Malicious KGC Attack in Certificateless Cryptography

Man Ho Au[1], Jing Chen[2], Joseph K. Liu[3], Yi Mu[1], Duncan S. Wong[2]⋆, and Guomin Yang[2]

[1] Centre for Information Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong, Australia
{mhaa456,ymu}@uow.edu.au
[2] Department of Computer Science
City University of Hong Kong
Hong Kong, China
{jingchen,duncan,csyanggm}@cs.cityu.edu.hk
[3] Department of Computer Science
University of Bristol
Bristol, UK
liu@cs.bris.ac.uk

Abstract. Identity-based cryptosystems have an inherent key escrow issue, that is, the Key Generation Center (KGC) always knows user secret key. Thus, if the KGC is malicious, it can always impersonate the user. Certificateless cryptography, introduced by Al-Riyami and Paterson in 2003, is intended to solve this problem. However, in all the previously proposed certificateless schemes, it is always assumed that the malicious KGC starts launching attacks (so-called Type II attacks) only after it has generated a master public/secret key pair honestly. In this paper, we propose new security models that remove this assumption for both certificateless signature and encryption schemes. Under the new models, we show that some previously proposed certificateless encryption/signature schemes still have the key escrow problem, while some other schemes do not. We also give new proofs for the schemes in the latter case.

1 Introduction

Certificateless cryptography, introduced by Al-Riyami and Paterson in 2003 [1], is intended to solve the key escrow problem which is inherent in identity-based (ID-based) cryptography [18,7], while at the same time, eliminate the use of certificates as in the conventional Public Key Infrastructure (PKI), which is generally considered to be costly to use and manage.

In a certificateless cryptosystem, a Key Generation Center (KGC) is involved in issuing *user partial key* to user whose identity is assumed to be unique in the system. The user also *independently* generates an additional user public/secret key pair. Cryptographic operations can then be performed successfully only when both the user partial key and the user secret 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. There are two types of attacks that are generally considered in certificateless cryptography:

Type I - Key Replacement Attack. A third party tries to impersonate a user after compromising the user secret key and/or *replacing* the user public key with some value chosen by the third party. However, it does not know the user partial key.

Type II - Malicious KGC Attack. The KGC, who knows the partial key of a user, is malicious and tries to impersonate the user. However, the KGC does not know the user secret key or being able to replace the user public key.

⋆ The author was supported by a grant from CityU (Project No. 7001844).

A defense against Type II attacks is for solving the key escrow issue that is an inherent problem in ID-based cryptography. 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 secret key, but the KGC knows the user partial key. In addition to this, even though we say that the KGC is *malicious*, we actually assume that the KGC is *passive*, in the sense that the KGC would not actively replace the user public key (which would have been published on a bulletin board in some real implementation) or corrupt the user secret key. For certificateless encryption as example, the malicious KGC may passively eavesdrop the ciphertexts sent to a user and try to decrypt them using its knowledge of the user partial key. In the rest of the paper, we refer to this KGC as *malicious-but-passive* KGC.

Of course, if the malicious KGC is active, that is, the KGC not only knows the user partial key but is also able to replace user public key or compromise user secret key, then the KGC is always able to impersonate the user. This situation also happens in PKI. A malicious and active CA (Certification Authority) can do similar damage to a user in PKI by generating a certificate on a contradictory public key for impersonating the user. With this comparison in mind, for certificateless cryptography, we target to alleviate damage caused by malicious-but-passive KGC rather than eliminate any trust to the KGC.

Now if we take a look at all previously proposed certificateless encryption and signature schemes and adversarial models [1,20,21,19,2,6,3,9,15,22,12,17,10], we will notice that all of them have an implicit assumption that this malicious-but-passive KGC always generates its master public/secret key pair honestly according to scheme specification. In other words, all of them assume that the KGC is originally benign, but once after setting up its own key pair, it suddenly becomes malicious and gets ready to impersonate users.

It seems to be more natural if we consider this malicious-but-passive KGC to have already been malicious at the very beginning of the setup stage of the system. This KGC may generate its master public/secret key pair maliciously so that it can launch the Type II attack more easily in the later stage of the system. The KGC may even have already targeted a particular victim, say the president, when choosing its master key pair. For example, in the Al-Riyami-Paterson certificateless encryption scheme [1], we find that the KGC can have its master key pair specifically generated so that all the encrypted messages for the president can also be decrypted by the KGC. This is because the KGC is able to derive the user secret key generated by the president once after the president has published the user public key (details are in Sec. 3 of this paper). In addition, this specifically generated master public key is computationally indistinguishable from an honestly generated master public key. Therefore, it is infeasible for the president to find out that he is the target of the Type II attack.

1.1 Our Results

In the example above, the KGC can find out the user secret key if the KGC maliciously generates its master public key which is computationally indistinguishable from a key generated honestly according to the master key generation specification of the underlying scheme. This means once we remove the assumption that the KGC must generate its master key pair honestly, the key escrow problem reappears in some of the previously proposed certificateless schemes.

In this paper, we propose to capture the malicious-but-passive KGC attacks by specifying extension and new Type II adversarial models. The new models (one for certificateless encryption schemes and another one for signature schemes) remove the assumption that the KGC must be benign during the master key generation step and when performing user partial key generation. The models also allow the KGC to choose a user to attack during the master key generation stage. We also adopt the notion of simplifying the definition and strengthening the adversarial models of certificateless signature schemes due to Hu, Wong, Zhang and Deng [14] to simplify the definition and strengthen the two adversarial models for certificateless encryption schemes. Our simplified definition of certificateless encryption schemes is composed of five algorithms, while all previous definitions require seven algorithms. The unique feature of certificateless encryption is also maintained, that is, the partial key generation and the user public/secret key pair generation can be carried out by the KGC and the user *independently*. Our new adversarial models also allow the adversary to compromise the target user secret key in Type I model and to replace non-target users' public keys in Type II model. These capabilities are not captured in previously proposed models for certificateless encryption schemes.

We show that the schemes proposed in [1] are vulnerable to malicious-but-passive KGC attacks (i.e. Type II attack) in our new models. The same attack technique can also be applied to schemes in [15,16] as they share the same key structure and generation procedures as that of [1]. On the other side, we give new proofs for the generic certificateless signature scheme proposed by Hu, Wong, Zhang and Deng [14] and the generic encryption scheme proposed by Libert and Quisquater in [17] to show its security under our new models.

1.2 Related Work

Certificateless encryption schemes and signature schemes were first defined and proposed by Al-Riyami and Paterson [1] in 2003. Each of the definitions for certificateless encryption and signature schemes consists of seven algorithms. This definition approach has then been adopted by all others [20,21,19,2,6,3,9,15,22,12,17,10] until a simplified definition for certificateless signature schemes was proposed by Hu, Wong, Zhang and Deng in [14] this year[4]. Their definition consists of only five algorithms and is shown to be more versatile than the previous one while maintaining the unique feature of certificateless cryptography, that is, the user public/secret key pair can be generated *independently* by the user even before obtaining the user partial key from the KGC. In Sec. 2, we adopt their approach and give a five-algorithm definition for certificateless encryption schemes.

In [3], Baek et al. proposed a certificateless encryption scheme that fits in a slightly different model that does not maintain the unique feature of certificateless cryptosystems. In their scheme, the user has to obtain a partial public/private key pair first before being able to generate a user public key. In this paper, we focus ourselves on constructing secure schemes which support the unique feature of certificateless cryptography.

On the security of certificateless encryption schemes and signature schemes, various kinds of security models have been defined. In [17,14,10], nice survey and discussions can be found and therefore, we skip the details in this paper, and only emphasize that all the current security models have the assumption that the KGC starts launching Type II attacks only after it has honestly generated a master public/secret key pair. In Sec. 2, we propose new Type II adversarial models for capturing malicious-but-passive KGC attacks that remove this assumption.

Paper organization. In Sec. 2, the simplified five-algorithm definition for certificateless signature schemes of [14] is reviewed and by following the approach, a new five-algorithm certificateless encryption scheme is defined. New security models for capturing malicious-but-passive KGC are also proposed. In Sec. 3, malicious-but-passive KGC attacks against some previously proposed schemes are described. In Sec. 4, the Hu-Wong-Zhang-Deng generic certificateless signature construction [14] is reviewed. A new proof is given to show its security under our new security model. In Sec. 5, the Libert-Quisquater generic certificateless encryption scheme [17] is also proven secure in our new certificateless encryption security model.

