle model to prove the security. In this paper, we first show that previous CL-PKE schemes in the standard model are not secure against malicious KGC. And then, we construct a new CL-PKE scheme with rigorous security proof against the attacks of a malicious KGC in the standard model, which is the first in the literature.

1 Introduction

In traditional Public Key Cryptosystem (PKC), to gurantee the authenticity of a public key, a trusted party called the Certification Authority (CA) issues a digitially signed certificate binding a user and his public key. However, the need for public key infrasructure (PKI) supporting certificates is considered the main difficulty on the deployment and management of a traditional PKC. Identity Based Cryptography (IBC), invented by Shamir [20], solved this problem by using any string (such as email address, user name or phone number) as public keys, where a trusted third party called the Private Key Generator (PKG) manages the generation and distribution of all system parameters including the user's private key. While it is no longer necessary to explicitly authenticate public keys, it has the key escrow problem since the PKG generates the private key of every user.

Certificateless Public Key Cryptography (CL-PKC), which is intermediate between IBC and PKC, was first introduced by Al-Riyami and Paterson in 2003 [1]. The main purpose of the CL-PKC is to solve the key escrow problem inherited from IBC [20, 7] without the use of certificates as in the traditional PKC, which is generally considered to be costly to use and manage. In a CL-PKC, a Key Generation Center (KGC) is involved in issuing *user partial private key* computed from the user's identity which is assumed to be unique in the system. The user also *independently* generates an additional user public/private key pair. Then the user combines the partial private key with his private key to generate the decryption key or signing key. Cryptographic operations such as decryption and signature generation can then be performed successfully only when both the user partial private key and the user private key are known. Knowing only one of them should not be able to impersonate the user, that is, carrying out any cryptographic operations as the user. In this way, even if the KGC knows the user partial private key, it is not able to impersonate the user, since it does not know the corresponding user private key.

There are two different types of adversaries that are generally considered in CL-PKC:

[*]Department of Computer Science; The Johns Hopkins University; 3400 N. Charles Street; Baltimore, MD 21218, USA. Email: yhhwang@cs.jhu.edu.

[†]Department of Computer Science; University of Bristol, UK. Email: liu@cs.bris.ac.uk.

Type I The adversary acts as a third party who tries to impersonate a user after compromising the user private key and/or *replacing* the user public key with some value chosen by the adversary. However, it does not know the user partial private key.

Type II The adversary acts as the KGC who knows the partial private key of a user. That is, the KGC is not considered as honest and thus tries to impersonate the user. However, the KGC does not know the user private key or being able to replace the user public key.

Type II security is for solving the key escrow problem inherited in IBC. That is, even if the KGC is malicious, the KGC should not be able to perform any cryptographic operation as the user, provided that the KGC cannot replace the user public key or find out the user private key, but the KGC knows the user partial private key. Nevertheless, when we take a look at most of the previously certificateless encryption/signature schemes and adversarial models [1, 22, 23, 24, 2, 6, 4, 9, 17, 25, 14, 18, 12], we can notice that all of them have an implicit assumption that the KGC always generates its master public/secret key pair *honestly* according to scheme specification. The model only allows the Type II adversary to know the master secret key of the KGC. This is definitely *not* strong enough and cannot practically reflect the situation in the real world. In the reality, the malicious KGC will try every effort to break the system. Obviously it is not reasonable to assume that it will generate the master public/secret key pair honestly according to the scheme specification. It may generate the key pair in any way it favours. We should not make any assumption on the key generation process. Otherwise it will be against the original spirit of the CL-PKC.

Unfortunately, most of the schemes in the literature only focus on the discussion or improvement on Type I security, while omit the importance of the malicious KGC matter (Type II security). Until recently, Au *et al.* [3] pointed out the weakness of previous security model and analyzed some previous schemes under the enhanced malicious KGC model. They re-proved the encryption scheme by Libert and Quisquater [18] and showed that they are secure against malicious KGC attacks. However, it requires the random oracle in the security proof.

Independently of [1], Gentry [15] introduced a different but related concept, called Certificate Based Encryption (CBE). This approach is closer to the context of a traditional PKC model as it also involves a CA providing an efficient implicit certification service for users' public keys. However, in this model certificate is a part of secret key, so that certification is implicit. Furthermore, there is no key escrow since certificate itself is only a part of the secret key while the other part is only known to the user himself. Many subsequent works have investigated the relations between IBC, CL-PKC, and CBE schemes and established the results of essential equivalences among them [22, 23, 24, 2, 18]. In addition, Dodis and Katz described generic methods to construct CCA-secure multiple-encryption schemes from CCA-secure public key encryption schemes [13]. They showed that their methods apply to the design of the CBE scheme and yield CBE schemes without random oracle. However, the previous techniques cannot be directly applied to construct a CL-PKE scheme secure against malicious KGC attacks, since their security models did not consider the malicious environments by the KGC at all. In [3], Au *et al.* showed that only one generic construction is secure against malicious KGC attackes in the random oracle model. However, constructing CL-PKE secure against malicious KGC attacks in the standard model remains as open problem.

Contribution. In this paper, we emphasis our focus on Type II security. First, we refine the adversarial model for a Type II adversary (malicious KGC) of CL-PKE. Then, we analyze some CL-PKE schemes whose security is proven in the standard model. We show that they are not secure under the malicious KGC attacks. We further propose a new CL-PKE scheme which is secure against malicious KGC attacks. Its security is also proven in the standard model. It is the first in the literature to achieve the strongest Type II security level without random oracles.

Organization. In the rest of the paper, it is organized as follow. In Section 2 the security model and some preliminaries are reviewed. In Section 3 we analyze the security of some previous CL-PKE schemes. Our new scheme is given in Section 4 and the paper is concluded in Section 5.

2 Preliminaries

2.1 Notation

We use the standard notation to describe probabilistic algorithms and experiments as follow. If $k \in \mathbb{N}$, then 1^k denotes the string of k ones. Let an algorithm $\mathcal{A}$ be a probabilistic polynomial-time (PPT) Turing machine. If $\mathcal{A}$ is an efficient algorithm and $x, y, \ldots$ are inputs for $\mathcal{A}$, then $s \leftarrow \mathcal{A}(x, y, \ldots)$ denotes the probability space that assigns to a string s the probability that $\mathcal{A}$, on input $x, y, \ldots$, outputs s. We write $\mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\cdots}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$ and denote by $z \leftarrow \mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\cdots}(x, y, \ldots)$ the operation of running $\mathcal{A}$ with inputs $x, y, \ldots$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$ and letting z be the output.

For a finite set X, $x \leftarrow X$ denotes the algorithm that samples an element uniformly at random from X. If w is neither an algorithm nor a set then $x \leftarrow w$ is a simple assignment statement. A function $\epsilon : N \rightarrow R$ is negligible if for every constant $c > 0$ there exists an integer n_c such that $\epsilon(n) < n^{-c}$ for all $n \geq n_c$. For a probability space P, $x \leftarrow P$ denotes the algorithm that samples a random element according to P. If $p(\cdot, \cdot, \cdots)$ is a boolean function, then $\Pr[p(x_1, x_2, \ldots)|x_1 \leftarrow P_1, x_2 \leftarrow P_2, \ldots]$ denotes the probability that $p(x_1, x_2, \ldots)$ is true after executing the algorithms $x_1 \leftarrow P_1, x_2 \leftarrow P_2, \ldots$.

2.2 Certificateless Public Key Encryption

We review the definition and security notions of CL-PKE. We define the simplified model as used in [16, 19, 3] which is slightly different from the original one by Al-Riyami and Paterson [1]. The discussion of the main difference can be referred to [16].

