9-2011

Certificateless cryptography with KGC trust level 3

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Chik How TAN

Theoretical Computer Science 412 (2011) 5446–5457

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Certificateless cryptography with KGC trust level 3

Guomin Yang *, Chik How Tan

Temasek Laboratories, National University of Singapore, Singapore

ARTICLE INFO

Article history:
Received 25 January 2011
Received in revised form 9 May 2011
Accepted 9 June 2011
Communicated by X. Deng

Keywords:
Certificateless cryptography
Public key encryption
Digital signature
Trust hierarchy

ABSTRACT

A normal certificateless cryptosystem can only achieve KGC trust level 2 according to the trust hierarchy defined by Girault. Although in the seminal paper introducing certificateless cryptography, Al-Riyami and Paterson introduced a binding technique to lift the KGC trust level of their certificateless schemes to level 3, many subsequent work on certificateless cryptography just focused on the constructions of normal certificateless schemes, and a formal study on the general applicability of the binding technique to these existing schemes is still missing. In this paper, to address the KGC trust level issue, we introduce the notion of Key Dependent Certificateless Cryptography (KD-CLC). Compared with conventional certificateless cryptography, KD-CLC can achieve stronger security, and more importantly, KGC trust level 3. We then study generic techniques for transforming conventional CLC to KD-CLC. We start with the binding technique by Al-Riyami and Paterson, and show that there are some technical difficulties in proving that the binding technique is generally applicable. However, we show that a slightly modified version of the binding technique indeed can be proved to work under the random oracle assumption. Finally, we show how to perform the transformation using a standard cryptographic primitive instead of a random oracle.

1. Introduction

Certificateless Cryptography (CLC), introduced by Al-Riyami and Paterson [1], aims to avoid the drawbacks of both traditional public key cryptography which requires a public key infrastructure (PKI), and identity-based cryptography [17] which has the inherent key escrow problem. In a normal certificateless cryptosystem, a user secret key *usk* is derived from two partial secrets: one is an identity-based secret key (also known as partial secret key) *psk* generated by a Key Generation Center (KGC) based on the user's identity *ID*, and the other is a user self-generated secret key *sk* which corresponds to an uncertified public key *pk*. In many existing certificateless cryptosystems (e.g. [4,10,11,5,12]), the user secret key is simply set as $usk = (psk, sk)$.

Since there is no authentication information (such as an X.509 certificate in a PKI) for the user public key *pk*, an adversary can replace the public key either in the transmission or in a public directory with another public key. If a key replacement attack happens, then the security of a certificateless cryptosystem would just rely on its identity-based component. On the other hand, if the partial secret key *psk* of a user is leaked, then an adversary can always launch the key replacement attack to break a conventional type certificateless cryptosystem. Because of this reason, in most of the existing certificateless cryptosystems, the KGC can only be trusted by all the users. Recall the trust hierarchy by Girault [8] for public key cryptography.

Level 1. The "trusted" authority (e.g. the CA in a PKI, the KGC in an identity-based or certificateless cryptosystem) knows the secret key of any user.

* Corresponding author. Tel.: +65 65161147.
E-mail addresses: tslyg@nus.edu.sg (G. Yang), tsltch@nus.edu.sg (C.H. Tan).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.06.015

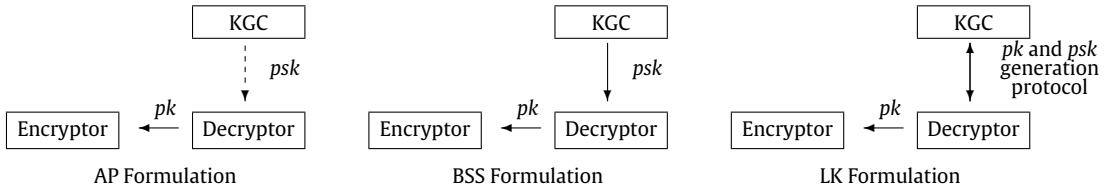

Fig. 1. The existing CLE architectures.

Level 2. The authority cannot compute users' secret keys. However, it can still impersonate a user by generating false guarantees (e.g. false certificates in a PKI, false public keys in a certificateless cryptosystem).

Level 3. The authority cannot compute users' secret keys, and it can be proven that it generates false guarantees of users if it does so.

We can see that identity-based cryptosystems [17] fall into Level 1, conventional certificateless cryptosystems fall into level 2, and a traditional PKI can achieve level 3.

In order to achieve the same trust level as that of a traditional PKI, Al-Riyami and Paterson proposed in their seminal paper [1] a simple binding technique for their certificateless cryptosystems without formal security proofs. However, most of the subsequent research work on certificateless cryptography ignored this important issue and just focused on designing certificateless schemes with KGC trust level 2, and it is unknown if there exist general and provably secure techniques to lift the KGC trust level of these existing schemes to level 3 — the same level as is enjoyed in a traditional PKI.

Our work. In this paper, to address the KGC trust level issue, we formalize the notion of *Key Dependent Certificateless Cryptography (KD-CLC)*. In a key dependent certificateless cryptosystem, the user partial secret key is generated in the following way: a user with identity ID first generates a public/private key pair (pk, sk), and sends pk to the KGC, who then generates the partial secret key psk based on both pk and ID. The advantage of this approach is that even if psk is exposed, an outside adversary cannot break the system by launching a key replacement attack. The reason is that for a replaced pk', the adversary needs to know the (new) partial secret key psk' corresponding to ID and pk', which can only be generated by the KGC. Now, the same statement as in [1] can be made here:

> "A KGC who replaces an entity's public key will be implicated in the event of a dispute: the existence of two working public keys for an identity can only result from the existence of two partial secret keys binding that identity to two different public keys; only the KGC could have created these two partial secret keys."

Then we propose a formal security model for key dependent certificateless cryptography, and study the problem of transforming conventional type certificateless cryptosystems into their key dependent counterparts. A good starting point is to consider the binding technique by Al-Riyami and Paterson. The idea is very simple: after a user with identity ID has created a public key pk, it simply uses $ID\|pk$ ($\|$ denotes string concatenation) as the "identity" for partial secret key generation. Although the idea looks reasonable, we show that some difficulties arise when one wants to formally prove this technique works. On the other hand, we show that a slightly modified binding can be proved to work: instead of using $ID\|pk$, we use $H(ID\|pk)$ where H is a cryptographic hash function. We prove that in the random oracle model, this simple (but useful) binding technique can transform any conventional certificateless encryption or signature scheme (with KGC trust level 2) into a key dependent scheme (with trust level 3). Finally, we show how to perform the transformation using another standard cryptographic primitive – a Trapdoor Hash Function, instead of a random oracle.