2 Definitions and Security Models

In this section, we first give the definitions of certificateless encryption and certificateless signature schemes. Then, we propose security models for both types of the schemes. The models corresponding to Type II attacks will capture those launched by the malicious-but-passive KGC.

[4] Recently, Dent in [10] also mentioned that the Set-Secret-Value and Set-Public-Key algorithms in the original seven-algorithm definition can be replaced with a single Set-User-Keys algorithm. The approach is the same as one of the two simplifications proposed by Hu, Wong, Zhang and Deng in [14] and can reduce the number of algorithms in the certificateless encryption definition to six.

In the definitions, we adopt the notion introduced by Hu, Wong, Zhang and Deng in [14] for defining schemes as sets of five algorithms rather than seven. As shown in [14], the new approach of defining certificateless signature schemes is more versatile than the original seven-algorithm definition [1,20,15], and still maintains the unique feature of certificateless signature schemes, that is, user partial key generation and user key generation can be done *independently* by the KGC and the user, respectively. In particular, a user with identity ID can generate a user public key denoted by upk_{ID} even before the KGC generates a user partial key denoted by $partial\_key_{ID}$ for the user. In the following, we apply their notion on defining a certificateless encryption scheme.

As the certificateless signature and certificateless encryption schemes are sharing the same set of key generation algorithms, in the following, we first define these key generation algorithms, and then define the specific algorithms for each of signature and encryption schemes.

A *certificateless cryptosystem* has three key generation algorithms: MasterKeyGen, PartialKeyGen, UserKeyGen. All of them are polynomial-time and may be randomized.

1. MasterKeyGen (Master Key Generation): On input 1^k where $k \in \mathbb{N}$ is a security parameter, it generates a master public/secret key pair (mpk, msk).
2. PartialKeyGen (User Partial Key Generation): On input msk and user identity $ID \in \{0,1\}^*$, it generates a user partial key $partial\_key$.
3. UserKeyGen (User Key Generation): On input mpk and user identity ID, it generates a user public/secret key pair (upk, usk).

A certificateless signature (CL-SIG) scheme has two additional polynomial-time algorithms: CL-Sign and CL-Ver.

1. CL-Sign (Signature Generation): On input user secret key usk, user partial key $partial\_key$ and message m, it generates a signature σ.
2. CL-Ver (Signature Verification): On input mpk, user identity ID, user public key upk, message m and signature σ, it returns 1 or 0 for accept or reject, respectively.

Both of them may be randomized but the second one is usually not, but for generality, we do not mandate that the verification algorithm must be deterministic. Boneh and Franklin [8] described a generic method for converting any ID-based encryption scheme into a standard signature scheme. The transformed signature scheme has a randomized verification algorithm. When this standard signature scheme is used in the construction of a CL-SIG scheme, for example, in the Hu-Wong-Zhang-Deng generic CL-SIG construction, the CL-SIG verification algorithm will also be randomized.

Signature Correctness. For all $k \in \mathbb{N}$, $m \in \{0,1\}^*$, $ID \in \{0,1\}^*$, we require that if $(mpk, msk) \leftarrow \mathsf{MasterKeyGen}(1^k)$, $partial\_key \leftarrow \mathsf{PartialKeyGen}(msk, ID)$, $(upk, usk) \leftarrow \mathsf{UserKeyGen}(mpk, ID)$, and $\sigma \leftarrow \mathsf{CL\text{-}Sign}(usk, partial\_key, m)$, then $\mathsf{CL\text{-}Ver}(mpk, ID, upk, m, \sigma) = 1$.

A certificateless encryption (CL-ENC) scheme has two polynomial-time algorithms in addition to the three key generation algorithms: CL-Encrypt and CL-Decrypt. Similar to the case of signature schemes, both of these algorithms may be randomized but usually the second one is not.

1. CL-Encrypt: On input mpk, user identity ID, user public key upk, message m, it returns a ciphertext c.
2. CL-Decrypt: On input user secret key usk, user partial key $partial\_key$ and ciphertext c, it returns a message m.

Cipher Correctness. For all $k \in \mathbb{N}$, $m \in \{0,1\}^*$, $ID \in \{0,1\}^*$, if $(mpk, msk) \leftarrow \mathsf{MasterKeyGen}(1^k)$, $partial\_key \leftarrow \mathsf{PartialKeyGen}(msk, ID)$, $(upk, usk) \leftarrow \mathsf{UserKeyGen}(mpk, ID)$, then we require that $m \leftarrow \mathsf{CL\text{-}Decrypt}(usk, partial\_key, \mathsf{CL\text{-}Encrypt}(mpk, ID, upk, m))$.

Case of invalid input: For each of the algorithms above, we implicitly assume that there is a domain for each of its inputs if it is not specified. An input is said to be *valid* if it falls in its corresponding domain. For example, the domain of msk is defined by the set of all possible output values of master secret key of MasterKeyGen for each given security parameter $k \in \mathbb{N}$. Hence if any of the inputs of an algorithm above is invalid, then the algorithm will output a symbol $\perp$ indicating that the execution of the algorithm is halted with failure.

In practice, the KGC (Key Generation Center) could be the one who performs MasterKeyGen and PartialKeyGen. The master public key mpk will then be published and assumed that everyone in the system has got a legitimate copy of it. The partial key is also assumed to be issued securely to the intended user so that no one except the intended user can get the partial key. For each user in the system, the user is supposed to be able to carry out UserKeyGen, and also CL-Sign and CL-Ver for CL-SIG or CL-Encrypt and CL-Decrypt for CL-ENC. It is the user's responsibility to forward the user public key upk to the intended signature verifier(s)/encryptor(s) and announces the user's identity.

In the rest of the paper, we denote the user partial key of a user with identity ID as $partial\_key_{ID}$ and the user public/secret key pair as (upk_{ID}, usk_{ID}).

2.1 Adversarial Model for Certificateless Signature (CL-SIG)

There are two types of adversaries, $\mathcal{A}_I$ and $\mathcal{A}_{II}$. Adversary $\mathcal{A}_I$ simulates attacks when the adversary (or signature forger) compromises the user secret key usk or replaces the user public key upk. However, $\mathcal{A}_I$ is not given the master secret key msk nor getting access to the user partial key $partial\_key$. Adversary $\mathcal{A}_{II}$ simulates attacks when the adversary controls msk and $partial\_key$. But $\mathcal{A}_{II}$ cannot get access to usk nor replace upk. Different from [14] and all other old models [1,20,15], in our model, $\mathcal{A}_{II}$ controls the key generation of the KGC while in all the previous models, the adversary is not allowed to do so. Informally, $\mathcal{A}_I$ models a third party launching key replacement attack and $\mathcal{A}_{II}$ models attacks launched by a malicious-but-passive KGC.