Definition 1 *A* certificateless public key encryption *is specified by 5 algorithms as follows:*

- Setup is a *setup algorithm* run by a key generation center (KGC). It takes a security parameter 1^k as input and returns the system parameters mpk and the master key msk. Intuitively, the system parameters will be publicly known, while the master secret key will be known only to the KGC.
- PSK is a *partial private key extraction algorithm* run by the KGC. It takes as inputs mpk, msk, and an identity ID of a user. It returns a partial private key psk_{ID} to a user with an identity ID.
- $\mathsf{UKeyGen}$ is a *user key generation algorithm* that takes as inputs mpk and psk_{ID}, and outputs the public/private key pair $(pk_{\mathsf{ID}}, sk_{\mathsf{ID}})$.
- Enc is an *encryption algorithm* that takes as inputs mpk, ID, pk_{ID}, and a message $m \in \mathcal{M}$ where $\mathcal{M}$ is a message space. It returns a ciphertext C.
- Dec is a *decryption algorithm* that takes as inputs mpk, sk_{ID}, and a ciphertext C. It returns a message $m \in \mathcal{M}$ if the ciphertext is valid and $\perp$ otherwise.

Correctness property requires that $\mathsf{Dec}(mpk, sk_{\mathsf{ID}}, \mathsf{Enc}(mpk, \mathsf{ID}, pk_{\mathsf{ID}}, m)) = m$.

Security Model. The IND-CCA security of CL-PKE distinguishes two types of adversaries. A Type I adversary $\mathcal{A}_I$ does not have the access to the master key of the KGC, but it can replace public keys of arbitrary users. In addition, it can obtain partial private key and (full) private key of arbitrary identities.

On the other hand, a Type II adversary $\mathcal{A}_{II}$ can access to the master key of the KGC, but is not allowed to replace public keys. This adversary is to model the security against an *eavesdropping* KGC. In this paper, we re-define the security model for the Type II adversary to prevent a *malicious* KGC from obtaining the information from the ciphertext in a real environment. In this model, the KGC is not just an eavesdropper, but an *active* adversary. To simulate two types of adversaries, $\mathcal{A}$ is allowed to access to the following oracles.

- $\mathsf{PSK\text{-}Oracle}$. A *partial private key extraction oracle* takes as input an identity ID and returns $\mathsf{psk}_{\mathsf{ID}}$.
- $\mathsf{PK\text{-}Oracle}$. A *public key broadcast oracle* takes as input an identity ID and returns $\mathsf{pk}_{\mathsf{ID}}$.
- $\mathsf{RPK\text{-}Oracle}$. A *public key replace oracle* takes as input an identity ID and a valid public key $\mathsf{pk}_{\mathsf{ID}}$. It replaces the associated user's public key with the new one $\mathsf{pk}'_{\mathsf{ID}}$.
- $\mathsf{SK\text{-}Oracle}$. A *private key extraction oracle* takes as input an identity ID and returns $\mathsf{sk}_{\mathsf{ID}}$ if the associated public key with ID was not replaced.
- $\mathsf{Dec\text{-}Oracle}$. A *decryption oracle* takes as inputs an identity ID and a ciphertext C, and returns the decrypted ciphertext with $\mathsf{sk}_{\mathsf{ID}}$. If the user's public key has been replaced, it requires an additional input of the corresponding private key for the decryption. If it is not given this private key, it outputs $\perp$.

First we consider the most common security goal and attack model: *Indistinguishability against adaptive chosen ciphertext attacks*, denoted by IND-CCA. For an efficient algorithm $\mathcal{A}$, which runs in two stages $(\mathcal{A}_1, \mathcal{A}_2)$, we define the adversary's advantage as

$$\mathsf{Adv}^{\texttt{IND-CL-CCA}}_{\mathcal{A}_{I(\text{ or}II)}}(k) = \left|\Pr\left[\mathfrak{b} = \mathfrak{b}' \middle| \begin{array}{l} (mpk, msk) \leftarrow \mathsf{Setup}(1^k), (m_0, m_1, \mathsf{ID}^*, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{Type I (or II)}}}(mpk, a), \\ \mathfrak{b} \leftarrow \{0,1\}, C^* \leftarrow \mathsf{Enc}(m_{\mathfrak{b}}, pk_{\mathsf{ID}^*}, \mathsf{ID}^*, mpk), \mathfrak{b}' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{Type I (or II)}}}(C^*, s) \end{array}\right] - \frac{1}{2}\right|$$

where $\mathcal{O}_{\text{Type I}} = \{\mathsf{PSK\text{-}Oracle}, \mathsf{PK\text{-}Oracle}, \mathsf{RPK\text{-}Oracle}, \mathsf{SK\text{-}Oracle}, \mathsf{Dec\text{-}Oracle}\}$, $\mathcal{O}_{\text{Type II}} = \{\mathsf{PK\text{-}Oracle}, \mathsf{SK\text{-}Oracle}, \mathsf{Dec\text{-}Oracle}\}$, and s is some internal state information. If $\mathcal{A}$ is a Type I adversary, $a = \emptyset$. Otherwise, $a = msk$. Actually, $\mathcal{A}_{II}$ does not need to issue partial private key queries, since it can compute them from the master key by itself. There are some restrictions as follows.

- $\mathcal{A}_I, \mathcal{A}_{II}$ cannot extract the private key for ID^*.
- $\mathcal{A}_I, \mathcal{A}_{II}$ cannot make a decryption query on C^*.
- $\mathcal{A}_I$ cannot request the private key for any identity whose public key has already been replaced.
- $\mathcal{A}_I$ cannot extract the partial private key for ID^* if it has replaced the public key for ID^* before the challenge phase.

In the Type I security model, we do not allow the Type I adversary to issue decryption queries made on identities for which it has replaced the public keys. However, the strongest security model of [1] does expect that the decryption oracle should be able to output consistent answers even for identities whose public keys have been replaced and for which they do not know the corresponding private keys. This is a very strong notion of security. Several schemes [6, 9, 10] have weaken this definition that the adversary does not make the decryption queries for which the public key has been replaced. We adopt this model for the Type I security. [1]

[1]It is also referred as Type I$^-$ security in [6]. In the rest of the paper, we omit the $^-$ sign for this security model.

In previous works, to simulate the actions of an honest-but-curious KGC, $\mathcal{A}_{II}$ is given msk. In the above model, we assume that the KGC honestly generates all the public parameters. However, we need a stronger security model to capture the actions of a malicious KGC, since it can maliciously compute the public parameters. In a real environment, the KGC generates the public parameters and the master secret key by itself, while in a security game the simulation algorithm $\mathcal{B}$ generates the public parameters and the master secret key which are then given to an adversary. From it, there is the gap between the real environment and the simulation environment. Therefore, to simulate a malicious KGC, we adopt the modification from [3] to allow $\mathcal{A}_{II}$ to generate all the public parameters and the master secret key. In our model, an adversarial algorithm $\mathcal{A}_{II}$ runs in three stages $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$. We define the Type II adversary's advantage as

$$\mathsf{Adv}^{\texttt{IND-CL-CCA}}_{\mathcal{A}_{II}}(k) = \left|\Pr\left[\mathfrak{b} = \mathfrak{b}' \;\middle|\; \begin{array}{l} (mpk, msk) \leftarrow \mathcal{A}_0(1^k), (m_0, m_1, \mathsf{ID}^*, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{Type II}}}(mpk, msk), \\ \mathfrak{b} \leftarrow \{0,1\}, C^* \leftarrow \mathsf{Enc}(m_{\mathfrak{b}}, pk_{\mathsf{ID}^*}, \mathsf{ID}^*, mpk), \mathfrak{b}' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{Type II}}}(C^*, s) \end{array}\right] - \frac{1}{2}\right|$$

The original model believes that the KGC honestly generates the public parameters. However, the KGC can maliciously generate them. For example, we assume that the KGC should pick three random elements (g, g_1, g_2) in a multiplicative group $\mathbb{G}$ of order p as the public parameters. The KGC maliciously generates random elements such as $g_1 = g^\alpha, g_2 = g^\beta$ from elements α, β of $\mathbb{Z}_p^*$ and it can try to break the system from α, β. Actually, we can show that there exists a CCA-secure CL-PKE scheme against Type II adversaries in previous model, but not secure against the modified Type II adversarial model [11, 19]. Some previous works are broken by this attack. Therefore, we allow the type II adversary to generate all public parameters.