Paper organization. In the next section, we review some related work on certificateless cryptography. In Section 3, we formally define key dependent certificateless encryption (KD-CLE) and two generic constructions (with and without random oracle, respectively) of KD-CLE schemes from conventional CLE schemes. Then in Section 4, we show that our generic transformations can also be applied to construct key dependent certificateless signature schemes. The paper is concluded in Section 5.

2. Related work

Certificateless cryptography, introduced by Al-Riyami and Paterson [1] with the purpose of avoiding the drawbacks of both traditional public key cryptography and identity-based cryptography [17], has drawn a lot of attentions in recent years. A detailed survey on certificateless encryption schemes can be found in [5].

According to the categorization by Dent [6], we can separate existing certificateless cryptosystems into three categories: AP Formulation [1], BSS Formulation [3], and LK Formulation [14], which are demonstrated in Fig. 1. The dotted arrow denotes the fact that the public key can be published before the partial secret key psk is obtained. Most of the existing certificateless cryptosystems (e.g. [15,7,10,11,5,4,12]) follow the AP formulation.

In [16], Liu et al. introduced the notion of self-generated-certificate public key cryptography which can prevent the Denial-of-Decryption attacks. They also proposed a generic construction of self-generated-certificate encryption scheme based on a certificateless encryption scheme and a certificateless signature scheme. It is observed by Dent [6] that the minimum requirement for a certificateless encryption scheme to achieve Denial-of-Decryption security is that it is expressed

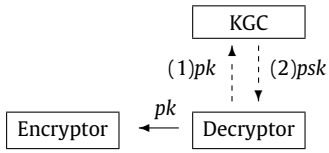

Fig. 2. Our KD-CLE architecture.

in the BSS or LK formulation (i.e. a user can only publish his/her public key after receiving the partial secret key from the KGC).

In this paper, we present a different CLE architecture which is shown in Fig. 2. The KD-CLE architecture differs from the conventional AP architecture in the essence that each partial secret key is tied with a user public key. As described earlier, in this way if an adversary replaces a user public key, then the adversary needs to compute the new partial secret key corresponding to the new public key, and by designing the KD-CLE system properly we can make sure that only the KGC can do so. So compared with the AP (and BSS) formulation, the KD-CLE architecture allows us to construct certificateless systems with KGC trust level 3. Notice that the KD-CLE architecture is different from a PKI, since the KGC only uses the user public key in the computation of the partial secret key, instead of issuing a certificate to the user. One additional remark is that although the LK formulation [14] also supports KGC trust level 3, it does not allow messages to be encrypted "into the future" (since in the LK formulation the user public key is available only after obtaining the partial secret key).

In [2], Au et al. defined certificateless security against a malicious-but-honest KGC who may generate its master public and private keys dishonestly but would not actively replace user public keys. Au et al.'s definition still falls in the AP architecture (Fig. 1), and hence does not support KGC trust level 3. On the other hand, in the security models given in this paper, we assume the master public and private keys are honestly generated by running the master key generation algorithm. So in general the two models (Au et al.'s and ours) are incomparable. However, an interesting point is that the *specific* malicious KGC attack presented in [2] cannot be directly applied to a key dependent certificateless cryptosystem, since at the time of master key generation, the KGC has no idea on the user public key that a user would choose.

It is worth noting that in [9], Hu et al. also attempted to define certificateless signature with KGC trust level 3. They defined the KGC trust level 3 security *separately* from the existential unforgeability under adaptive chosen message and chosen identity attacks (EUF-CMIA). Their definition basically requires that an adversary who has seen partial secret keys corresponding to $(ID, pk_1), (ID, pk_2), \ldots, (ID, pk_n)$ cannot come up with a valid partial secret key for the pair (ID, pk^*) where $pk^* \neq pk_i$ $(1 \leq i \leq n)$. Compared with Hu et al.'s work, we directly define KGC trust level 3 security inside the Type-I and Type-II EUF-CMIA security models, rather than using a separate definition.

3. Key dependent certificateless encryption

Definition 1. A Key Dependent Certificateless Encryption scheme $\mathcal{KD}$-$\mathcal{CLE}$ consists of the following algorithms.

- $\mathsf{Setup}(1^k)$ takes 1^k as input where k is the security parameter, and returns a master public/secret key pair (mpk, msk).
- $\mathsf{SetSecretValue}(mpk)$ takes mpk as input and returns a user secret value s.
- $\mathsf{SetPublicKey}(mpk, s)$ takes mpk and s as input and returns a user public key pk.
- $\mathsf{ExtractPartialKey}(msk, ID, pk)$ takes the master secret key msk, a user identity ID and a user public key pk as input and returns a user partial secret key psk.
- $\mathsf{SetPrivateKey}(mpk, s, psk)$ takes mpk, s and psk as input and returns a user private key usk. This algorithm is deterministic.
- $\mathsf{Enc}(mpk, ID, pk, m)$ takes mpk, a user identity ID, a user public key pk, and a message m as input and returns a ciphertext c.
- $\mathsf{Dec}(mpk, usk, c)$ takes mpk, a user private key usk, and a ciphertext c as input and returns a plaintext m or a decryption failure symbol $\perp$.

As usual, we assume the partial key generation process, which includes the submission of the user identity ID and the user public key pk to the KGC and the issuing of the partial secret key psk to the user, is done in a secure and authenticated manner.

3.1. KD-CLE Security Model

There are two types of adversaries, A_I and A_{II}. A Type-I adversary (or A_I) is allowed to perform key replacement attacks, while a Type-II adversary (or A_{II}) is a KGC who controls msk and also the generation of user partial secret keys.

Below we present the adversarial game. X can be instantiated with I or II in the description below, st denotes some state information, and m_0 and m_1 are two messages of equal length.

KD-Type-X-ATK Adversarial Game

1 $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$
2 $(ID^*, st, m_0, m_1) \leftarrow A_1{}^{\mathcal{O}}(mpk, [msk])$
3 $b \leftarrow \{0, 1\}$
4 $c^* \leftarrow \mathsf{Enc}(mpk, ID^*, pk^*_{ID^*}, m_b)$
5 $b' \leftarrow A_2{}^{\mathcal{O}}(c^*, st)$

In line 2 of the game, msk is only passed to the adversary in the KD-Type-II game. And in line 4 of the game, $pk^*_{ID^*}$ refers to the *current* public key of ID^*. We say the adversary wins the game if $b' = b$. The advantage of the adversary is defined to be

$$\mathbf{Adv}^{\text{KD-Type-X-ATK}}_{\mathcal{KD}\text{-}\mathcal{CLE},A}(k) = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

where ATK can be CPA or CCA. We now turn to the oracles $\mathcal{O}$ available to the adversary:

- **RequestPublicKey**(ID): Given an identity ID, this oracle returns to the adversary pk_{ID}. If the identity has no associated public key, then a public key is generated by running the SetSecretValue, and SetPublicKey algorithms.
- **ReplacePublicKey**(ID, pk'): Given an identity ID and a public key pk' in the public key space, this oracle allows the adversary to replace ID's public key with pk'.
- **ExtractPartialPrivateKey**(ID, pk): Given an identity ID and a public key pk, this oracle returns the partial secret key psk corresponding to identity ID and public key pk.
- **ExtractFullPrivateKey**(ID): Given an identity ID, this oracle returns the full user private key usk_{ID}.
- **StrongDecrypt**(ID, c): Given an user identity ID and a ciphertext c, this oracle returns the decryption of c, where the decryption is performed using the user private key corresponding to the *current* public key.