There are five oracles which can be accessed by the adversaries according to the game specifications which will be given shortly. For simulating singing oracle, we assume that the game simulator/challenger keeps a history of "query-answer" while interacting with adversaries. The five oracles are:

1. CreateUser: On input an identity $ID \in \{0,1\}^*$, if ID has already been created, nothing is to be carried out. Otherwise, the oracle generates $partial\_key_{ID} \leftarrow \mathsf{PartialKeyGen}(msk, ID)$ and $(upk_{ID}, usk_{ID}) \leftarrow \mathsf{UserKeyGen}(mpk, ID)$. In this case, ID is said to be *created*. In both cases, upk_{ID} is returned.
2. RevealPartialKey: On input an identity ID, it returns $partial\_key_{ID}$ if ID has been created. Otherwise, a symbol $\perp$ is returned.
3. RevealSecretKey: On input an identity ID, it returns the corresponding user secret key usk_{ID} if ID has been created. Otherwise, a symbol $\perp$ is returned.
4. ReplaceKey: On input an identity ID and a user public/secret key pair (upk^*, usk^*), the original user public/secret key pair of ID is replaced with (upk^*, usk^*) if ID has been created. Otherwise, no action will be taken.
5. Sign: On input an identity ID and a message $m \in \{0,1\}^*$, the signing oracle proceeds in one of the three cases below.
   (a) A *valid* signature σ is returned if ID has been created but the pair (upk_{ID}, usk_{ID}) has not been replaced. The signature σ is *valid* if $\mathsf{CL\text{-}Ver}(mpk, ID, upk_{ID}, m, \sigma) = \mathsf{accept}$.
   (b) If ID has not been created, a symbol $\perp$ is returned.
   (c) If the pair (upk_{ID}, usk_{ID}) has been replaced with, say (upk^*, usk^*), then the oracle returns $\sigma \leftarrow \mathsf{CL\text{-}Sign}(usk^*, partial\_key_{ID}, m)$ if σ is valid. If σ is not valid, the oracle searches over the "query-answer" list for a user secret key usk' such that $\sigma' \leftarrow \mathsf{CL\text{-}Sign}(usk', partial\_key_{ID}, m)$ is valid, and returns σ'. However, if such a usk' cannot be found, the oracle runs a special "knowledge extractor" to obtain a valid signature and returns it to the adversary. Note that the construction of knowledge extractor is specific to each CL-SIG scheme.

Remark 1: When querying the oracle $\mathsf{ReplaceKey}$, usk^* can be an empty string. In this case, it means that the user secret key is not provided. If usk^* is an empty string and the original user secret key of an identity ID is replaced with usk^*, then the empty string will be returned if the $\mathsf{RevealSecretKey}$ oracle is queried on ID. Also note that even if usk^* is not an empty string, it does not mean that usk^* is the corresponding secret key of upk^*. Hence as mentioned, the signature generated by the signing oracle Sign using usk^* for case (c) may not be valid.

Remark 2: The definition of signing oracle above follows that original idea of the model for CL-ENC in [1], that is, an adversary is expected to obtain valid decryption from a decryption oracle, even after the corresponding user public key has been replaced. This means that the simulator of the model/game should be able to correctly answer decryption queries for public keys where the corresponding secret keys may not be known to the simulator. To do this, a scheme-specific knowledge extractor [1] will be used by the game simulator to decrypt a requested ciphertext if the corresponding decryption key cannot be found in the list of "query-answer". In our signing oracle, case (c) above, we adopt this idea.

Remark 3: We should also note that case (c) of the signing oracle is a very strong security requirement. In addition, it is unclear how realistic this requirement is. As later adopted and argued by many other researchers [20,14,22,6,9,10], the game simulator is not required to provide valid signatures or correct decryption of ciphertexts (for CL-ENC schemes) after the corresponding user public key has been replaced. Instead, they only require that valid signatures are generated or ciphertexts can be decrypted if the user public key has been replaced while the corresponding user secret key has also been supplied by the adversary. We recommend that if this strong security requirement is not needed, one may use the signing oracle defined in [14] to replace the definition above.

We define two games, one for $\mathcal{A}_I$ and the other one for $\mathcal{A}_{II}$. Each of the games will capture the notion of existential unforgeability against chosen message attack in the sense of [13]. The game for $\mathcal{A}_I$ below is the same as the corresponding game defined in [14], except that the signing oracle is specified differently.

Game Sign-I: Let $\mathcal{S}_I$ be the game simulator/challenger and $k \in \mathbb{N}$ be a security parameter.
1. $\mathcal{S}_I$ executes $\mathsf{MasterKeyGen}(1^k)$ to get (mpk, msk).
2. $\mathcal{S}_I$ runs $\mathcal{A}_I$ on 1^k and mpk. During the simulation, $\mathcal{A}_I$ can make queries onto $\mathsf{CreateUser}$, $\mathsf{RevealPartialKey}$, $\mathsf{RevealSecretKey}$, $\mathsf{ReplaceKey}$ and Sign.
3. $\mathcal{A}_I$ is to output (ID^*, m^*, σ^*).

$\mathcal{A}_I$ *wins* if $\mathsf{CL\text{-}Ver}(mpk, ID^*, upk_{ID^*}, m^*, \sigma^*) = \mathsf{accept}$ for some created ID^* and the oracle Sign has never been queried with (ID^*, m^*). One **additional restriction** is that $\mathcal{A}_I$ has never queried $\mathsf{RevealPartialKey}(ID^*)$ to get the user partial key $partial\_key_{ID^*}$.

A CL-SIG scheme is secure in **Game Sign-I** if for all probabilistic polynomial-time (PPT) algorithm $\mathcal{A}_I$, it is negligible for $\mathcal{A}_I$ to win the game. Note that $\mathcal{A}_I$ may have queried $\mathsf{RevealSecretKey}(ID^*)$ and got the user secret key usk_{ID^*} or queried $\mathsf{ReplaceKey}(ID^*, \cdot, \cdot)$ and replaced the user public key upk_{ID^*} before generating a forgery (ID^*, m^*, σ^*).

The following game is for $\mathcal{A}_{II}$. It captures malicious-but-passive KGC attacks.

Game Sign-II: Let $\mathcal{S}_{II}$ be the game simulator and $k \in \mathbb{N}$ be a security parameter. There are two phases of interactions between $\mathcal{S}_{II}$ and adversary $\mathcal{A}_{II}$.

Phase 1. $\mathcal{S}_{II}$ executes $\mathcal{A}_{II}$ on 1^k and a special tag $\mathsf{master\text{-}key\text{-}gen}$. $\mathcal{A}_{II}$ returns a master public key mpk. Note that $\mathcal{A}_{II}$ is not allowed to query any oracle in this phase[5].

Phase 2. This phase starts when $\mathcal{S}_{II}$ invokes $\mathcal{A}_{II}$ again with 1^k but with another tag forge. During the simulation, $\mathcal{A}_{II}$ can make queries onto the oracles $\mathsf{RevealSecretKey}$, $\mathsf{ReplaceKey}$ and Sign as described above. $\mathcal{A}_{II}$ can also make queries to $\mathsf{CreateUser}$. However, the oracle is changed as follows.

[5] If the security analysis is done under the random oracle model [5], then the adversary is always allowed to access the specified random oracles of the underlying scheme at any stage of the game.

CreateUser: On input an identity $ID \in \{0,1\}^*$ and a user partial key $partial\_key_{ID}$, if ID has already been created, nothing is to be carried out. Otherwise, the oracle generates $(upk_{ID}, usk_{ID}) \leftarrow \mathsf{UserKeyGen}(mpk, ID)$. In this case, ID is said to be *created*. Also, $\mathcal{S}_{II}$ associates $partial\_key_{ID}$ and (upk_{ID}, usk_{ID}) to ID. In both cases, upk_{ID} is returned.

Note that oracle RevealPartialKey is not accessible. This oracle is no longer needed because all the user partial keys are now generated by $\mathcal{A}_{II}$.

At the end of this phase, $\mathcal{A}_{II}$ is to output a triple (ID^*, m^*, σ^*).

$\mathcal{A}_{II}$ *wins* if $\mathsf{CL\text{-}Ver}(mpk, ID^*, upk_{ID^*}, m^*, \sigma^*) = \mathsf{accept}$ for some created ID^* and the oracle Sign has never been queried with (ID^*, m^*). One additional restriction is that $\mathcal{A}_{II}$ has never queried $\mathsf{RevealSecretKey}(ID^*)$ to get the user secret key usk_{ID^*} nor queried $\mathsf{ReplaceKey}(ID^*, \cdot, \cdot)$ to replace the user public key upk_{ID^*}.

A CL-SIG scheme is secure in **Game Sign-II** if for all PPT algorithm $\mathcal{A}_{II}$, it is negligible for $\mathcal{A}_{II}$ to win the game. Note that $\mathcal{A}_{II}$ in the game above can not only generate master key pair maliciously, but also generate user partial key maliciously.

2.2 Adversarial Model for Certificateless Encryption (CL-ENC)

Similar to the adversarial model for CL-SIG above, there are two types of adversaries, $\mathcal{B}_I$ and $\mathcal{B}_{II}$. Adversary $\mathcal{B}_I$ models a third party launching key replacement attack and $\mathcal{B}_{II}$ models attacks launched by a malicious-but-passive KGC.