2.3 Complexity Assumptions

We briefly review bilinear maps and describe complexity assumptions related to our construction. Let $\mathbb{G}$ and $\mathbb{G}_T$ be the two multiplicative cyclic groups of order p for some large prime p. The *bilinear* map should be satisfied the following properties:

1. Bilinear: We say that a map e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is bilinear if $e(g^a, h^b) = e(g,h)^{ab}$ for all $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$.
2. Non-degenerate: The map does not send all pairs in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_T$. Observe that since $\mathbb{G}$, $\mathbb{G}$ are groups of prime order this implies that if g is a generator of $\mathbb{G}$ then $e(g, g)$ is a generator of $\mathbb{G}_T$.
3. Computable: There is an efficient algorithm to compute $e(g, h)$ for any $g, h \in \mathbb{G}$.

A bilinear map satisfying the three properties above is said to be an *admissible* bilinear map.

The security of our construction is based on a complexity assumption called the Decisional Bilinear Diffie-Hellman (DBDH) assumption. In addition, our construction uses a collision resistant hash function to check the validity of a ciphertext.

Definition 2 (Decisional Bilinear DH (DBDH) Assumption) *Given a group $\mathbb{G}$ of prime order p with generator g, a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ and elements $g^a, g^b, g^c \in \mathbb{G}$, $e(g,g)^z \in \mathbb{G}_T$ where a, b, c, z are selected uniformly at random from $\mathbb{Z}_p^*$. A fair binary coin $\mathfrak{b} \in \{0,1\}$ is flipped. If $\mathfrak{b} = 1$, it outputs the tuple $(g, A = g^a, B = g^b, C = g^c, Z = e(g,g)^{abc})$. If $\mathfrak{b} = 0$, it outputs the tuple $(g, A = g^a, B = g^b, C = g^c, Z = e(g,g)^z)$. The problem is to guess the value of $\mathfrak{b}$.*

We define the advantage of a probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ as

$$Adv_{\mathcal{A}}^{\text{DBDH}}(k) = \left| \ \Pr[1 \leftarrow \mathcal{A}(g, g^a, g^b, g^c, e(g,g)^{abc})] - \Pr[1 \leftarrow \mathcal{A}(g, g^a, g^b, g^c, e(g,g)^z)] \ \right|$$

where the probability is oven the randomly chosen a, b, c, z and the random bits consumed by $\mathcal{A}$. We assume that $Adv_{\mathcal{A}}^{DBDH}(k)$ is a negligible function for all PPT algorithms $\mathcal{A}$.

Definition 3 (Collision Resistant (CR) Assumption) *A hash function $H \leftarrow \mathcal{H}(k)$ is collision resistant if for all PPT algorithms $\mathcal{A}$ the advantage*

$$Adv_{\mathcal{A}}^{\text{CR}}(k) = \Pr[H(x) = H(y) \wedge x \neq y \mid (x,y) \leftarrow \mathcal{A}(1^k, H) \wedge H \leftarrow \mathcal{H}(k)]$$

is negligible as a function of the security parameter.

3 Analysis of Liu-Au-Susilo CL-PKE scheme

Recently Liu *et al* [19] proposed a secure CL-PKE scheme (called LAS scheme) in the standard model. We show that the scheme is not secure against malicious KGC attacks.[2]

3.1 LAS Scheme review

We first review the LAS scheme which is constructed from Waters' ID-based encryption (IBE) scheme [21] by using a *message authentication code* and an *encapsulation* scheme as building blocks.

A message authentication code is a pair of PPT algorithms (Mac, Vrfy) such that Mac takes as input a key sk and a message m to produce a tag tag. The algorithm Vrfy takes as input a key sk, a message m and tag and outputs either `accept` or `reject`. It is required that for all sk and m, $\mathsf{Vrfy}(sk, m, \mathsf{Mac}(sk, m)) =$ `accept`. In addition, an encapsulation scheme is a weak variant of commitment and is defined by a triple of PPT algorithms (Init,S,R) as follow. On input security parameter k', Init outputs some public parameters pub. On input k' and pub, S outputs com, dec on some appropriate range and a string $r \in \{0,1\}^{k'}$. On input pub, com and dec, R outputs r.

The LAS scheme is as follows.

- $\mathsf{Setup}(1^{k'})$. The KGC selects groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p such that an admissible pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ can be constructed. Let g be a generator of $\mathbb{G}$. Randomly pick $\alpha \leftarrow \mathbb{Z}_p$, $g_2 \leftarrow \mathbb{G}_1$, and compute $g_1 = g^\alpha$ and $pub = \mathsf{Init}(k')$. Also randomly select $u', g_1', h_1 \leftarrow \mathbb{G}$ and $u_i \leftarrow \mathbb{G}$ for $i = 1, \ldots, n$. Let $U = \{u_i\}$. The public parameters mpk are $(e, \mathbb{G}, \mathbb{G}_T, g, g_1, g_2, u', g_1', h_1, U, pub)$ and the master secret key msk is g_2^α.
- $\mathsf{PSK}(mpk, msk, \mathsf{ID})$. Let ID be a bit string of length n and $\mathsf{ID}[i]$ be the i-th bit. Define $\mathcal{U} \subset \{1, \ldots, n\}$ to be the set of indices i such that $\mathsf{ID}[i] = 1$. The KGC picks a random value $r \in \mathbb{Z}_p^*$ and computes;

$$(psk_1, psk_2) = (g_2^\alpha \cdot F_u(\mathsf{ID})^r, g^r) \qquad \text{where } F_u(\mathsf{ID}) = u' \prod_{i \in \mathcal{U}} u_i.$$

A user with an identity ID is given $psk_{\mathsf{ID}} = (psk_1, psk_2)$ as a partial private key.

[2]We should mention that the authors in [19] did not claim that the LAS scheme is secure against malicious KGC attacks. They just proved the security of the LAS scheme in the security models which do not consider the malicious KGC. Our aim is to show that our proposed scheme is the first construction secure against malicious KGC attacks in the standard.

- $\mathsf{UKeyGen}(mpk, psk_{\mathsf{ID}})$. User selects a secret value $x \in \mathbb{Z}_p$, sets his private key $sk_{\mathsf{ID}} = (sk_1, sk_2, sk_3) = (psk_1, psk_2, x)$ and computes his public key pk_{ID} as $(g^x, g_1^x) = (pk_1, pk_2)$

- $\mathsf{Enc}(mpk, \mathsf{ID}, pk_{\mathsf{ID}}, m)$. To encrypt a message $m \in \{0,1\}^{\kappa'}$, where $\kappa' = \lfloor \frac{\kappa-1}{2} \rfloor$ and κ is the number of bit representing $\mathbb{G}_T$, for an identity ID and public key $pk_{\mathsf{ID}} = (pk_1, pk_2)$, first check whether $pk_1, pk_2 \in \mathbb{G}$ and $e(pk_1, g_1) = e(pk_2, g)$. If not, output `reject` and abort encryption. Otherwise, run $\mathsf{S}(pub)$ to obtain $(r \in \{0,1\}^{k'}, com \in \mathbb{Z}_p, dec \in \{0,1\}^{\kappa'})$ and set $M = m||dec$. Randomly select $s \in_R \mathbb{Z}_p$, and compute

$$C_1 = e(pk_2, g_2)^s \cdot M \qquad C_2 = g^s \qquad C_3 = F_u(\mathsf{ID})^s \qquad C_4 = ({g'_1}^{com} h_1)^s.$$

  Let $\hat{C} = (C_1, C_2, C_3, C_4)$. Compute $tag = \mathsf{Mac}(r, \hat{C})$. The ciphertext is $C = (\hat{C}, com, tag)$.

- $\mathsf{Dec}(mpk, sk_{\mathsf{ID}}, C)$. On receiving C, compute $M = C_1 \left(\frac{e(psk_2, C_3) e(g, C_4)}{e(psk_1 \cdot {g'_1}^{com} h_1, C_2)} \right)^x$ and obtain m and dec. Compute $r = \mathsf{R}(pub, com, dec)$. If $\mathsf{Vrfy}(r, \hat{C}, tag) = $ `accept`, then the plaintext is m, else output `invalid ciphertext`.