KD-Type-I-CCA Security: The adversary has the following restrictions in accessing the oracles

1. A cannot extract the full private key for any identity if the corresponding public key has been replaced.
2. A cannot extract the full private key for ID^*.
3. A cannot replace the public key for ID^* before c^* has been issued *and* make a **ExtractPartialSecretKey**(ID^*, $pk^*_{ID^*}$) query (at any point).
4. A_2 cannot make the StrongDecrypt query on (ID^*, c^*) unless at the time of this query user ID^*'s public key is not $pk^*_{ID^*}$.

KD-Type-II-CCA security: In the KD-Type-II-CCA game, the adversary is given the master secret key msk, and has the following restrictions in accessing the oracles

1. A cannot extract the full private key for any identity if the corresponding public key has been replaced.
2. A cannot extract the full private key for ID^*.
3. A cannot make any partial secret key extraction query.[1]
4. A_1 cannot output an identity ID^* for which it has replaced the public key.
5. A_2 cannot make the StrongDecrypt query on (ID^*, c^*) unless at the time of this query user ID^*'s public key is not $pk^*_{ID^*}$.

Definition 2. A key dependent certificateless encryption scheme $\mathcal{KD}\text{-}\mathcal{CLE}$ is said to be secure in the KD-Type-X-CCA model if for any PPT adversary A, $\mathbf{Adv}^{\text{KD-Type-X-CCA}}_{\mathcal{KD}\text{-}\mathcal{CLE},A}(k)$ is a negligible function of the security parameter k.

Comparison with conventional CLE security model. In our KD-CLE security model, we define **ExtractPartialSecretKey** query differently from that in the conventional CLE security model [1,4,5]. Besides the user identity, the query also takes a public key as input, and in the definition of KD-Type-I security, we disallow the adversary to extract the partial secret key with respect to ID^* and $pk^*_{ID^*}$ if the public key $pk^*_{ID^*}$ used to generate the challenge ciphertext is a replaced key, but *allow* the adversary to extract partial secret keys with respect to ID^* and any other pk_{ID^*} such that $pk_{ID^*} \neq pk^*_{ID^*}$. This captures the following key replacement attack carried out by an adversary: after the adversary learns a partial secret key of the user ID^*, the adversary replaces the original public key of ID^* with another key $pk^*_{ID^*}$, and request a challenge ciphertext under ID^* and $pk^*_{ID^*}$. We require a secure KD-CLE to be able to defend against such an attack. Notice that in the conventional Type-I game, such an adversarial behavior is not allowed since in this way an adversary can *always* break a conventional type certificateless encryption scheme.

Strong and weak decryption oracles. For the StrongDecrypt oracle defined in our model (and in the models of [1,7]), it is required that the decryption is performed using the user private key corresponding to the *current* public key of a user even if that public key is a replaced key provided by the adversary. This is a very strong oracle. Alternatively, one can define a weaker decryption oracle where the decryption is performed using the original user private key. This approach has been adopted in [10,11]. Also, in [4], Bentahar et al. defined another (intermediate) decryption oracle where if an adversary has replaced a public key and it subsequently requires a decryption query that involves a decryption with the corresponding secret value, it must supply this secret value to the decryption oracle.

[1] The partial secret key extraction query is unnecessary as a Type-II adversary can compute the partial secret keys.

3.2. A generic construction of KD-CLE from conventional CLE in ROM

The binding technique. A good starting point to construct KD-CLE is to apply the binding technique due to Al-Riyami and Paterson [1]. The binding technique works as follows: given a conventional certificateless encryption scheme $\mathcal{CLE}$ = (Setup, SetSecretValue, SetPublicKey, ExtractPartialKey, SetPrivateKey, Enc, Dec), we construct a KD-CLE scheme $\mathcal{KD}$-$\mathcal{CLE}$ = (Setup′, SetSecretValue′, SetPublicKey′, ExtractPartialKey′, SetPrivateKey′, Enc′, Dec′) as follows:

- Setup′(1^k): same as Setup(1^k).
- SetSecretValue′(mpk): same as SetSecretValue(mpk).
- SetPublicKey′(mpk, s): same as SetPublicKey(mpk, s).
- ExtractPartialKey′(msk, ID, pk): let $\tilde{ID} \leftarrow ID\|pk$, run

  $psk \leftarrow$ ExtractPartialKey$(msk, \tilde{ID})$

  and return psk.
- SetPrivateKey′(mpk, s, psk): same as SetPrivateKey(mpk, s, psk).
- Enc′(mpk, ID, pk, m): let $\tilde{ID} \leftarrow ID\|pk$, run

  $c \leftarrow$ Enc$(mpk, \tilde{ID}, pk, m)$

  and return c.
- Dec′(mpk, usk, c): same as Dec(mpk, usk, c).

Difficulties in giving a generic proof. Although it is possible to give individual security proofs for concrete KD-CLE schemes which are generated by applying the binding technique to conventional CLE schemes, there are some difficulties in giving a generic proof. One may try to prove that the above generic transformation would generate secure KD-CLE schemes based on the assumption that the underlying CLE is secure. By following a normal proof strategy, assume there exists an adversary A that can break the transformed scheme in the KD-Type-X-ATK game, we try to construct another adversary B that breaks the original CLE in the Type-X-ATK game. B would simulate the KD-Type-X-ATK game for A, and hopefully can relay his challenging ciphertext C^* encrypted using identity ID^* and public key pk^* to A and let A find out the solution for him. However, the following problem would occur in the simulation: when A requests a public key for the user ID^*, B needs to reply a public key pk^* to A, and in order to make B's own game consistent with the game simulated for A (i.e. A and B have the same challenging identity ID^* and the same challenging public key pk^*), B needs to supply $\tilde{ID}^* = ID^*\|pk^*$ to his own challenger via a RequestPublicKey query, but the problem is that with overwhelming probability the public key returned by B's challenger would not be the same as the public key pk^* in $\tilde{ID}^*$. To solve this problem, below we show a modified version of the binding technique.