There are five oracles which can be accessed by the adversaries. They are CreateUser, RevealPartialKey, RevealSecretKey, ReplaceKey and Decrypt. For simulating the decryption oracle Decrypt, we assume that the game simulator keeps a history of "query-answer" while interacting with adversaries. The first four oracles are defined in the same way as that in Sec. 2.1. Below is the definition of oracle Decrypt.

Decrypt: On input an identity ID and a ciphertext c, the decryption oracle proceeds in one of the three cases below.

1. A message m is returned if ID has been created while the pair (upk_{ID}, usk_{ID}) has not been replaced. In this case, $m \leftarrow \mathsf{CL\text{-}Decrypt}(usk_{ID}, partial\_key_{ID}, c)$.
2. If ID has not been created, a symbol $\perp$ is returned.
3. If the user public/secret key pair of ID has been replaced with, say (upk^*, usk^*), then the oracle returns $m \leftarrow \mathsf{CL\text{-}Decrypt}(usk^*, partial\_key_{ID}, c)$ if $m \neq \perp$. However, if $m = \perp$, the oracle searches over the "query-answer" list for a user secret key usk' such that $m' \neq \perp$ where $m' \leftarrow \mathsf{CL\text{-}Decrypt}(usk', partial\_key_{ID}, c)$, and returns m'. If such a usk' cannot be found, the oracle runs a special "knowledge extractor" to decrypt the ciphertext c and returns the message to the adversary. Note that the construction of knowledge extractor is specific to each CL-ENC scheme.

Again, as a remark similar to Remark 3 on page 6, if we do not require the simulator to correctly answer a query to Decrypt oracle when the user secret key is not known, we should modify item 3 of Decrypt oracle above accordingly.

Game Enc-I: Let $\mathcal{C}_I$ be the game challenger/simulator and $k \in \mathbb{N}$ be a security parameter.

1. $\mathcal{C}_I$ executes $\mathsf{MasterKeyGen}(1^k)$ to get (mpk, msk).
2. $\mathcal{C}_I$ runs adversary $\mathcal{B}_I$ on 1^k and mpk. During the simulation, $\mathcal{B}_I$ can make queries onto CreateUser, RevealPartialKey, RevealSecretKey, ReplaceKey and Decrypt. At the end of this phase, $\mathcal{B}_I$ outputs two equal-length messages (M_0, M_1) and a target identity ID^*.
3. $\mathcal{C}_I$ picks a bit $b \in \{0,1\}$ at random and gives a challenge ciphertext c^* to $\mathcal{B}_I$ where $c^* \leftarrow \mathsf{CL\text{-}Encrypt}(mpk, ID^*, upk_{ID^*}, M_b)$.
4. $\mathcal{B}_I$ makes queries as in step 2. At the end of the game, $\mathcal{B}_I$ outputs its guess $b' \in \{0,1\}$.

$\mathcal{B}_I$ *wins* if $b' = b$. The restrictions are:
- $\mathcal{B}_I$ has never queried $\mathsf{RevealPartialKey}(ID^*)$ to get the user partial key $partial\_key_{ID^*}$; and
- $\mathcal{B}_I$ has never queried $\mathsf{Decrypt}$ on the pair (ID^*, c^*).

A CL-ENC scheme is secure in **Game Enc-I** if for all probabilistic polynomial-time (PPT) algorithm $\mathcal{B}_I$, it is negligible for $\mathcal{B}_I$ to win the game.

Note that in **Game Enc-I** above, $\mathcal{B}_I$ may have queried $\mathsf{RevealSecretKey}(ID^*)$ *before* any key replacement queries made on ID^*. This adversarial capability has not been captured in any previous Type II adversarial models for CL-ENC. In previous models, although the adversary corresponding to $\mathcal{B}_I$ is able to replace user public/secret key pairs, the adversary is not able to retrieve the 'original' user secret key generated by the honest user with identity ID^*.

Game Enc-II: Let $\mathcal{C}_{II}$ be the game simulator and $k \in \mathbb{N}$ be a security parameter.
1. $\mathcal{C}_{II}$ runs adversary $\mathcal{B}_{II}$ on 1^k and a special tag $\mathsf{master\text{-}key\text{-}gen}$. $\mathcal{B}_{II}$ returns a master public key mpk. In this phase, $\mathcal{B}_{II}$ is not allowed to query any oracle with one exception specified in footnote 5 on page 6.
2. $\mathcal{C}_{II}$ invokes $\mathcal{B}_{II}$ again with 1^k but with another tag choose. During the simulation, $\mathcal{B}_{II}$ can make queries onto the oracles $\mathsf{RevealSecretKey}$, $\mathsf{ReplaceKey}$ and $\mathsf{Decrypt}$ as described above. $\mathcal{B}_{II}$ can also make queries to $\mathsf{CreateUser}$ as described in **Game Sign-II**. As in **Game Sign-II**, oracle $\mathsf{RevealPartialKey}$ is not provided to the adversary as all the user partial keys are now generated by the adversary. At the end of this phase, $\mathcal{B}_{II}$ outputs two equal-length messages (M_0, M_1) and a target identity ID^*.
3. $\mathcal{C}_{II}$ picks a bit $b \in \{0, 1\}$ at random and runs $\mathcal{B}_{II}$ on input a challenge ciphertext c^* and a tag guess where $c^* \leftarrow \mathsf{CL\text{-}Encrypt}(mpk, ID^*, upk_{ID^*}, M_b)$.
4. $\mathcal{B}_{II}$ makes queries as in step 2. At the end of the game, $\mathcal{B}_{II}$ outputs its guess $b' \in \{0, 1\}$.

$\mathcal{B}_{II}$ *wins* if $b' = b$. The restrictions are:
- $\mathcal{B}_{II}$ has never queried $\mathsf{RevealSecretKey}(ID^*)$ to get the user secret key usk_{ID^*};
- $\mathcal{B}_{II}$ has never queried $\mathsf{ReplaceKey}(ID^*, \cdot, \cdot)$ to replace the user public key upk_{ID^*}; and
- $\mathcal{B}_{II}$ has never queried $\mathsf{Decrypt}$ on the pair (ID^*, c^*).

A CL-ENC scheme is secure in **Game Enc-II** if for all PPT algorithm $\mathcal{B}_{II}$, it is negligible for $\mathcal{B}_{II}$ to win the game.

In previous models corresponding to **Game Enc-II**, the adversary cannot replace the user public key of any user in the system. In **Game Enc-II** above, we relax this restriction and allow the adversary to access $\mathsf{ReplaceKey}$ as long as the user public key corresponding to ID^* has never been replaced.

3 Malicious-but-passive KGC Attack

In [1], Al-Riyami and Paterson proposed a CL-ENC scheme and also a CL-SIG scheme. These schemes share the same set of key generation algorithms, which we will show to be vulnerable to malicious-but-passive KGC attack. In the following, we first review these schemes using the definitions in Sec. 2, then we describe how the malicious-but-passive KGC attack works.

1. $\mathsf{MasterKeyGen}$: On input 1^k, choose a bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ of prime order q with pairing operation $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where q is of k bits long. Choose a random generator g of $\mathbb{G}_1$. Set the master secret key msk as $s \in_R \mathbb{Z}_q^*$. Compute $W = g^s$, and choose two hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^n$, where n is the bit-length of message. The master public key mpk is $(\mathbb{G}_1, \mathbb{G}_2, e, g, W, H_1, H_2)$. For the signature scheme, an additional hash function $H_3 : \{0, 1\}^* \times \mathbb{G}_2 \rightarrow \mathbb{Z}_q^*$ is needed.
2. $\mathsf{PartialKeyGen}$: On input s and user identity $ID \in \{0, 1\}^*$, compute $Q_{ID} = H_1(ID)$ and output user partial key $partial\_key_{ID}$ as $D_{ID} = Q_{ID}^s$.
3. $\mathsf{UserKeyGen}$: On input mpk and user identity ID, randomly select $x \in_R \mathbb{Z}_q^*$, compute user secret key $usk_{ID} = D_{ID}^x$, and set user public key upk_{ID} as $(X_{ID} = g^x, Y_{ID} = W^x)$.