3.2 Analysis

In this section, we show that the LAS scheme is vulnerable to malicious KGC attacks. Liu *et al.* proved the security of the LAS scheme in the most common security models with the assumption that the KGC starts launching Type II attacks only after it has honestly generated a master public/secret key pair. However, the KGC can maliciously generate the public parameters in a real environment as follows and then it can decrypt every ciphertext.

- Setup. When generating the public parameters, the malicious KGC computes u' and u_i as follows:

  1. Select random values β and μ_i in $\mathbb{Z}_p$ for $i = 1, \ldots, n$.
  2. Compute $u' = g_2^{\beta}$ and $u_i = g_2^{\mu_i}$ for $i = 1, \ldots, n$.

  It then publishes $mpk = (e, \mathbb{G}, \mathbb{G}_T, g, g_1, g_2, u', g'_1, h_1, U, pub)$ and securely keeps $(\beta, \mu_1, \ldots, \mu_n)$ with msk.

- Dec. Eavesdropping a ciphertext $C = (\hat{C}, com, tag)$ where $\hat{C} = (C_1, C_2, C_3, C_4)$, the KGC can obtain the encoded message M by computing:

  1. $g_2^s \leftarrow (C_3)^{1/(\beta + \sum_{i \in \mathcal{U}} \mu_i)}$ $\quad (C_3 = F_u(\mathsf{ID})^s = (u' \prod_{i \in \mathcal{U}} u_i)^s = (g_2^{\beta + \sum_{i \in \mathcal{U}} \mu_i})^s = (g_2^s)^{\beta + \sum_{i \in \mathcal{U}} \mu_i}.)$
  2. $M \leftarrow C_1 / e(pk_2, g_2^s)$

The scheme in [11] is of similar structure as the one in [19]. The same analysis technique can be applied.

4 New Construction

We construct a CCA-secure CL-PKE scheme against Type I and Type II adversaries in our modified model. Our scheme is constructed by a similar way to [11]. We apply the techniques of [8] to the 2-level hierarchical extension of Waters' IBE [21] to achieve the CCA-security as in [11].

However, there are important differences between both of constructions. While in [11] the KGC generates all the parameters for 2-level extension of Waters' IBE, in our scheme the KGC generates parameters for the first level as the public parameters and a user generates parameters for the second level as his public key, and then the first level parameters are used to generate a partial private key for an identity of a user by KGC and the second level parameters are used to check the validity of a ciphertext by a user. The main difficulty to prove the security against the malicious KGC attacks is to simulate the ciphertexts for a target identity in the Type II security game, because the adversary (malicious KGC) can handle every parameter except for the public key of the target identity. We use the second level parameters generated by a user to handle the decryption queries for the target identity.

In addition, our analysis technique in Section 3.2 can be directly applied to [11], since the user key generation algorithm of [11] has the same as that of the LAS scheme and the KGC generates the parameters for an identity vector. To prevent this attack, our scheme uses a different public key generation algorithm. Our construction follows.

- $\mathsf{Setup}(1^k)$. The KGC chooses groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p such that an admissible paring $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ can be constructed and selects a generator g of $\mathbb{G}$. Then it picks random values $\alpha, \beta, \mu', \mu_1, \ldots, \mu_n$ in $\mathbb{Z}_p^*$ and computes $g_1 = g^{\alpha}, h = e(g^{\alpha}, g^{\beta}), u' = g^{\mu'}, u_1 = g^{\mu_1}, \ldots, u_n = g^{\mu_n}$, where n is the length of an identity in binary string representation. Let $H : \{0,1\}^* \to \{0,1\}^n$ be a collision-resistant hash function. The public parameters mpk and the mater secret key msk are set as
$$mpk \leftarrow (e, \mathbb{G}, \mathbb{G}_T, g, g_1, h, u', u_1, \ldots, u_n, H), \quad msk \leftarrow (\alpha, \beta, \mu', \mu_1, \ldots, \mu_n).$$

- $\mathsf{PSK}(mpk, msk, \mathsf{ID})$. Let ID be a bit string of length n and $\mathsf{ID}[i]$ be the i-th bit. Define $\mathcal{U} \subset \{1, \ldots, n\}$ to be the set of indices i such that $\mathsf{ID}[i] = 1$. The KGC picks a random value $r \in \mathbb{Z}_p^*$ and computes;
$$(psk_1, psk_2) = (g_1^{\beta} \cdot F_u(\mathsf{ID})^r, g^r) \qquad \text{where } F_u(\mathsf{ID}) = u' \prod_{i \in \mathcal{U}} u_i.$$
A user with an identity ID is given $psk_{\mathsf{ID}} = (psk_1, psk_2)$ as a partial private key.

- $\mathsf{UKeyGen}(mpk, psk_{\mathsf{ID}})$. Pick a secret value $x_{\mathsf{ID}} \in \mathbb{Z}_p^*$ and a random vector $\mathbb{V}_{\mathsf{ID}} = (v', v_1, \ldots, v_n) \in \mathbb{G}^{n+1}$. The public key pk_{ID} is generated as;
$$pk_{\mathsf{ID}} = (X_{\mathsf{ID}}, \mathbb{V}_{\mathsf{ID}}) = (pk_1, pk_2) \qquad \text{where } X_{\mathsf{ID}} = h^{x_{\mathsf{ID}}}$$
Then it selects a random value r' in $\mathbb{Z}_p^*$ and computes the private key $sk_{\mathsf{ID}} = (sk_1, sk_2)$ as;
$$(sk_1, sk_2) = (psk_1^{x_{\mathsf{ID}}} \cdot F_u(\mathsf{ID})^{r'}, psk_2^{x_{\mathsf{ID}}} \cdot g^{r'}) = (g_1^{\beta x_{\mathsf{ID}}} \cdot F_u(\mathsf{ID})^t, g^t)$$
where $t = r x_{\mathsf{ID}} + r'$.

- $\mathsf{Enc}(mpk, \mathsf{ID}, pk_{\mathsf{ID}}, m)$. To encrypt $m \in \mathbb{G}_T$, first check whether the public key pk_{ID} is correct by $X_{\mathsf{ID}}^p = 1_{\mathbb{G}_T}$. If not, output `reject` and abort the algorithm. Otherwise, select a random value $s \in \mathbb{Z}_p^*$ and compute: (Let w be a n-bit string and w_i the i-th bit of w.)
$$C = (C_0, C_1, C_2, C_3) = (m \cdot (X_{\mathsf{ID}})^s, g^s, F_u(\mathsf{ID})^s, F_{\mathbb{V}_{\mathsf{ID}}}(w)^s)$$
where $w = H(C_0, C_1, C_2, \mathsf{ID}, pk_{\mathsf{ID}}) \in \{0,1\}^n$ and $F_{\mathbb{V}_{ID}}(w) = v' \prod_{j=1}^{n} v_j^{w_j}$.

- $\mathsf{Dec}(mpk, sk_{\mathsf{ID}}, C)$. On receiving a ciphertext $C = (C_0, C_1, C_2, C_3)$, check that
$$e(C_1, F_u(\mathsf{ID}) \cdot F_{\mathbb{V}_{\mathsf{ID}}}(w)) = e(g, C_2 C_3)$$
where $w = H(C_0, C_1, C_2, \mathsf{ID}, pk_{\mathsf{ID}}) \in \{0,1\}^n$. If not, output `reject`. Otherwise, compute
$$m = C_0 \cdot e(C_2, sk_2)/e(C_1, sk_1).$$

REMARK: The master keys except for α, β do not need to be generated in practical systems. Actually, $(u', u_1, \ldots, u_n)$ can be randomly selected from $\mathbb{G}$. However, $(\mu', \mu_1, \ldots, \mu_n)$ are required to simulate the security game. In addition, to guarantee the security against the *malicious* KGC, a user in our construction should generate the public key $\mathbb{V}_{\mathsf{ID}}$ by itself as mentioned above, while the KGC generates $\mathbb{V}$ as the public parameters in the Waters' 2-level IBE.