A Hash-Binding Variant. We do a slight modification to the binding technique: instead of using $\tilde{ID} \leftarrow ID\|pk$, we use $\tilde{ID} \leftarrow H(ID\|pk)$ where $H : \{0,1\}^* \rightarrow \{0,1\}^k$ denotes a cryptographic hash function. In the following we prove that this slightly modified version works if we assume H to be a random oracle.

Theorem 1. *If $\mathcal{CLE}$ is secure in the conventional Type-I-CCA security model, then $\mathcal{KD}$-$\mathcal{CLE}$ is secure in the KD-Type-I-CCA security model assuming H is a random oracle.*

Proof. The proof is by contradiction. Assume there exists an adversary A (against $\mathcal{KD}$-$\mathcal{CLE}$) that has a non-negligible advantage in the KD-Type-I-CCA game, we construct another adversary B (against $\mathcal{CLE}$) which has a non-negligible advantage in the conventional Type-I-CCA game.

B simulates the KD-Type-I-CCA game for A. In the simulation, if a hash collision happens, B aborts with failure. B first passes mpk to A and answers A's queries as follows.

- H queries: this oracle is simulated by consistently returning random strings in $\{0,1\}^k$. That is, B maintains a table T which records the inputs to the hash oracle and their corresponding hash values.
- **RequestPublicKey**(ID): upon receiving this query, if the user ID has already been created, then the current public key is returned. Otherwise, B randomly selects a string $\tilde{ID} \leftarrow \{0,1\}^k$, and makes a **RequestPublicKey**$(\tilde{ID})$ query to its own oracle. After receiving the public key pk from the challenger, B sets $H(ID\|pk) = \tilde{ID}$ by putting $(ID\|pk, \tilde{ID})$ into table T, and returns pk to A.
- **ReplacePublicKey**(ID, pk'): B first simulates an H query with input $ID\|pk'$ to get a hash value $\tilde{ID}' = H(ID\|pk')$. B then makes a **RequestPublicKey**$(\tilde{ID}')$ query and a **ReplacePublicKey**$(\tilde{ID}', pk')$ query to its own oracles.
- **ExtractPartialSecretKey**(ID, pk): upon receiving this query, B first simulates an H query with input $ID\|pk$ to get a hash value $\tilde{ID} = H(ID\|pk)$. B then makes a **ExtractPartialSecretKey**$(\tilde{ID})$ query to its own oracle and returns to A the answer B gets from its oracle.

- **ExtractFullPrivateKey**(ID): upon receiving this query, B first simulates an H query with input $ID\|pk$ where pk is the public key of ID and get a hash value $\tilde{ID} = H(ID\|pk)$. B then makes a **ExtractFullPrivateKey**($\tilde{ID}$) query to its own oracle and returns to A the answer B gets from its oracle.
- **StrongDecrypt**(ID, c): let pk_{ID} denotes the public key of ID at the time of the query, and let $\tilde{ID} = H(ID\|pk_{ID})$. B makes a **StrongDecrypt**($\tilde{ID}$, c) query to its own oracle and returns to A the answer B gets from its oracle.

When A outputs (ID^*, m_0, m_1) as the challenge identity and messages, B outputs $(\tilde{ID}^*, m_0, m_1)$ to its own challenger where $\tilde{ID}^* = H(ID^*, pk^*_{ID^*})$. After receiving the challenging ciphertext c^* from the challenger, B returns c^* to A as the challenging ciphertext. B then simulates the oracle queries as before. Finally, when A outputs a bit b' and halts, B outputs b' as well and halts.

Analysis. If no hash collision occurs (denote this event by $\neg\mathsf{col}$), then the simulated KD-Type-I-CCA game is perfect, and also if A obeys all the restrictions in the KD-Type-I-CCA game, B obeys all the restrictions in the conventional Type-I-CCA game. Lastly, if A guesses the value of b correctly, so does B. So we have

$$\Pr[B \text{ wins in Type-I-CCA game}|\neg\mathsf{col}] = \Pr[A \text{ wins in KD-Type-I-CCA game}|\neg\mathsf{col}]$$

and hence

$$|\Pr[A \text{ wins in KD-Type-I-CCA game}] - \Pr[B \text{ wins in Type-I-CCA game}]| \leq \Pr[\mathsf{col}].$$

Therefore, we have

$$\begin{aligned}\mathbf{Adv}^{\text{KD-Type-I-CCA}}_{\mathcal{KD}\text{-}\mathcal{CLE},A}(k) &\leq \mathbf{Adv}^{\text{Type-I-CCA}}_{\mathcal{CLE},B}(k) + \Pr[\mathsf{col}] \\ &\leq \mathbf{Adv}^{\text{Type-I-CCA}}_{\mathcal{CLE},B}(k) + \frac{Q^2}{2^k}\end{aligned}$$

where Q denotes the total number of queries A has made in the game. □

Theorem 2. *If $\mathcal{CLE}$ is secure in the conventional Type-II-CCA security model, then $\mathcal{KD}$-$\mathcal{CLE}$ is secure in the KD-Type-II-CCA security model assuming H is a random oracle.*

Proof. The proof is similar to that of Theorem 1. Assume there exists an adversary A (against $\mathcal{KD}$-$\mathcal{CLE}$) that has a non-negligible advantage in the KD-Type-II-CCA game, we construct another adversary B (against $\mathcal{CLE}$) which has a non-negligible advantage in the conventional Type-II-CCA game.

Compared with the proof of Theorem 1, the only difference is in the setup of the simulation for the adversary A. In the conventional Type-II-CCA game, B's challenger would first run the Setup algorithm to generate a master public/secret key pair (mpk, msk) and passes the master key pair to B. To setup the simulation of the KD-Type-II-CCA game for A, after receiving (mpk, msk) from his challenger, B simply passes (mpk, msk) to A.

The rest of the proof is just the same as in the proof of Theorem 1. B simulates the KD-Type-II-CCA game for A by answering the **RequestPublicKey**, **ReplacePublicKey**, **ExtractFullPrivateKey**, **StrongDecrypt**, and H queries in the same way as in the proof of Theorem 1. By programming the random oracle H, B can make the simulated game for A consistent with its own game, so B can answer A's queries successfully by querying its own oracles, and relay his challenge to A. Finally, B outputs whatever A outputs.

Following the same analysis as in the proof of Theorem 1, if no hash collision occurs, B's simulation for A is perfect and if A has a non-negligible winning advantage in the simulated KD-Type-II-CCA game, B also has a non-negligible winning advantage in the Type-II-CCA game. Finally, since we model H as a random oracle, the probability of a hash collision event is negligible. □

3.3. A new generic transformation without random oracle

In this section, we study the (more challenging) problem of transforming conventional CLE to KD-CLE without random oracle. We show that the random oracle can be replaced by a standard cryptographic primitive – Trapdoor Hash Functions (THFs) [18] (also known as Chameleon Hash Functions [13]).