In [1], a basic encryption system is first described, followed by a final system which possesses security against chosen-ciphertext attack (CCA). The final system is obtained by transforming the basic one using the Fujisaki-Okamoto conversion [11]. In the following, we only review their basic system since the malicious-but-passive KGC attack to be described allows the KGC to compromise all the secret information of a user, and therefore the attack can be applied directly to the final system.

- $\mathsf{CL\text{-}Encrypt}$: On input mpk, user identity ID, user public key (X_{ID}, Y_{ID}), message $m \in \{0,1\}^n$,
  1. check that $X_{ID}, Y_{ID} \in \mathbb{G}_1$ and the equality $e(X_{ID}, W) = e(Y_{ID}, g)$ holds, abort otherwise;
  2. compute $Q_{ID} = H_1(ID)$ and randomly choose $r \in_R \mathbb{Z}_q^*$; and
  3. output the ciphertext $c = \langle g^r, m \oplus H_2(e(Q_{ID}, Y_{ID})^r)\rangle$.
- $\mathsf{CL\text{-}Decrypt}$: On input user secret key $usk_{ID} = D_{ID}^x$, user partial key $partial\_key_{ID} = D_{ID}$ and ciphertext $c = \langle U, V\rangle$, return a message $m = V \oplus H_2(e(usk_{ID}, U))$.

- $\mathsf{CL\text{-}Sign}$: On input user (with identity ID) secret key $usk_{ID} = D_{ID}^x$, user partial key D_{ID}, and message m,
  1. choose a random $a \in_R \mathbb{Z}_q^*$, compute $r = e(g, g)^a$;
  2. compute $v = H_3(m, r)$ and $U = (usk_{ID})^v g^a$; and
  3. output signature $\sigma = \langle U, v\rangle$.
- $\mathsf{CL\text{-}Ver}$: On input mpk, identity ID, user public key (X_{ID}, Y_{ID}), message m and signature σ,
  1. Check that $X_{ID}, Y_{ID} \in \mathbb{G}_1$ and the equality $e(X_{ID}, W) = e(Y_{ID}, g)$ hold, abort otherwise;
  2. compute $r = e(U, g)e(Q_{ID}, Y_{ID})^{-v}$; and
  3. check if $v = H_3(m, r)$ holds, output 1 for accept if yes and output 0 for reject otherwise.

Malicious-but-passive KGC Attack. We now show how a malicious-but-passive KGC can obtain the user secret key of the victim user of the KGC's choice. Also note that this attack is not captured in the original security model in [1], but only in the models proposed in Sec. 2.

In order to obtain the user secret key of an arbitrarily chosen identity ID^* (not necessarily to be random), the KGC randomly chooses $\alpha \in_R \mathbb{Z}_q$ and computes $g = H(ID^*)^\alpha$. The rest follows the original $\mathsf{MasterKeyGen}$ and $\mathsf{PartialKeyGen}$.

Suppose the user public key published by the user with identity ID^* is $(X_{ID*} = g^{x^*}, Y_{ID*} = g^{sx^*})$, where $x^* \in_R \mathbb{Z}_q$. From the user public key, the KGC can compute the user secret key by $usk_{ID^*} = Y_{ID*}^{\alpha^{-1}}$. Since the user partial key is generated by the KGC, after obtaining the user secret key of the victim, the KGC can then decrypt all the ciphertexts for or generate any signature on behalf of the victim.

Several other certificateless cryptosystems [15,16], employing the same key structure as [1], are also vulnerable to this attack.

4 Hu-Wong-Zhang-Deng Generic Certificateless Signature Scheme

In [14], Hu, Wong, Zhang and Deng proposed a generic CL-SIG construction (HWZD-CL-SIG scheme for short), which is based on a standard signature scheme denoted by $\Pi^{SS} = (\mathsf{Setup}^{SS}, Sign^{SS}, Ver^{SS})$ and an ID-based signature scheme denoted by $\Pi^{IBS} = (\mathsf{Setup}^{IBS}, \mathsf{Extract}^{IBS}, Sign^{IBS}, Ver^{IBS})$. The HWZD-CL-SIG scheme is proven secure in the Type I and Type II games defined in [14] provided that Π^{SS} is existentially unforgeable against chosen message attack ($\mathsf{euf\text{-}cma}$) [13] and Π^{IBS} is existentially unforgeable against chosen message and identity attack ($\mathsf{euf\text{-}cma\text{-}ida}$) [4]. For detailed definitions of Π^{SS} and Π^{IBS}, please refer to Appendix A.

If we replace the signing oracle defined in Sec. 2.1 with the signing oracle defined in [14], we can see that our **Game Sign-I** (that is the Type I game) is identical to that in [14]. Hence HWZD-CL-SIG scheme is secure in **Game Sign-I** provided that we use the signing oracle of [14]. Please refer to Sec. 2.1 for more details on the difference between the signing oracle defined in Sec. 2.1 and the signing oracle of [14]. On Type II game, different from that of [14], we also add the ingredient of malicious-but-passive KGC attack in **Game Sign-II**. In the following, we review the HWZD-CL-SIG scheme and show that it is secure in **Game Sign-II** provided that the signing oracle of [14] is used.

```
$(mpk, msk) \leftarrow \textsf{\textbf{MasterKeyGen}}(1^k)$
    Run $(mpk^{IBS}, msk^{IBS}) \leftarrow \textsf{Setup}^{IBS}(1^k)$; and set $mpk := mpk^{IBS}$ and $msk := msk^{IBS}$.
$partial\_key_{ID} \leftarrow \textsf{\textbf{PartialKeyGen}}(msk, ID)$
    Run $usk^{IBS} \leftarrow \textsf{Extract}^{IBS}(msk, ID)$; and set $partial\_key_{ID} := \langle usk^{IBS} \| ID \| mpk \rangle$.
$(upk_{ID}, usk_{ID}) \leftarrow \textsf{\textbf{UserKeyGen}}(mpk, ID)$
    Run $(upk^{SS}, usk^{SS}) \leftarrow \textsf{Setup}^{SS}(1^k)$; and set $upk_{ID} := upk^{SS}$ and $usk_{ID} := \langle usk^{SS} \parallel upk^{SS} \rangle$.
$\sigma \leftarrow \textsf{\textbf{CL-Sign}}(usk_{ID}, partial\_key_{ID}, m)$
    Run $\sigma^{SS} \leftarrow Sign^{SS}(usk^{SS},\ m \| mpk \| ID \| upk^{SS})$; then
    run $\sigma^{IBS} \leftarrow Sign^{IBS}(usk^{IBS},\ m \| mpk \| ID \| upk^{SS} \| \sigma^{SS})$; and
    set $\sigma := \langle\ \sigma^{SS} \parallel \sigma^{IBS}\ \rangle$.
$1/0 \leftarrow \textsf{\textbf{CL-Ver}}(mpk, ID, upk_{ID}, m, \sigma)$
    Parse $\sigma$ into $\langle\ \sigma^{SS} \parallel \sigma^{IBS}\ \rangle$;
    run $b_1 \leftarrow Ver^{IBS}(mpk, ID,\ m \| mpk \| ID \| upk^{SS} \| \sigma^{SS}, \sigma^{IBS})$;
    run $b_2 \leftarrow Ver^{SS}(upk^{SS},\ m \| mpk \| ID \| upk^{SS}, \sigma^{SS})$; and
    set output to $b_1 \wedge b_2$.
```

Theorem 1. *The HWZD-CL-SIG scheme is secure in **Game Sign-II** if the standard signature scheme Π^{SS} is* euf-cma *secure and the signing oracle in **Game Sign-II** is replaced by the one defined in [14].*

Proof. We construct a PPT algorithm/forger $\mathcal{S}_{II}$ which breaks the euf-cma security of the signature scheme Π^{SS} by running $\mathcal{A}_{II}$ and answering $\mathcal{A}_{II}$'s queries as defined in **Game Sign-II**. Consider a game for euf-cma [13] simulated by a simulator $\mathcal{S}'$. $\mathcal{S}'$ gives a challenge public key upk^* to $\mathcal{S}_{II}$ and simulates a signing oracle with respect to upk^*. $\mathcal{S}_{II}$ is to forge a message-signature pair such that the signature is valid with respect to upk^*.