Security Proof. Here, we show that our construction above is secure against a malicious KGC under the Decisional Bilinear Diffie-Hellman (DBDH) assumption in the standard model.

Theorem 1 *Let $\mathcal{A}_{II}$ be a Type II adversary that makes at most q_d decryption queries, q_{pk} public key queries, then we have*

$$Adv_{\mathcal{A}_{II}}^{\mathit{IND\text{-}CL\text{-}CCA}} \leq 8q_{pk}q_d(n+1) \cdot Adv_{\mathcal{A}'}^{\mathit{DBDH}}(k) + 2q_{pk} \cdot Adv_{\mathcal{A}''}^{\mathit{CR}}(k)$$

where $\mathcal{A}'$ and $\mathcal{A}''$ are algorithms that run in approximately the same time as $\mathcal{A}_{II}$.

Proof. Our proof uses the game hopping proof technique [5]. We define a sequence of modified attack games. Each of the games operates on the same underlying probability space. The attacker attempts to distinguish a hidden bit $\mathfrak{b}$ and eventually outputs a guess $\mathfrak{b}'$, where the hidden bit $\mathfrak{b}$ takes on identical values across all games, while some of the rules that define how the environment responds to oracle queries may differ from game to game.

We let S_i be the event that $\mathfrak{b} = \mathfrak{b}'$ in the Game i and Adv_i denote the adversary's advantage in the Game i. Then, $Adv_i = |\Pr[S_i] - 1/2|$. We start from the Game 1 and show from the definition of Game i for $i > 1$ that $|\Pr[S_i] - 1/2|$ is negligible if and only if $|\Pr[S_{i-1}] - 1/2|$ is negligible. Let E be an event that can occur during the execution of the adversary and it is independent of S_i (i.e $\Pr[S_i|E] = \Pr[S_i]$). Let Game $i+1$ be the attack environment which is identical to Game i until the moment that E occurs. If E does not occur, the adversary will choose the same bit that it did in Game i (i.e. $\Pr[S_{i+1}|\neg E] = \Pr[S_i|\neg E] = \Pr[S_i]$.) Otherwise, it outputs a random bit $\mathfrak{b}'$ (i.e. $\Pr[S_{i+1}|E] - 1/2$). Then we have

$$\begin{aligned}
\big|\Pr[S_{i+1}] - 1/2\big| &= \big|\Pr[S_{i+1}|E]\Pr[E] + \Pr[S_{i+1}|\neg E]\Pr[\neg E] - 1/2\big| \\
&= \big|\Pr[E]/2 + \Pr[S_i|\neg E]\Pr[\neg E] - 1/2\big| \\
&= \big|(1 - \Pr[\neg E])/2 + \Pr[S_i]\Pr[\neg E] - 1/2\big| \\
&= \Pr[\neg E]\big|\big|\Pr[S_i] - 1/2\big|.
\end{aligned}$$

Therefore, $Adv_{i+1} = \Pr[\neg E] \cdot Adv_i$.

Game 1. This game is identical to the original attack environment. A Type II adversary $\mathcal{A}_{II}$ first outputs (mpk, msk) to the attack environment $\mathcal{B}$ and interacts with $\mathcal{B}$. It issues up to q_{pk}, q_{sk}, and q_d queries to $\mathsf{PK\text{-}Oracle}, \mathsf{SK\text{-}Oracle}$, and $\mathsf{Dec\text{-}Oracle}$ respectively. We define the following sets.

- $\mathsf{pk_L} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_{q_{pk}}\}$: the set of identities queried for public key oracle.
- $\mathsf{sk_L} = \{\mathsf{ID}'_1, \ldots, \mathsf{ID}'_{q_{sk}}\}$: the set of identities queried for private key extract oracle.
- $\mathsf{D_w} = \{w_1, \ldots, w_{q_d}\}$: the set of string $w_j = H(C_0, C_1, C_2, \mathsf{ID}_j, pk_j)$ involved in decryption queries.

$\mathcal{A}_{II}$ selects a target identity/public key pair $(\mathsf{ID}^*, pk_{\mathsf{ID}^*})$ with two equal length messages m_0, m_1, where $\mathsf{ID}^* \notin \mathsf{sk_L}$, and sends them to $\mathcal{B}$. It is given $C^* = (C_0^*, C_1^*, C_2^*, C_3^*)$ as the challenge ciphertext. At this time, we denote $w^* = H(C_0^*, C_1^*, C_2^*, \mathsf{ID}^*, pk_{\mathsf{ID}^*})$.

Game 2. In this game, $\mathcal{B}$ first selects an identity ID_i in $\mathsf{pk_L}$ at random. Let g^a, g^b be random elements such that a, b are unknown to $\mathcal{B}$. Then, it sets X_{ID_i} by $e(g^a, g^b)^{\alpha\beta}$. At this time, x_{ID_i} is regarded as ab because $h = e(g^\alpha, g^\beta)$. It also picks $\kappa \in \{0, \dots, n\}$ and let τ be an integers such that $\tau(n+1) < p$. In addition, it randomly selects vectors $(x', x_1, \dots, x_n)$ in $\mathbb{Z}_\tau$ and $(y', y_1, \dots, y_n)$ in $\mathbb{Z}_p$. Then $\mathbb{V}_{\mathsf{ID}_i}$ is set as:

$$v' = (g^a)^{x' - \kappa\tau} g^{y'}, \qquad v_j = (g^a)^{x_j} g^{y_j} \text{ for } 1 \leq j \leq n.$$

If $\mathcal{A}$ does not select ID_i as a target identity (namely, $\mathsf{ID}_i \neq \mathsf{ID}^*$), then $\mathcal{B}$ aborts. Therefore, $Adv_2 = \frac{1}{q_{pk}} Adv_1$.

Game 3. This game is identical to Game 2 except that the environment halts if the attacker submits a decryption query $(C, \mathsf{ID}, pk_{\mathsf{ID}})$ for a well-formed ciphertext $C = (C_0, C_1, C_2, C_3)$ where w is either equal to the same value as a previously submitted ciphertext or w is equal to w^* in the post challenge phase. For such a legal decryption query, we have $C \neq C^*$ or $(\mathsf{ID}, pk_{\mathsf{ID}}) \neq (\mathsf{ID}^*, pk_{\mathsf{ID}^*})$. In either case, this implies a collision for H. Hence, we can construct an algorithm $\mathcal{A}''$ such $\mid \Pr[S_2] - \Pr[S_3] \mid \leq Adv^{\mathsf{CR}}_{\mathcal{A}''}(k)$.

Game 4. We define the following functions from the values of Game 2 as

$$J(w) = x' + \Sigma_{j=1}^n w_j x_j - \kappa\tau, \qquad K(w) = y' + \Sigma_{j=1}^n w_j y_j$$

taking as input n-bit string $w = w_1 \dots w_n$. Then $F_{\mathbb{V}_{\mathsf{ID}^*}}(w) = v' \prod_{j=1}^n v_j^{w_j} = (g^a)^{J(w)} \cdot g^{K(w)}$.

Game 4 is the same as Game 3 except that, after $\mathcal{A}$ outputs her guess $\mathfrak{b}'$ for $\mathfrak{b}$, $\mathcal{B}$ checks whether $J(w^*) = 0 \bmod p$. If $J(w^*) \neq 0 \bmod p$, then $\mathcal{B}$ aborts and outputs a random bit $\mathfrak{b}'$. The event that $J(w^*) = 0 \bmod p$ happens by chance because $\mathcal{A}_{II}$ does not know information on all values $(x', x_1, \dots, x_n, \kappa, \tau)$ to compute $J(w)$ at all. Actually, $\Pr[J(w^*) = 0 \bmod p] = \Pr[\kappa\tau = (x' + \Sigma_{j=1}^n w_j x_j)]$ since $(x' + \Sigma_{j=1}^n w_j x_j) < \tau(n+1), \kappa\tau < \tau(n+1)$ and $\tau(n+1) < p$. Therefore,

$$\Pr[J(w^*) = 0 \bmod p] = \frac{1}{\tau(n+1)}$$

and $Adv_4 = \frac{1}{\tau(n+1)} Adv_3$.