Definition 3 (*Trapdoor Hash Family [18]*)**.** A trapdoor hash function family consists of a pair $(\mathcal{I}, \mathcal{H})$ such that:

- $\mathcal{I}$ is a probabilistic polynomial-time key generation algorithm that on input 1^k outputs a hash/trapdoor key pair (HK, TK), such that the sizes of HK and TK are polynomially related to k.
- $\mathcal{H}$ is a family of randomized hash functions. Every hash function in $\mathcal{H}$ is associated with a hash key HK, and is applied to a message from a space $\mathcal{M}$ and a random element from a finite space $\mathcal{R}$. The output of the hash function h_{HK} does not depend on TK.

A trapdoor hash family $(\mathcal{I}, \mathcal{H})$ has the following properties:

1. Efficiency: Given a hash key HK and a pair $(m, r) \in \mathcal{M} \times \mathcal{R}$, the hash value $h_{HK}(m; r)$ is computable in polynomial time.

2. Collision resistance: there is no probabilistic polynomial-time algorithm $\mathcal{A}$ that on input HK outputs, with a probability which is not negligible, two pairs $(m_1, r_1), (m_2, r_2) \in \mathcal{M} \times \mathcal{R}$ that satisfy $m_1 \neq m_2$ and $h_{HK}(m_1; r_1) = h_{HK}(m_2; r_2)$ (the probability is over HK, where $(HK, TK) \leftarrow \mathcal{I}(1^k)$, and over the random coin tosses of algorithm $\mathcal{A}$).
3. Trapdoor collisions: There exists a probabilistic polynomial-time algorithm that given a pair $(HK, TK) \leftarrow \mathcal{I}(1^k)$, a pair $(m_1, r_1) \in \mathcal{M} \times \mathcal{R}$, and an additional message $m_2 \in \mathcal{M}$, outputs a value $r_2 \in \mathcal{R}$ such that:
   (a) $h_{HK}(m_1; r_1) = h_{HK}(m_2; r_2)$.
   (b) If r_1 is uniformly distributed in $\mathcal{R}$ then the distribution of r_2 is computationally indistinguishable from uniform in $\mathcal{R}$.

A trapdoor hash family based on DLP [18]

- The key generation algorithm $\mathcal{I}$. Choose at random a safe prime p (i.e., a prime p such that $q = (p-1)/2$ is prime) and an element $g \in \mathbb{Z}_p^*$ of order q. Choose a random element $\alpha \in \mathbb{Z}_q^*$ and compute $y = g^\alpha \pmod p$. The public hash key is (p, g, y) and the private trapdoor key is α.
- The hash family $\mathcal{H}$. For $HK = (p, g, y)$, $h_{HK} : \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{Z}_p^*$ is defined as follows: $h_{HK}(m; r) = g^m y^r \pmod p$.

Strong collision resistance. In this paper, we need a trapdoor hash family with *strong collision resistance*: There is no probabilistic polynomial-time algorithm $\mathcal{A}$ that on input HK outputs, with a probability which is not negligible, two pairs $(m_1, r_1), (m_2, r_2) \in \mathcal{M} \times \mathcal{R}$ that satisfy $(m_1, r_1) \neq (m_2, r_2)$ and $h_{HK}(m_1; r_1) = h_{HK}(m_2; r_2)$ (the probability is over HK, where $(HK, TK) \leftarrow \mathcal{I}(1^k)$, and over the random coin tosses of algorithm $\mathcal{A}$). It is easy to verify that the trapdoor hash family based on DLP also satisfies the strong collision resistance.

The transformation. Given a conventional certificateless encryption scheme $\mathcal{CLE}$ = (Setup, SetSecretValue, SetPublicKey, ExtractPartialKey, SetPrivateKey, Enc, Dec), and a trapdoor hash function family $(\mathcal{I}, \mathcal{H})$, we construct a KD-CLE scheme $\mathcal{KD}$-$\mathcal{CLE}$ = (Setup′, SetSecretValue′, SetPublicKey′, ExtractPartialKey′, SetPrivateKey′, Enc′, Dec′) as follows:

- $\mathsf{Setup}'(1^k)$: compute $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$ and $(HK, TK) \leftarrow \mathcal{I}(1^k)$. Set $mpk' = (mpk, HK)$ and $msk' = msk$.
- $\mathsf{SetSecretValue}'(mpk')$: compute $s \leftarrow \mathsf{SetSecretValue}(mpk)$, set the secret value as $s' = s$.
- $\mathsf{SetPublicKey}'(mpk', s')$: compute $pk \leftarrow \mathsf{SetPublicKey}(mpk, s')$, randomly select $r \leftarrow \mathcal{R}$, and set the public key as $pk' \leftarrow (pk, r)$.
- $\mathsf{ExtractPartialKey}'(msk', ID, pk')$: compute $\tilde{ID} \leftarrow h_{HK}(ID \| pk; r)$, run

  $$psk' \leftarrow \mathsf{ExtractPartialKey}(msk', \tilde{ID})$$

  and return psk'.
- $\mathsf{SetPrivateKey}'(mpk', s', psk')$: compute $usk' \leftarrow \mathsf{SetPrivateKey}(mpk, s', psk')$, and return usk'.
- $\mathsf{Enc}'(mpk', ID, pk', m)$: let $\tilde{ID} \leftarrow h_{HK}(ID \| pk; r)$, run

  $$c \leftarrow \mathsf{Enc}(mpk, \tilde{ID}, pk, m)$$

  and return c.
- $\mathsf{Dec}'(mpk', usk', c)$: return $\mathsf{Dec}(mpk, usk', c)$.

Theorem 3. *If $\mathcal{CLE}$ is secure in the conventional Type-I-CCA security model, and $(\mathcal{I}, \mathcal{H})$ is a secure trapdoor hash function family with strong collision resistance, then $\mathcal{KD}$-$\mathcal{CLE}$ is secure in the KD-Type-I-CCA security model.*

Proof. Similar to the proof for Theorem 1, assume there exists an adversary A (against $\mathcal{KD}$-$\mathcal{CLE}$) that has a non-negligible advantage in the KD-Type-I-CCA game, we construct another adversary B (against $\mathcal{CLE}$) which has a non-negligible advantage in the conventional Type-I-CCA game.

After B gets a master public key mpk, which is generated by running $\mathsf{Setup}(1^k)$, from his challenger, B runs $\mathcal{I}(1^k)$ to generate (HK, TK). B then passes (mpk, HK) to A and simulates the KD-Type-I-CCA game for A as follows.