At the beginning of **Game Sign-II**, $\mathcal{A}_{II}$ is executed and a master public key mpk is returned. Note that $\mathcal{A}_{II}$ may not run MasterKeyGen to get mpk. Below are the oracle simulations.

1. CreateUser: On input an identity ID and a user partial key $partial\_key_{ID}$, $\mathcal{S}_{II}$ executes $\textsf{Setup}^{SS}$ of Π^{SS} to generate (upk^{SS}, usk^{SS}). $\mathcal{S}_{II}$ returns upk^{SS} as user public key upk_{ID}. $\mathcal{S}_{II}$ also maintains the restriction that each identity ID can only be created once. The above is carried out every time when CreateUser is queried except one:

   Suppose $\mathcal{A}_{II}$ *creates* at most q_c distinct identities. Among all the distinct identities, $\mathcal{S}_{II}$ randomly picks one, say the i-th query with identity ID^*, and answers the query with upk^*, that is, $\mathcal{S}_{II}$ sets $upk_{ID^*} := upk^*$.
2. RevealSecretKey: If ID is not created, $\perp$ is returned. Otherwise, if $ID \neq ID^*$, $\mathcal{S}_{II}$ returns the corresponding user secret key; if $ID = ID^*$, $\mathcal{S}_{II}$ halts with failure.
3. ReplaceKey: If ID is not created, no action will be taken. Otherwise, if $ID \neq ID^*$, $\mathcal{S}_{II}$ replaces its copy of user public/secret key pair with the query inputs denoted by (upk^*, usk^*); if $ID = ID^*$, $\mathcal{S}_{II}$ halts with failure.
4. Sign: If $ID \neq ID^*$, $\mathcal{S}_{II}$ executes CL-Sign according to the scheme specification. If $ID = ID^*$, $\mathcal{S}_{II}$ sets $m' := \langle\ m \parallel mpk \parallel ID^* \parallel upk_{ID^*}\ \rangle$, queries the signing oracle of Π^{SS} with m' to get σ^{SS}, and generates σ^{IBS} using $Sign^{IBS}$. Finally, $\langle\ \sigma^{SS} \parallel \sigma^{IBS}\ \rangle$ is returned.

When $\mathcal{A}_{II}$ outputs a triple $(\hat{ID}, \hat{m}, \hat{\sigma})$ where $\hat{\sigma} = \langle\ \hat{\sigma}^{SS} \parallel \hat{\sigma}^{IBS}\ \rangle$, $\mathcal{S}_{II}$ outputs $(\hat{m}', \hat{\sigma}^{SS})$, where $\hat{m}' := \langle\ \hat{m} \parallel mpk \parallel \hat{ID} \parallel upk_{\hat{ID}}\ \rangle$. When $\mathcal{A}_{II}$ halts, $\mathcal{S}_{II}$ halts.

For the event that $\mathcal{S}_{II}$ does not fail, we can see that the simulation is correct as all the operations involving Π^{IBS} are carried out according to the scheme specification and the game specification, and all the operations involving Π^{SS} are carried out accordingly as well provided that $ID \neq ID^*$. If ID^* is involved, the Sign query can still be performed correctly with the help of the signing oracle simulated by $\mathcal{S}'$. In addition, the running time of $\mathcal{S}_{II}$ is in polynomial of that of $\mathcal{A}_{II}$. If $\mathcal{S}_{II}$ does not fail, all queries simulated by $\mathcal{S}_{II}$ will be indistinguishable from that of a real game.

If $\mathcal{A}_{II}$ wins the game and $\hat{ID} = ID^*$, this implies that oracle Sign has never been queried with $(\hat{ID}, \hat{m})$. In addition, the corresponding user secret key of ID^* is neither revealed via RevealSecretKey nor replaced via $\mathsf{ReplaceKey}$. Since ID^* is randomly chosen among at most q_c identities, the probability that $\hat{ID} = ID^*$ is at least $1/q_c$. Since the only case that the signing oracle simulated by $\mathcal{S}'$ may be queried by $\mathcal{S}_{II}$ is when simulating the Sign oracle. Without querying Sign with $(\hat{ID}, \hat{m})$, it is impossible for $\mathcal{S}_{II}$ to have queried the signing oracle simulated by $\mathcal{S}'$ with m' where m' is an encoding of the form $\langle\ \hat{m} \parallel mpk \parallel \hat{ID} \parallel \cdots \rangle$. Therefore, $(\hat{m}', \hat{\sigma}^{SS})$ must be a valid forgery with respect to the signature scheme Π^{SS} under the public key upk^*. □

5 Libert-Quisquater Generic Certificateless Encryption Scheme

Libert and Quisquater [17] proposed a generic CL-ENC construction that achieves security against chosen ciphertext attack (CCA). The construction is based on conventional Public Key Encryption (PKE) and ID-Based Encryption (IBE). In the following, we use $\Pi^{PKE} = (\mathcal{K}^{PKE}, \mathcal{E}^{PKE}, \mathcal{D}^{PKE})$ and $\Pi^{IBE} = (\mathsf{Setup}^{IBE}, \mathsf{Extract}^{IBE}, \mathcal{E}^{IBE}, \mathcal{D}^{IBE})$ to denote a PKE scheme and an IBE scheme, respectively. For their full definitions, please refer to Appendix A. We now review the Libert-Quisquater generic CL-ENC scheme.

- $\mathsf{MasterKeyGen}$: Run $(mpk^{IBE}, msk^{IBE}) \leftarrow \mathsf{Setup}^{IBE}(1^k)$ and set master public/secret key pair $(mpk, msk) := (mpk^{IBE}, msk^{IBE})$.
- $\mathsf{PartialKeyGen}$: Run $usk^{IBE} \leftarrow \mathsf{Extract}^{IBE}(msk^{IBE}, ID)$ and set $partial\_key_{ID} := usk^{IBE}$.
- $\mathsf{UserKeyGen}$: Run $(upk^{PKE}, usk^{PKE}) \leftarrow \mathcal{K}^{PKE}(1^k)$ and set $(upk_{ID}, usk_{ID}) := (upk^{PKE}, usk^{PKE})$. The message space is defined as the message space of Π^{PKE} with respect to upk^{PKE}. It is required that the ciphertext space of Π^{PKE} with respect to upk^{PKE} is a subset of the message space of Π^{IBE} with respect to mpk.
- $\mathsf{CL\text{-}Encrypt}$: To encrypt a message m under identity ID and upk_{ID}, the algorithm computes

$$C = \mathcal{E}^{IBE}\big(mpk, ID, \mathcal{E}^{PKE}(upk_{ID}, m)\big)$$

- $\mathsf{CL\text{-}Decrypt}$: To decrypt a ciphertext C, the algorithm computes
  1. $\tilde{m} \leftarrow \mathcal{D}^{IBE}(partial\_key_{ID}, ID, C)$. If $\tilde{m} = \perp$, output $\perp$ and halt.
  2. Otherwise, compute $m' \leftarrow \mathcal{D}^{PKE}(usk_{ID}, \tilde{m})$ and output m'.

This scheme has been shown to be semantically secure (i.e. CPA secure) [17] provided that the underlying Π^{PKE} and Π^{IBE} are CPA secure. The following theorem shows that it is also CPA secure in our new models.

Theorem 2. *The scheme described above is secure in* ***Game Enc-I*** *and* ***Game Enc-II*** *as defined in Sec. 2 provided that the* Decrypt *oracle cannot be accessed.*

Proof. Without access to the decryption oracle $\mathsf{Decrypt}$, the games **Game Enc-I** and **Game Enc-II** only capture the CPA (chosen plaintext attack) security.