Game 5. We modify the way the challenge ciphertext is constructed. The environment $\mathcal{B}$ introduces a new variable $c \leftarrow \mathbb{Z}_p^*$ and $C_1^* = g^c$. It flips a coin $\mathfrak{b}$ and computes

$$C_0^* = m_{\mathfrak{b}} \cdot X_{\mathsf{ID}^*}^c \qquad C_2^* = C_1^{*U_{\mathsf{ID}^*}} = (g^c)^{U_{\mathsf{ID}^*}} \qquad C_3^* = C_1^{*K(w^*)} = (g^c)^{K(w^*)}$$

where $w^* = H(C_0^*, C_1^*, C_2^*, \mathsf{ID}^*, pk_{\mathsf{ID}^*})$ and $U_{\mathsf{ID}^*} = \sum_{i \in \mathcal{U}} \mu_j$. We clearly have $Adv_5 = Adv_4$.

Game 6. We change Game 5 so that, after $\mathcal{A}$ outputs her guess $\mathfrak{b}'$, the environment $\mathcal{B}$ aborts and replaces $\mathcal{A}$'s output by a random bit $\mathfrak{b}'$ if $J(w_\ell) = 0 \bmod \tau$ for some $w_\ell \in \mathsf{D_w}$ where $\ell \in \{1, \dots, q_d\}$. Since $\Pr[J(w) = 0 \bmod \tau] = 1/\tau$, $Adv_6 = (1 - \frac{1}{\tau})^{q_d} \cdot Adv_5 \geq (1 - \frac{q_d}{\tau}) \cdot Adv_5$ as $\Pr[J(w) = 0 \bmod \tau] = 1/\tau$. If we set $\tau = 2q_d$, then $Adv_6 \geq \frac{1}{2} Adv_5$.

Game 7. We effectively change the treatment of $\mathcal{A}$'s queries. For all public key queries, private key queries and decryption key queries not involving ID^*, $\mathcal{B}$ can respond to queries by running the algorithms $\mathsf{PSK}(mpk, msk, \mathsf{ID})$, $\mathsf{UKeyGen}(mpk, psk_{\mathsf{ID}})$ and $\mathsf{Dec}(mpk, sk_{\mathsf{ID}}, C)$. In addition, it responds to all decryption queries involving ID^* as follows. When it receives decryption queries for a valid ciphertext (C_0, C_1, C_2, C_3) for ID^*, $\mathcal{B}$ aborts and outputs a random bit $\mathfrak{b}'$ as in Game 6 if $J(w) = 0 \bmod \tau$, Otherwise,

$\mathcal{B}$ can extract m by computing

$$\begin{aligned}
&w \leftarrow H(C_0, C_1, C_2, \mathsf{ID}^*, pk_{\mathsf{ID}^*})\\
&(g^a)^s \leftarrow (C_3/C_1^{K(w)})^{1/J(w)}\\
&m \leftarrow C_0/e(g^{as}, g^b)^{\alpha\beta} = C_0/e(g^\alpha, g^\beta)^{abs} = C_0/X_{\mathsf{ID}^*}^s.
\end{aligned}$$

Note that we can compute $(C_3/C_1^{K(w)})^{1/J(w)}$, since $J(w) \neq 0 \bmod p$ if $J(w) \neq 0 \bmod \tau$. We observe that $\mathcal{B}$ correctly answers $\mathcal{A}$'s queries as in Game 6. This implies $Adv_7 = Adv_6$.

Game 8. We again alter the generation of the challenge ciphertext. For a variable c introduced in Game 5, let $C_1^* = g^c$ and $Z = e(g^a, g^b)^c$. $\mathcal{B}$ retrieves values α, β, flips a coin $\mathfrak{b}$, and computes

$$C_0^* = m_{\mathfrak{b}} \cdot Z^{\alpha\beta}, \qquad C_2^* = (g^c)^{U_{\mathsf{ID}^*}}, \qquad C_3^* = (g^c)^{K(w^*)}$$

where $w^* = H(C_0^*, C_1^*, C_2^*, \mathsf{ID}^*, pk_{\mathsf{ID}^*})$. We have $Adv_8 = Adv_7$.

Game 9. We again alter the challenge phase. This time, the environment "forgets" the value c and simply retains C_1^*. The challenge ciphertext is constructed as in Game 8 but using a randomly chosen $Z \in \mathbb{G}_T$ this time. The whole simulation only depends on the values g^a, g^b, g^c and the simulator does not use a, b, c at all. Therefore, $|\Pr[S_8] - \Pr[S_9]| \leq Adv_{\mathcal{A}'}^{\mathtt{DBDH}}(k)$ and $\Pr[S_9] = 1/2$.

$$\begin{aligned}
Adv_6 = Adv_7 = Adv_8 \leq Adv_{\mathcal{A}'}^{\mathtt{DBDH}}(k)\\
Adv_4 = Adv_5 \leq 2 \cdot Adv_6
\end{aligned}$$

Since $Adv_4 = Adv_3/(\tau(n+1))$ and $\tau = 2q_d$, we get

$$Adv_3 \leq 4q_d(n+1) \cdot Adv_{\mathcal{A}'}^{\mathtt{DBDH}}(k)$$

$$Adv_2 \leq Adv_{\mathcal{A}''}^{\mathtt{CR}}(k) + Adv_3 \leq 4q_d(n+1) \cdot Adv_{\mathcal{A}'}^{\mathtt{DBDH}}(k) + Adv_{\mathcal{A}''}^{CR}(k)$$

In consequence, since $Adv_2 = q_{pk} \cdot Adv_1$, we obtain

$$Adv_1 \leq 8q_{pk}q_d(n+1) \cdot Adv_{\mathcal{A}'}^{\mathtt{DBDH}}(k) + 2q_{pk} \cdot Adv_{\mathcal{A}''}^{\mathtt{CR}}(k).$$

We complete the proof of the theorem. $\square$

We can also provide the security proof of our scheme against Type I adversaries by a similar way to that of Theorem 2.

Theorem 2 *Let $\mathcal{A}_I$ be a Type I adversary that makes at most q_d decryption queries, q_{pk} public key queries, then we have*

$$Adv_{\mathcal{A}_I}^{\mathit{IND\text{-}CL\text{-}CCA}} \leq 8q_{pk}q_d(n+1) \cdot Adv_{\mathcal{A}'}^{\mathit{DBDH}}(k) + 2q_{pk} \cdot Adv_{\mathcal{A}''}^{\mathit{CR}}(k)$$

where $\mathcal{A}'$ and $\mathcal{A}''$ are algorithms that run in approximately the same time as $\mathcal{A}_I$.

Proof. **Game 1.** This game is identical to the original attack environment. The environment $\mathcal{B}$ runs $\mathsf{Setup}(1^k)$ and outputs (mpk, msk). A Type I adversary $\mathcal{A}_I$ is given mpk. Then it issues up to q_{psk}, q_{pk}, q_{sk}, and q_d queries to PSK-Oracle, PK-Oracle, SK-Oracle, and Dec-Oracle respectively. We define the following sets.

- $\mathsf{pk_L} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_{q_{pk}}\}$: the set of identities queried for public key oracle.

- $\mathsf{sk_L} = \{\mathsf{ID}'_1, \ldots, \mathsf{ID}'_{q_{sk}}\}$: the set of identities queried for private key extract oracle.
- $\mathsf{D_w} = \{w_1, \ldots, w_{q_d}\}$: the set of string $w_j = H(C_0, C_1, C_2, \mathsf{ID}_j, pk_j)$ involved in decryption queries.