- **RequestPublicKey**(ID): upon receiving this query, if the user ID has already been created, then the current public key is returned. Otherwise, B first generates a public key pk_0 by executing the SetSecretValue and SetPublicKey algorithms, B then randomly selects $r_0 \leftarrow \mathcal{R}$, and computes $\tilde{ID} \leftarrow h_{HK}(ID \| pk_0; r_0)$. After that B makes a **RequestPublicKey**$(\tilde{ID})$ query to its own oracle and receives a public key pk from his challenger. B then uses TK to find r such that $h_{HK}(ID \| pk; r) = h_{HK}(ID \| pk_0; r_0) = \tilde{ID}$, and returns $pk' = (pk, r)$ to A.
- **ReplacePublicKey**$(ID, (\hat{pk}, \hat{r}))$: B first computes $\tilde{ID}' = h_{HK}(ID \| \hat{pk}; \hat{r})$, and then makes a **RequestPublicKey**$(\tilde{ID}')$ query and a **ReplacePublicKey** $(\tilde{ID}', \hat{pk})$ query to its own oracles.
- **ExtractFullPrivateKey**(ID): upon receiving this query, B first computes $\tilde{ID} \leftarrow h_{HK}(ID \| pk; r)$ where (pk, r) is the public key of ID. B then makes a **ExtractFullPrivateKey**$(\tilde{ID})$ query to its own oracle and returns to A the answer B gets from its oracle.
- **ExtractPartialSecretKey**$(ID, (pk, r))$: upon receiving this query, B first computes $\tilde{ID} = h_{HK}(ID \| pk; r)$. B then makes a **ExtractPartialSecretKey**$(\tilde{ID})$ query to its own oracle and returns to A the answer B gets from its oracle.

– **StrongDecrypt**(ID, c): let (pk, r) denotes the public key of ID at the time of the query, and let $\tilde{ID} = h_{HK}(ID\|pk; r)$. B makes a **StrongDecrypt**$(\tilde{ID}, c)$ query to its own oracle and returns to A the answer B gets from its oracle.

When A outputs (ID^*, m_0, m_1) (denote $(pk^*_{ID^*}, r^*_{ID^*})$ the current public key of ID^*) as the challenge identity and messages, B outputs $(\tilde{ID}^*, m_0, m_1)$ to its own challenger where $\tilde{ID}^* = h_{HK}(ID^*\|pk^*_{ID^*}; r^*_{ID^*})$. After receiving the challenging ciphertext c^* from the challenger, B returns c^* to A as the challenging ciphertext. B then simulates the oracle queries as before. Finally, when A outputs a bit b' and halts, B outputs b' as well and halts.

Analysis. Since the trapdoor hash function family is strong collision resistant, a collision event

$$h_{HK}(ID\|pk; r) = h_{HK}(ID'\|pk'; r') \wedge (ID\|pk, r) \neq (ID'\|pk', r')$$

only happens with negligible probability ϵ_{col} in the simulated KD-Type-I-CCA game. If no collision occurs, then the simulated KD-Type-I-CCA game is perfect, and also if A obeys all the restrictions in the KD-Type-I-CCA game, B obeys all the restrictions in the conventional Type-I-CCA game. Lastly, if A guesses the value of b correctly, so does B. So we have

$$\mathbf{Adv}^{\text{KD-Type-I-CCA}}_{\mathcal{KD}\text{-}\mathcal{CLE},A}(k) \leq \mathbf{Adv}^{\text{Type-I-CCA}}_{\mathcal{CLE},B}(k) + \epsilon_{\mathsf{col}}. \quad \square$$

Remark. In our transformation, we require the trapdoor hash family to be strong collision resistant. The reason is that if an adversary can find a collision $h_{HK}(ID\|pk; r_1) = h_{HK}(ID\|pk; r_2)$ such that $r_1 \neq r_2$, then to win the game, the adversary A can ask a challenging ciphertext c^* under ID and (pk, r_1), then replace the public key of ID with (pk, r_2), and then ask a strong decryption query for c^*.

Theorem 4. *If $\mathcal{CLE}$ is secure in the conventional Type-II-CCA security model, and $(\mathcal{I}, \mathcal{H})$ is a secure trapdoor hash function family with strong collision resistance, then $\mathcal{KD}$-$\mathcal{CLE}$ is secure in the KD-Type-II-CCA security model.*

Proof. The proof is similar to that of Theorem 3. Suppose there exists an adversary A (against $\mathcal{KD}$-$\mathcal{CLE}$) that has a non-negligible advantage in the KD-Type-II-CCA game, we construct another adversary B (against $\mathcal{CLE}$) which has a non-negligible advantage in the conventional Type-II-CCA game.

Compared with the proof of Theorem 3, the only difference is in the setup of the simulation for the adversary A. In the Type-II-CCA game, B first receives a pair of master public and secret keys (mpk, msk), which are generated by running the Setup algorithm, from his challenger. After that, B runs $\mathcal{I}(1^k)$ to generate (HK, TK), and passes $mpk' = (mpk, HK)$ and $msk' = msk$ to A. Notice that TK is not passed to A.

The rest of the proof is the same as in the proof of Theorem 3. B simulates the KD-Type-II-CCA game for A by answering the **RequestPublicKey**, **ReplacePublicKey**, **ExtractFullPrivateKey**, and **StrongDecrypt** queries in the same way as in the proof of Theorem 3. Since B has the trapdoor key TK, B can make the simulated game for A consistent with its own game. In this way B can answer A's queries successfully by querying its own oracles, and relay his challenge to A. Finally, B outputs whatever A outputs.

Then Following the same analysis as in the proof of Theorem 3, since the trapdoor hash function family is strong collision resistant, a collision event happens only with negligible probability in the simulated KD-Type-II-CCA game. Also, if no collision event happens, then B's simulation for A is perfect, and if A has a non-negligible advantage in the simulated KD-Type-II-CCA game, B also has a non-negligible advantage in the conventional Type-II-CCA game. $\square$

One important point in our transformation in the standard model is that the trapdoor key TK is not part of the master secret key. As a result, in the proof of Theorem 4, when simulating the KD-Type-II-CCA game, B does not need to pass TK to the adversary A, so we can guarantee that no collision would occur in the simulated KD-Type-II-CCA game. In the real practice, a collision event can be used as an evidence showing that the KGC must have cheated (the trapdoor key TK must have been used).

3.4. A concrete KD-CLE scheme in the standard model

To give a clearer picture of our generic construction of KD-CLE schemes from conventional CLE schemes in the standard model, below we present a concrete KD-CLE scheme constructed from the Dent–Libert–Paterson CLE [7].