We first show how an attacker $\mathcal{C}_I$ uses adversary $\mathcal{B}_I$ (i.e. Type I adversary) to break the CPA security of Π^{IBE}[6]. $\mathcal{C}_I$ obtains ID-based master public key mpk^{IBE} from its challenger/simulator $\mathcal{S}^{IBE}$. It forwards mpk^{IBE} to $\mathcal{B}_I$ as mpk. In $\mathcal{C}_I$'s interaction with $\mathcal{B}_I$, we denote ID_i as the i^{th} distinct identity among the queries made by $\mathcal{B}_I$. Let q_{ID} be the total number of distinct identities involved among all the queries. $\mathcal{C}_I$ randomly chooses an index $\ell \in_R \{1, \ldots, q_{ID}\}$. $\mathcal{C}_I$ also simulates the following oracles:

- $\mathsf{CreateUser}$: on input ID_i (assumed to be distinct), $\mathcal{C}_I$ runs $\mathcal{K}^{PKE}$ to generate user public/secret key pair $(upk_i^{PKE}, usk_i^{PKE})$. If $i \neq \ell$, $\mathcal{C}_I$ also queries $\mathcal{S}^{IBE}$ for ID_i's user partial key usk_i^{IBE}, otherwise, usk_ℓ^{IBE} is set to $\perp$. $\mathcal{C}_I$ stores $(ID_i, usk_i^{PKE}, upk_i^{PKE}, usk_i^{IBE})$ in its database and returns upk_i^{PKE}.

[6] We omit the review of the security model of IBE schemes and refer readers to [7] for details.

– $\mathsf{RevealPartialKey}$: on input ID_i, if $i = \ell$, $\mathcal{C}_I$ aborts. Otherwise it looks up its database for ID_i's user partial key usk_i^{IBE} and forwards it to $\mathcal{B}_I$ if there exists. Otherwise, $\perp$ is returned.
– $\mathsf{RevealSecretKey}$: on input ID_i, $\mathcal{C}_I$ returns usk_i^{PKE} if there exists. Otherwise, $\perp$ is returned.
– $\mathsf{ReplaceKey}$: on input ID_i and (upk^*, usk^*), $\mathcal{C}_I$ replaces the user public key of ID_i as upk^* and the user secret key as usk^* if ID_i has been created. Otherwise no action will be taken. Note that usk^* could be an empty string.

At the challenge step, $\mathcal{B}_I$ outputs two equal-length messages (m_0, m_1) and a target identity ID^*. $\mathcal{C}_I$ aborts if $ID^* \neq ID_\ell$. Otherwise, it encrypts both m_0 and m_1 into $c_0 = \mathcal{E}^{PKE}(upk_\ell^{PKE}, m_0)$ and $c_1 = \mathcal{E}^{PKE}(upk_\ell^{PKE}, m_1)$ which are sent to $\mathcal{S}^{IBE}$ together with the target identity ID_ℓ as a challenge request. The challenge $c^* = \mathcal{E}^{IBE}(mpk, ID_\ell, c_b)$, $b \in_R \{0,1\}$, prepared by $\mathcal{S}^{IBE}$ is relayed to $\mathcal{B}_I$.

$\mathcal{B}_I$'s output $b' \in \{0,1\}$ is output by $\mathcal{C}_I$ as its guess for the hidden bit b of $\mathcal{S}^{IBE}$. If $\mathcal{B}_I$ is successful, $\mathcal{C}_I$ is also successful. The latter has a probability of at least $1/q_{ID}$ to successfully guess the identity on which $\mathcal{B}_I$ produces its attack.

Next, we show that the scheme is CPA secure against Type II adversary (in particular, against malicious-and-passive KGC). We describe how an attacker $\mathcal{C}_{II}$ uses adversary $\mathcal{B}_{II}$ to break the CPA security of Π^{PKE}. In the first step of **Game Enc-II**, $\mathcal{B}_{II}$ is executed and $\mathcal{B}_{II}$ returns a master public key mpk. Note that $\mathcal{B}_{II}$ may not execute Setup^{IBE} to generate mpk. $\mathcal{C}_{II}$ then randomly selects an index $\ell \in_R \{1, \ldots, q_{ID}\}$, where q_{ID} is the total number of distinct identities involved among all the queries, and obtains a challenge public key pk^* from its challenger/simulator $\mathcal{S}^{PKE}$. $\mathcal{C}_{II}$ also simulates the following oracles.

– $\mathsf{CreateUser}$: on input ID_i (assumed to be distinct) and user partial key $partial\_key_i$, if $i = \ell$, it sets $upk_i^{PKE} := upk^*$. Otherwise, it runs $\mathcal{K}^{PKE}$ to generate $(upk_i^{PKE}, usk_i^{PKE})$ and stores $(ID_i, partial\_key_i, upk_i^{PKE}, usk_i^{PKE})$ in its database. upk_i^{PKE} is returned.
– $\mathsf{RevealSecretKey}$: on input ID_i, if $i = \ell$, $\mathcal{C}_{II}$ aborts. Otherwise it looks up its database for ID_i's user secret key usk_i^{PKE} and returns it to $\mathcal{B}_{II}$ if there exists. Otherwise, $\perp$ is returned.
– $\mathsf{ReplaceKey}$: on input ID_i and (upk^*, usk^*), if $i = \ell$, $\mathcal{C}_{II}$ aborts. Otherwise, $\mathcal{C}_{II}$ replaces the user public key of ID_i as upk^* and the user secret key as usk^* if ID_i has been created. Otherwise no action will be taken.

At the challenge step, $\mathcal{B}_{II}$ outputs two messages (m_0, m_1) and a target identity ID^*. $\mathcal{C}_{II}$ aborts if $ID^* \neq ID_\ell$. Otherwise, it forwards (m_0, m_1) as a challenge query to $\mathcal{S}^{PKE}$ which responds with $c^* = \mathcal{E}^{PKE}(pk^*, m_b)$ for a random bit $b \in_R \{0,1\}$. This ciphertext is further encrypted into $C^* = \mathcal{E}^{IBE}(mpk, ID^*, c^*)$ and is given as a challenge to $\mathcal{B}_{II}$.

$\mathcal{B}_{II}$'s final result $b' \in \{0,1\}$ is output by $\mathcal{C}_{II}$ as a guess for the hidden bit of $\mathcal{S}^{PKE}$. If $\mathcal{B}_{II}$ is successful, $\mathcal{C}_{II}$ is also successful. The latter has a probability of $1/q_{ID}$ to successfully guess the identity on which $\mathcal{B}_{II}$ produces its attack. □

Let $C = \mathsf{CL\text{-}Encrypt}^{CPA}(mpk, ID, upk_{ID}, m;\ coin)$ be the CL-ENC algorithm described above, where $coin$ is the randomness involved in the generation of C. Let the corresponding decryption algorithm be denoted by $\mathsf{CL\text{-}Decrypt}^{CPA}(usk_{ID}, partial\_key_{ID}, C)$. To transform this CPA-secure CL-ENC scheme to a CCA-secure one, the following modifications are proposed by Libert and Quisquater [17], where the superscripts of CL-Encrypt and CL-Decrypt are replaced with CCA.

$$\mathsf{CL\text{-}Encrypt}^{CCA}(mpk, ID, upk_{ID}, m) := \mathsf{CL\text{-}Encrypt}^{CPA}(mpk, ID, upk_{ID}, m\|coin;\ r)$$

where $r = H(m\|coin\|upk_{ID}\|ID)$ where H is a hash function. For security analysis, it is considered to behave as a random oracle [5]. The decryption algorithm is modified as follows:

1. Compute $(m'\|coin') \leftarrow \mathsf{CL\text{-}Decrypt}^{CPA}(usk_{ID}, partial\_key_{ID}, C)$.
2. If the output is $\perp$, output $\perp$ and halt. Otherwise, compute

$$C' \leftarrow \mathsf{CL\text{-}Encrypt}^{CPA}(mpk, ID, upk_{ID}, m'\|coin';\ H(m'\|coin'\|upk_{ID}\|ID)).$$

3. If $C' = C$, output m', otherwise, output $\perp$.

Corollary 1. *The modification above is secure in **Game Enc-I** and **Game Enc-II** as defined in Sec. 2 under the random oracle model.*

It is straightforward by applying the proof of [17, Theorem 1].

Acknowledgements

We would like to thank Ron Steinfeld for his valuable comments and suggestions.