$\mathcal{A}_I$ selects a target identity/public key pair $(\mathsf{ID}^*, pk_{\mathsf{ID}^*})$ with two equal length messages m_0, m_1, where $\mathsf{ID}^* \notin \mathsf{sk_L}$, and sends them to $\mathcal{B}$. It is given $C^* = (C_0^*, C_1^*, C_2^*, C_3^*)$ as the challenge ciphertext. At this time, we denote $w^* = H(C_0^*, C_1^*, C_2^*, \mathsf{ID}^*, pk_{\mathsf{ID}^*})$.

Game 2. This game is identical to Game 1, except that g_1 and h in the system parameters are replaced with following. Let g^a be a random element in $\mathbb{G}$ such that a is unknown to $\mathcal{B}$. It randomly selects $\beta \in \mathbb{Z}_p^*$, and sets $g_1 = g^a, h = (g^a, g^\beta)$. The replaced public key has the same distribution as the public key generated in the previous game. Hence, $\Pr[S_1] = \Pr[S_2]$ and it implies $Adv_2 = Adv_1$. Note that one value α of msk is regarded as a and $\mathcal{B}$ does not know it. However, it securely keeps other values $(\beta, \mu', \mu_1, \ldots, \mu_n)$ of msk.

Game 3. In this game, $\mathcal{B}$ first selects an identity ID_i in $\mathsf{pk_L}$ at random. Let g^b be a random element in $\mathbb{G}$ such that b is unknown to $\mathcal{B}$. Then it sets X_{ID_i} by $e(g^a, g^{b\beta})$. At this time, x_{ID_i} is regarded as b because $h = e(g^a, g^\beta)$. It also picks $\kappa_v \in \{0, \ldots, n\}$ and let τ_v be an integers such that $\tau_v(n+1) < p$. In addition, it randomly selects vectors $(x'_v, x_{v,1}, \ldots, x_{v,n})$ in $\mathbb{Z}_{\tau_v}$ and $(y'_v, y_{v,1}, \ldots, y_{v,n})$ in $\mathbb{Z}_p$. Then $\mathbb{V}_{\mathsf{ID}_i}$ is set as:

$$v' = (g^a)^{x'_v - \kappa_v \tau_v} g^{y'_v}, \qquad v_j = (g^a)^{x_{v,j}} g^{y_{v,j}} \text{ for } 1 \le j \le n.$$

If $\mathcal{A}$ does not select ID_i as a target identity (namely, $\mathsf{ID}_i \neq \mathsf{ID}^*$), then $\mathcal{B}$ aborts. Therefore, $Adv_3 = \frac{1}{q_{pk}} Adv_2$.

Game 4. This game is identical to Game 3 except that the environment halts if the attacker submits a decryption query $(C, \mathsf{ID}, pk_{\mathsf{ID}})$ for a well-formed ciphertext $C = (C_0, C_1, C_2, C_3)$ where w is either equal to the same value as a previously submitted ciphertext or w is equal to w^* in the post challenge phase. For such a legal decryption query, we have $C \neq C^*$ or $(\mathsf{ID}, pk_{\mathsf{ID}}) \neq (\mathsf{ID}^*, pk_{\mathsf{ID}^*})$. In either case, this implies a collision for H. Hence, we can construct an algorithm $\mathcal{A}''$ such $|\Pr[S_3] - \Pr[S_4]| \le Adv^{\mathsf{CR}}_{\mathcal{A}''}(k)$.

Game 5. We define the following functions from the values of Game 3 as

$$J(w) = x' + \Sigma_{j=1}^n w_j x_j - \kappa\tau, \qquad K(w) = y' + \Sigma_{j=1}^n w_j y_j$$

taking as input n-bit string $w = w_1 \ldots w_n$. Then $F_{\mathbb{V}_{ID^*}}(w) = v' \prod_{j=1}^n v_j^{w_j} = (g^a)^{J(w)} \cdot g^{K(w)}$.

Game 5 is the same as Game 4 except that, after $\mathcal{A}$ outputs her guess $\mathfrak{b}'$ for $\mathfrak{b}$, $\mathcal{B}$ checks whether $J(w^*) = 0 \bmod p$. If $J(w^*) \neq 0 \bmod p$, then $\mathcal{B}$ aborts and outputs a random bit $\mathfrak{b}'$. The event that $J(w^*) = 0 \bmod p$ happens by chance because $\mathcal{A}_{II}$ does not know information on all values $(x', x_1, \ldots, x_n, \kappa, \tau)$ to compute $J(w)$ at all. Actually, $\Pr[J(w^*) = 0 \bmod p] = \Pr[\kappa\tau = (x' + \Sigma_{j=1}^n w_j x_j)]$ since $(x' + \Sigma_{j=1}^n w_j x_j) < \tau(n+1), \kappa\tau < \tau(n+1)$ and $\tau(n+1) < p$. Therefore,

$$\Pr[J(w^*) = 0 \bmod p] = \frac{1}{\tau(n+1)}$$

and $Adv_5 = \frac{1}{\tau(n+1)} Adv_4$.

Game 6. We modify the way the challenge ciphertext is constructed. The environment $\mathcal{B}$ introduces a new variable $c \leftarrow \mathbb{Z}_p^*$ and $C_1^* = g^c$. It flips a coin $\mathfrak{b}$ and computes

$$C_0^* = m_{\mathfrak{b}} \cdot X^c_{\mathsf{ID}^*} \qquad C_2^* = C_1^{*U_{\mathsf{ID}^*}} = (g^c)^{U_{\mathsf{ID}^*}} \qquad C_3^* = C_1^{*K(w^*)} = (g^c)^{K(w^*)}$$

where $w^* = H(C_0^*, C_1^*, C_2^*, \mathsf{ID}^*, pk_{\mathsf{ID}^*})$ and $U_{\mathsf{ID}^*} = \sum_{i \in \mathcal{U}} \mu_j$. We clearly have $Adv_6 = Adv_5$.

Game 7. We change Game 6 so that, after $\mathcal{A}$ outputs her guess $\mathfrak{b}'$, the environment $\mathcal{B}$ aborts and replaces $\mathcal{A}$'s output by a random bit $\mathfrak{b}'$ if $J(w_\ell) = 0 \bmod \tau$ for some $w_\ell \in \mathsf{D_w}$ where $\ell \in \{1, \ldots, q_d\}$. Since $\Pr[J(w) = 0 \bmod \tau] = 1/\tau$, $Adv_6 = (1 - \frac{1}{\tau})^{q_d} \cdot Adv_5 \geq (1 - \frac{q_d}{\tau}) \cdot Adv_5$ as $\Pr[J(w) = 0 \bmod \tau] = 1/\tau$. If we set $\tau = 2q_d$, then $Adv_7 \geq \frac{1}{2} Adv_6$.

Game 8. We effectively change the treatment of $\mathcal{A}$'s queries. For all public key queries, private key queries and decryption key queries not involving ID^*, $\mathcal{B}$ can respond to queries by running the algorithms $\mathsf{PSK}(mpk, msk, \mathsf{ID})$ [3], $\mathsf{UKeyGen}(mpk, psk_{\mathsf{ID}})$ and $\mathsf{Dec}(mpk, sk_{\mathsf{ID}}, C)$. When it receives a public key replace query, it replaces a previously generated public key pk_{ID} for ID with a new one pk'_{ID}. In addition, it responds to all decryption queries involving ID^* as follows. When it receives decryption queries for a valid ciphertext (C_0, C_1, C_2, C_3) for ID^*, $\mathcal{B}$ aborts and outputs a random bit $\mathfrak{b}'$ as in Game 6 if $J(w) = 0 \bmod \tau$, Otherwise, $\mathcal{B}$ can extract m by computing

$$\begin{aligned} & w \leftarrow H(C_0, C_1, C_2, \mathsf{ID}^*, pk_{\mathsf{ID}^*}) \\ & (g^a)^s \leftarrow (C_3 / C_1^{K(w)})^{1/J(w)} \\ & m \leftarrow C_0 / e(g^{as}, g^b)^\beta = C_0 / e(g^a, g^{b\beta})^s = C_0 / X_{\mathsf{ID}^*}^s. \end{aligned}$$

Note that we can compute $(C_3 / C_1^{K(w)})^{1/J(w)}$, since $J(w) \neq 0 \bmod p$ if $J(w) \neq 0 \bmod \tau$. We observe that $\mathcal{B}$ correctly answers $\mathcal{A}$'s queries as in Game 6. This implies $Adv_8 = Adv_7$.