The construction uses bilinear map groups. Let $\mathbb{G}_1, \mathbb{G}_2$ denote two groups of prime order q, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ a bilinear map between them. The map satisfies the following properties:

1. Bilinear: For any $U, V \in \mathbb{G}_1$, and $a, b \in \mathbb{Z}_q$, we have $e(U^a, V^b) = e(U, V)^{ab}$;
2. Non-degenerate: If g is a generator of $\mathbb{G}_1$, then $e(g, g)$ is a generator of $\mathbb{G}_2$;
3. Computable: there exists an efficient algorithm to compute $e(U, V)$ for any $U, V \in \mathbb{G}_1$.

The KD-CLE scheme works as follows:

– Setup(1^k) : Let e denote a bilinear map as described above and g a generator of $\mathbb{G}_1$. Set $g_1 = g^\gamma$ for a random $\gamma \leftarrow \mathbb{Z}^*_q$. Without loss of generality, let θ_I denote the bit-length of a user identity and θ_G the bit-length of a $\mathbb{G}_1$ group element. Pick

a random $g_2 \leftarrow \mathbb{G}_1$, and vectors $\mathbf{u} = (u', u_1, \ldots, u_n) \in \mathbb{G}_1^{n+1}$, $\mathbf{v} = (v', v_1, \ldots, v_n) \in \mathbb{G}_1^{n+1}$ where $n = \theta_I + 2\theta_G + 2$. Define the Waters Hash Functions

$$F_u(S) = u' \prod_{i=1}^{n} u_i^{s_i} \quad \text{and} \quad F_v(W) = v' \prod_{i=1}^{n} v_i^{w_i}$$

where $S = s_1 s_2 \ldots s_n \in \{0,1\}^n$ and $W = w_1 w_2 \ldots w_n \in \{0,1\}^n$. Choose an n-bit safe prime $p' = 2q' + 1$, a generator $g' \in \mathbb{Z}_{p'}^*$ of order q', a random element $\alpha' \in \mathbb{Z}_{q'}^*$, and compute $y' = g'^{\alpha'} \bmod p'$. Define the trapdoor hash function

$$h_{p',g',y'}(A\|B\|C; r) = g'^{0\|A\|B\|C} y'^{r} \bmod p'$$

for $A \in \{0,1\}^{\theta_I}$, $B \in \mathbb{G}_1$, $C \in \mathbb{G}_1$, $r \in \mathbb{Z}_{q'}^*$ (notice that $|q'| = \theta_I + 2\theta_G + 1$, so we append a "0" bit before the bit-string "$A\|B\|C$"). Also select a collision resistant hash function $H' : \{0,1\}^* \to \{0,1\}^n$. The master public key is

$$mpk = (g, g_1, g_2, \mathbf{u}, \mathbf{v}, p', q', g', y', h_{p',g',y'}, H')$$

and the master secret is $msk = g_2^{\gamma}$.

- SetSecretValue(mpk): Randomly choose the secret value $s \leftarrow \mathbb{Z}_q^*$.
- SetPublicKey(mpk, s): Compute $(X, Y) = (g^s, g_1^s)$, randomly select $r \leftarrow \mathcal{R}$, and set the public key as $pk = (X, Y, r)$.
- ExtractPartialKey(msk, ID, pk): Compute $\tilde{ID} \leftarrow h_{p',g',y'}(ID\|X\|Y; r)$, randomly pick $\lambda \leftarrow \mathbb{Z}_q^*$, and return $psk \leftarrow (d_1, d_2) = (g_2^{\gamma} \cdot F_u(\tilde{ID})^{\lambda}, g^{\lambda})$.
- SetPrivateKey(mpk, s, psk): Parse psk into (d_1, d_2), randomly choose $\lambda' \leftarrow \mathbb{Z}_q^*$ and set the full user private key as

$$usk \leftarrow (\chi_1, \chi_2) = (d_1^s \cdot F_u(\tilde{ID})^{\lambda'}, d_2^s \cdot g^{\lambda'}) = (g_2^{\gamma s} \cdot F_u(\tilde{ID})^t, g^t)$$

  where $t = \lambda s + \lambda'$.
- Enc(mpk, ID, pk, m): To encrypt $m \in \mathbb{G}_2$, parse pk into (X, Y, r), then check that $e(X, g_1)/e(g, Y) = 1$. If so, compute $\tilde{ID} \leftarrow h_{p',g',y'}(ID\|X\|Y; r)$, randomly choose $\ell \leftarrow \mathbb{Z}_q^*$ and compute

$$C = (C_0, C_1, C_2, C_3) = (m \cdot e(Y, g_2)^{\ell}, g^{\ell}, F_u(\tilde{ID})^{\ell}, F_v(\rho)^{\ell})$$

  where $\rho = H'(C_0, C_1, C_2, \tilde{ID}, X, Y)$.
- Dec(mpk, usk, C): Parse usk into (χ_1, χ_2) and C into (C_0, C_1, C_2, C_3). Check that

$$e(C_1, F_u(\tilde{ID}) \cdot F_v(\rho)) = e(g, C_2 \cdot C_3)$$

  where $\tilde{ID} = h_{p',g',y'}(ID\|X\|Y; r)$ and $\rho = H'(C_0, C_1, C_2, \tilde{ID}, X, Y)$. If the condition does not hold, return $\perp$. Otherwise, return

$$m \leftarrow C_0 \cdot \frac{e(C_2, \chi_2)}{e(C_1, \chi_1)}.$$

Based on Theorems 3 and 4, and the results of [18,7], we can directly obtain the following corollary.

Corollary 1. *The concrete KD-CLE scheme presented above is both KD-Type-I-CCA and KD-Type-II-CCA secure under the 3-DDH assumption in the group $\mathbb{G}_1$, the Discrete Log assumption in the group defined by (p', q', g'), and the Collision Resistant assumption on the hash function H'.*

4. Key dependent certificateless signature

In this section, we define Key Dependent Certificateless Signature (KD-CLS) schemes and investigate the applicability of our generic transformations to CLS schemes.

Definition 4. A Key Dependent Certificateless Signature scheme $\mathcal{KD}$-$\mathcal{CLS}$ consists of the following algorithms.

- Setup(1^k) takes 1^k as input where k is the security parameter, and returns a master public/secret key pair (mpk, msk).
- SetSecretValue(mpk) takes mpk as input and returns a user secret value s.
- SetPublicKey(mpk, s) takes mpk and s as input and returns a user public key pk.
- ExtractPartialKey(msk, ID, pk) takes the master secret key msk, a user identity ID and a user public key pk as input and returns a user partial secret key psk.
- SetPrivateKey(mpk, s, psk) takes mpk, s and psk as input and returns a user private key usk. This algorithm is deterministic.
- Sig(mpk, usk, m) takes the master public key mpk, a user private key usk and a message m as input and returns a signature σ.
- Ver(mpk, ID, pk, m, σ) takes mpk, a user identity ID, a user public key pk, a message and a signature σ as input and returns a decision bit $b \leftarrow \{0,1\}$. This algorithm is deterministic.