References

1. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *Proc. ASIACRYPT 2003*, pages 452–473. Springer-Verlag, 2003. LNCS 2894.
2. S. S. Al-Riyami and K. G. Paterson. CBE from CL-PKE: A generic construction and efficient schemes. In *8th International Workshop on Theory and Practice in Public Key Cryptography (PKC 2005)*, pages 398–415. Springer, 2005. LNCS 3386.
3. J. Baek, R. Safavi-Naini, and W. Susilo. Certificateless public key encryption without pairing. In *8th Information Security Conference (ISC'05)*, pages 134–148. Springer, 2005. LNCS 3650.
4. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In *Proc. EUROCRYPT 2004*, pages 268–286. Springer-Verlag, 2004. LNCS 3027 (Full paper is available at Bellare's homepage URL: `http://www-cse.ucsd.edu/users/mihir`).
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993. ACM.
6. K. Bentahar, P. Farshim, J. Malone-Lee, and N. P. Smart. Generic construction of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058, 2005. `http://eprint.iacr.org/2005/058`.
7. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Proc. CRYPTO 2001*, pages 213–229. Springer-Verlag, 2001. LNCS 2139.
8. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003.
9. Z. H. Cheng and R. Comley. Efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/012, 2005. `http://eprint.iacr.org/2005/012`.
10. 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`.
11. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Proc. CRYPTO 99*, pages 537–554. Springer-Verlag, 1999. LNCS 1666.
12. D. Galindo, P. Morillo, and C. Ràfols. Breaking Yum and Lee generic constructions of certificate-less and certificate-based encryption schemes. In *3rd European PKI Workshop: Theory and Practice (EuroPKI 2006)*, pages 81–91. Springer, 2006. LNCS 4043.
13. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308, April 1988.
14. B. C. Hu, D. S. Wong, Z. Zhang, and X. Deng. Key replacement attack against a generic construction of certificateless signature. In *Information Security and Privacy: 11th Australasian Conference, ACISP 2006*, pages 235–246. Springer-Verlag, 2006. LNCS 4058.
15. X. Huang, W. Susilo, Y. Mu, and F. Zhang. On the security of certificateless signature schemes from Asiacrypt 2003. In *Cryptology and Network Security, 4th International Conference, CANS 2005*, pages 13–25. Springer-Verlag, 2005. LNCS 3810.
16. X. Li, K. Chen, and L. Sun. Certificateless signature and proxy signature schemes from bilinear pairings. *Lithuanian Mathematical Journal*, 45(1):76–83, 2005.
17. B. Libert and J.-J. Quisquater. On constructing certificateless cryptosystems from identity based encryption. In *9th International Conference on Theory and Practice in Public Key Cryptography (PKC 2006)*, pages 474–490. Springer, 2006. LNCS 3958.

18. A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. CRYPTO 84*, pages 47–53. Springer, 1984. LNCS 196.
19. D. H. Yum and P. J. Lee. Generic construction of certificateless encryption. In *ICCSA'04*, pages 802–811. Springer, 2004. LNCS 3043.
20. D. H. Yum and P. J. Lee. Generic construction of certificateless signature. In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004*, pages 200–211. Springer-Verlag, 2004. LNCS 3108.
21. D. H. Yum and P. J. Lee. Identity-based cryptography in public key management. In *EuroPKI'04*, pages 71–84. Springer, 2004. LNCS 3093.
22. Z. Zhang, D. Wong, J. Xu, and D. Feng. Certificateless public-key signature: Security model and efficient construction. In *4th International Conference on Applied Cryptography and Network Security (ACNS 2006)*, pages 293–308. Springer, 2006. LNCS 3989.

A Definitions

In the following, we review the definitions of Standard Signature (SS) schemes, Public Key Encryption (PKE) schemes, ID-based Signature (IBS) schemes and ID-based Encryption (IBE) schemes.

Let $\Pi^{SS} = (\mathsf{Setup}^{SS}, Sign^{SS}, Ver^{SS})$ be an SS scheme, where the three algorthms are polynomial-time and all of them may be randomized although the last one is usually not. Setup^{SS} takes 1^k for some security parameter $k \in \mathbb{N}$, outputs a public/secret key pair (upk^{SS}, usk^{SS}). $Sign^{SS}$ takes usk^{SS} and message m, outputs a signature σ^{SS}. Ver^{SS} takes upk^{SS}, message m and signature σ^{SS}, and outputs 1 or 0 for accept or reject. For correctness, we require that for any $k \in \mathbb{N}$, $(upk^{SS}, usk^{PKE}) \leftarrow \mathsf{Setup}^{SS}(1^k)$ and any message m in the message space defined by the public key, $Ver^{SS}(upk^{SS}, m, Sign^{SS}(usk^{SS}, m)) = 1$.

Let $\Pi^{PKE} = (\mathcal{K}^{PKE}, \mathcal{E}^{PKE}, \mathcal{D}^{PKE})$ be a PKE scheme which consists of three polynomial-time algorithms. All of them may be randomized although the last one is usually not. $\mathcal{K}^{PKE}$ takes 1^k and outputs a public/secret key pair (upk^{PKE}, usk^{PKE}). $\mathcal{E}^{PKE}$ takes upk^{PKE} and some message m, and outputs a ciphertext C. $\mathcal{D}^{PKE}$ takes usk^{PKE} and C, and outputs either a message m or a symbol $\perp$ meaning that the decryption failed. For correctness, we require that for any $k \in \mathbb{N}$, $(usk^{PKE}, upk^{PKE}) \leftarrow \mathcal{K}^{PKE}(1^k)$ and any message m in the message space defined by the public key, $m = \mathcal{D}^{PKE}(usk^{PKE}, \mathcal{E}^{PKE}(upk^{PKE}, m))$.

Let an IBS scheme $\Pi^{IBS} = (\mathsf{Setup}^{IBS}, \mathsf{Extract}^{IBS}, Sign^{IBS}, Ver^{IBS})$ be denoted by four polynomial-time algorithms. All of these algorithms may be randomized although the last one is usually not. Setup^{IBS} takes 1^k and outputs a master public/secret key pair (mpk^{IBS}, msk^{IBS}). $\mathsf{Extract}^{IBS}$ takes msk^{IBS} and user identity $ID \in \{0,1\}^*$, and outputs a user secret key usk^{IBS}. $Sign^{IBS}$ takes usk^{IBS}, ID and a message m, and outputs a signature σ^{IBS}. Ver^{IBS} takes mpk^{IBS}, ID, a message m and a signature σ^{IBS}, and outputs 1 or 0 for accept or reject. For correctness, we require that $Ver^{IBS}(mpk^{IBS}, ID, m, Sign^{IBS}(usk^{IBS}, ID, m)) = 1$ provided that $(msk^{IBS}, mpk^{IBS}) \leftarrow \mathsf{Setup}^{IBS}(1^k)$, $usk^{IBS} \leftarrow \mathsf{Extract}^{IBS}(msk^{IBS}, ID)$ and message m in the message space according to the scheme specification.

Let $\Pi^{IBE} = (\mathsf{Setup}^{IBE}, \mathsf{Extract}^{IBE}, \mathcal{E}^{IBE}, \mathcal{D}^{IBE})$ be an IBE scheme consisting of four polynomial-time algorithms. All of them may be randomized although the last one is usually not. Setup^{IBE} and $\mathsf{Extract}^{IBE}$ are similar to Setup^{IBS} and $\mathsf{Extract}^{IBS}$, respectively. We use (msk^{IBE}, mpk^{IBE}) to represent master secret/public key pair and use usk^{IBE} to represent user secret key. $\mathcal{E}^{IBE}$ takes mpk^{IBE}, ID and message m, and outputs a ciphertext C. $\mathcal{D}^{IBE}$ takes usk^{IBE}, ID and ciphertext C, and outputs either a message m or a symbol $\perp$ meaning that the decryption failed. For correctness, we require that the equation $m = \mathcal{D}^{IBE}(usk^{IBE}, ID, \mathcal{E}^{IBE}(mpk^{IBE}, ID, m))$ holds as long as $(msk^{IBE}, mpk^{IBE}) \leftarrow \mathsf{Setup}^{IBE}(1^k)$, $usk^{IBE} \leftarrow \mathsf{Extract}^{IBE}(msk^{IBE}, ID)$ and message m in the message space defined by the master public key.