Game 9. We again alter the generation of the challenge ciphertext. For a variable c introduced in Game 5, let $C_1^* = g^c$ and $Z = e(g^a, g^b)^c$. $\mathcal{B}$ retrieves values α, β, flips a coin $\mathfrak{b}$, and computes

$$C_0^* = m_{\mathfrak{b}} \cdot Z^\beta, \qquad C_2^* = (g^c)^{U_{\mathsf{ID}^*}}, \qquad C_3^* = (g^c)^{K(w^*)}$$

where $w^* = H(C_0^*, C_1^*, C_2^*, \mathsf{ID}^*, pk_{\mathsf{ID}^*})$. We have $Adv_9 = Adv_8$.

Game 10. We again alter the challenge phase. This time, the environment "forgets" the value c and simply retains C_1^*. The challenge ciphertext is constructed as in Game 9 but using a randomly chosen $Z \in \mathbb{G}_T$ this time. The whole simulation only depends on the values g^a, g^b, g^c and the simulator does not use a, b, c at all. Therefore, $|\Pr[S_9] - \Pr[S_{10}]| \leq Adv_{\mathcal{A}'}^{DBDH}(k)$ and $\Pr[S_{10}] = 1/2$.

$$\begin{gathered} Adv_7 = Adv_8 = Adv_9 \leq Adv_{\mathcal{A}'}^{\mathtt{DBDH}}(k) \\ Adv_5 = Adv_6 \leq 2 \cdot Adv_7 \end{gathered}$$

Since $Adv_5 = Adv_4 / (\tau(n+1))$ and $\tau = 2q_d$, we get

$$Adv_4 \leq 4q_d(n+1) \cdot Adv_{\mathcal{A}'}^{\mathtt{DBDH}}(k)$$

$$Adv_3 \leq Adv_{\mathcal{A}''}^{\mathtt{CR}}(k) + Adv_4 \leq 4q_d(n+1) \cdot Adv_{\mathcal{A}'}^{\mathtt{DBDH}}(k) + Adv_{\mathcal{A}''}^{\mathtt{CR}}(k)$$

In consequence, since $Adv_3 = q_{pk} \cdot Adv_2$ and $Adv_1 = Adv_2$, we obtain

$$Adv_1 \leq 8q_{pk}q_d(n+1) \cdot Adv_{\mathcal{A}'}^{\mathtt{DBDH}}(k) + 2q_{pk} \cdot Adv_{\mathcal{A}''}^{\mathtt{CR}}(k).$$

We complete the proof of the theorem. $\square$

[3] To run the algorithm PSK, the environment $\mathcal{B}$ needs msk as input and is not given a part α (or a) of msk during the simulation. However, $\mathcal{B}$ can generate psk_{ID} for a queried ID, since only β of msk is actually required to run PSK as input. Hence, we can replace $\mathsf{PSK}(mpk, msk, \mathsf{ID})$ with $\mathsf{PSK}(mpk, \beta, \mathsf{ID})$

5 Concluding Remarks

In this paper, we have showed that some previous CL-PKE scheme in the standard model are not secure against the malicious KGC attacks. In addition, we proposed a new CL-PKE scheme which is provably secure against the malicious KGC attacks in the standard model. It is believed to be the first in the literature to achieve the strongest Type II security without random oracles.

We remark that one may also construct a certificateless signature scheme secure against malicious KGC in the standard model using a similar technique presented in this paper, which will be our future work.

References

[1] S. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In *ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer-Verlag, 2003.

[2] S. S. Al-Riyami and K. G. Paterson. CBE from CL-PKE: A generic construction and efficient schemes. In *PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 398–415. Springer, 2005.

[3] M. Au, J. Chen, J. Liu, Y. Mu, D. Wong, and G. Yang. Malicious KGC attacks in certificateless cryptography. In *ASIACCS 2007*, pages 302–311. ACM Press, 2007. This paper can be found on eprint `http://eprint.iacr.org/2006/255/`.

[4] J. Baek, R. Safavi-Naini, and W. Susilo. Certificateless public key encryption without pairing. In *ISC 2005*, volume 3650 of *Lecture Notes in Computer Science*, pages 134–148. Springer-Verlag, 2005.

[5] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT 2004*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer-Verlag, 2006.

[6] K. Bentahar, P. Farshim, J. Malone-Lee, and N. P. Smart. Generic construction of identity-based and certificateless KEMs. To Appear, Journal of Cryptology, 2007. Also appear in eprint: `http://eprint.iacr.org/2005/058`.

[7] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[8] X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. In *the 12th ACM Conference on Computer and Communications Security*, pages 320–329. ACM Press, 2005.

[9] Z. Cheng and R. Comley. Efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/012, 2005. `http://eprint.iacr.org/2005/012/`.

[10] S. Chow, C. Boyd, and J. Gonzalez. Security-mediated certificateless cryptography. In *PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 508–524. Springer-Verlag, 2006.

[11] A. Dent, B. Libert, and K. Paterson. Certificateless encryption schemes strongly secure in the standard model. Cryptology ePrint Archive, Report 2007/121, 2007. `http://eprint.iacr.org/2007/121/`.

[12] A. W. Dent. A survey of certificateless encryption schemes and security models. Cryptology ePrint Archive, Report 2006/211, 2006. `http://eprint.iacr.org/2006/211`.

[13] Y. Dodis and J. Katz. Chosen-ciphertext security of multiple encryption. In *TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 188–209. Springer-Verlag, 2005.

[14] D. Galindo, P. Morillo, and C. Ràfols. Breaking Yum and Lee generic constructions of certificate-less and certificate-based encryption schemes. In *EuroPKI 2006*, volume 4043 of *Lecture Notes in Computer Science*, pages 81–91. Springer, 2006.

[15] C. Gentry. Certificate-based encryption and the certificate revocation problem. In *EUROCRYPT 2003*, volume 3958 of *Lecture Notes in Computer Science*, pages 272–293. Springer-Verlag, 2003.

[16] B. Hu, D. Wong, Z. Zhang, and X. Deng. Key replacement attack against a generic construction of certificateless signature. In *ACISP 2006*, volume 4058 of *Lecture Notes in Computer Science*, pages 235–246. Springer-Verlag, 2006.

[17] X. Huang, W. Susilo, Y. Mu, and F. Zhang. On the security of certificateless signature schemes from Asiacrypt 2003. In *CANS 2005*, volume 3810 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 2005.

[18] B. Libert and J. Quisquater. On constructing certificateless cryptosystems from identity based encryption. In *PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 474–490. Springer-Verlag, 2006.

[19] J. Liu, M. Au, and W. Susilo. Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model. In *ASIACCS 2007*, pages 273–283. ACM Press, 2007. Full Version of this paper can be found on eprint `http://eprint.iacr.org/2006/373/`.

[20] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1984.

[21] B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer-Verlag, 2005.

[22] D. H. Yum and P. J. Lee. Generic construction of certificateless encryption. In *ICCSA 2004*, volume 3040 of *Lecture Notes in Computer Science*, pages 802–811. Springer-Verlag, 2004.

[23] D. H. Yum and P. J. Lee. Generic construction of certificateless signature. In *ACISP 2004*, volume 3108 of *Lecture Notes in Computer Science*, pages 200–211. Springer-Verlag, 2004. LNCS No. 3108.

[24] D. H. Yum and P. J. Lee. Identity-based cryptography in public key management. In *EuroPKI 2004*, volume 3093 of *Lecture Notes in Computer Science*, pages 71–84. Springer, 2004.

[25] Z. Zhang, D. Wong, J. Xu, and D. Feng. Certificateless public-key signature: Security model and efficient construction. In *ACNS 2006*, volume 3989 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2006.