4.1. KD-CLS security model

Similar to the security model of KD-CLE, we define two types of adversaries F_I and F_{II} against a KD-CLS scheme as follows.

KD-type-I-CMIA game: the existential unforgeability of a KD-CLS scheme under KD-Type-I adaptive chosen message and chosen identity attacks is defined in the following game:

Phase 1: The challenger runs the algorithm generates $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$ and returns mpk to F_I.

Phase 2: In this phase, the adversary F_I can adaptively access the **RequestPublicKey**, **ReplacePublicKey**, **ExtractPartial SecretKey**, and **ExtractFullPrivateKey** queries as defined before. But F_I cannot extract the full private key for any user if the corresponding public key has been replaced. In addition, F_I can adaptively make signing queries defined below

Sign(ID, m): Given an identity ID and a message m, this oracle returns a signature σ such that $\mathsf{Ver}(mpk, ID, pk, m, \sigma) = 1$ where pk is the *current* public key of user ID at the time the query is performed.

Phase 3: After all the queries, F_I outputs a forgery (ID^*, m^*, σ^*). Let $pk^*_{ID^*}$ be the current public key of the user ID^*. We say F_I wins the game if:

1. F_I has never extracted the full private key for ID^*,
2. F_I has never made a **ReplacePublicKey**$(ID^*, pk^*_{ID^*})$ query *and* an **ExtractPartialSecretKey**$(ID^*, pk^*_{ID^*})$ query,
3. F_I has never submitted (ID^*, m^*) to the signing oracle unless at the time of this query the public key of ID^* is not $pk^*_{ID^*}$,
4. $1 \leftarrow \mathsf{Ver}(mpk, ID^*, pk^*_{ID^*}, m^*, \sigma^*)$.

KD-type-II-CMIA game: the existential unforgeability of a KD-CLS scheme under KD-Type-II adaptive chosen message and chosen identity attacks is defined in the following game:

Phase 1: The challenger runs the algorithm generates $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$ and returns (mpk, msk) to F_{II}.

Phase 2: In this phase, the adversary F_{II} can adaptively access the **RequestPublicKey**, **ReplacePublicKey**, and **Extract FullPrivateKey** queries as defined before. But F_{II} cannot extract the full private key for any user if the corresponding public key has been replaced. In addition, F_{II} can adaptively make signing queries defined below

Sign(ID, m): Given an identity ID and a message m, this oracle returns a signature σ such that $\mathsf{Ver}(mpk, ID, pk, m, \sigma) = 1$ where pk is the *current* public key of user ID at the time the query is performed.

Phase 3: After all the queries, F_{II} outputs a forgery (ID^*, m^*, σ^*). Let $pk^*_{ID^*}$ be the current public key of the user ID^*. We say F_{II} wins the game if:

1. F_{II} has never extracted the full private key for ID^*,
2. F_{II} has never replaced the public key for ID^*,
3. F_{II} has never submitted (ID^*, m^*) to the signing oracle,
4. $1 \leftarrow \mathsf{Ver}(mpk, ID^*, pk^*_{ID^*}, m^*, \sigma^*)$.

The advantage of an adversary F is defined to be

$$\mathbf{Adv}^{\text{KD-Type-X-CMIA}}_{\mathcal{KD}\text{-}\mathcal{CLS},F}(k) = \Pr[F \text{ wins in the KD-Type-X-CMIA game}].$$

Definition 5. A key dependent certificateless signature scheme $\mathcal{KD}$-$\mathcal{CLS}$ is said to be secure in the KD-Type-X-CMIA model if for any PPT adversary F, $\mathbf{Adv}^{\text{KD-Type-X-CMIA}}_{\mathcal{KD}\text{-}\mathcal{CLS},F}(k)$ is a negligible function of the security parameter k.

4.2. Generic transformations

It is straightforward to see that the generic transformations we presented in Section 3 for constructing KD-CLE can also be applied to construct KD-CLS from conventional CLS. For completeness, we present the transformations in Fig. 3.

Theorem 5. *If $\mathcal{CLS}$ is secure in the conventional Type-X-CMIA security model and H is a random oracle, then $\mathcal{KD}$-$\mathcal{CLS}_{ROM}$ is secure in the KD-Type-X-CMIA security model.*

Proof (*Sketch*)**.** The proof is very similar to that of Theorems 1 and 2, given an adversary F that can break the KD-Type-X-CMIA security of $\mathcal{KD}$-$\mathcal{CLS}_{\mathsf{ROM}}$, we can construct another adversary E to break the Type-X-CMIA security of $\mathcal{CLS}$. E simulates the KD-Type-X-CMIA game for F by answering F's **RequestPublicKey**, **ReplacePublicKey**, **ExtractPartialSecretKey**, **ExtractFullPrivateKey**, and H queries in the same way as in the proof of Theorem 1. To answer F's signing queries **Sign**(ID, m), E first simulates an H query to get $\tilde{ID} = H(ID\|pk)$ where pk is the public key of ID, and then makes a signing query to its own signing oracle with input $(\tilde{ID}, m)$ and returns the answer from the its oracle to F. By programming the random oracle, E can make the simulated game for F consistent with its own game, so E can answer F's queries successfully by querying its own oracles. Finally, if F can successfully produce a forgery in the KD-Type-X-CMIA game, E can also produce a valid forgery in the Type-X-CMIA security game and break the underlying conventional certificateless signature scheme. □

Also, by following the proofs of Theorems 3–5, the following theorem can be proved in a straightforward way.

Theorem 6. *If $\mathcal{CLS}$ is secure in the conventional Type-X-CMIA security model, and $(\mathcal{I}, \mathcal{H})$ is a secure trapdoor hash function family with strong collision resistance, then $\mathcal{KD}$-$\mathcal{CLS}_{THF}$ is secure in the KD-Type-X-CMIA security model.*

Input: a conventional certificateless signature scheme

$\mathcal{CLS} = (\mathsf{Setup}, \mathsf{SetSecretValue}, \mathsf{SetPublicKey}, \mathsf{ExtractPartialKey}, \mathsf{SetPrivateKey}, \mathsf{Sig}, \mathsf{Ver})$

$\mathcal{KD}$-$\mathcal{CLS}_{\mathsf{ROM}}$: Key Dependent Certificateless Signature Based on Random Oracle H

- $\mathsf{Setup}'(1^k)$: compute $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$, set $mpk' = (mpk, H)$ and $msk' = msk$.
- $\mathsf{SetSecretValue}'(mpk')$: compute

$$s \leftarrow \mathsf{SetSecretValue}(mpk),$$

  set the secret value as $s' = s$.
- $\mathsf{SetPublicKey}'(mpk', s')$: compute

$$pk \leftarrow \mathsf{SetPublicKey}(mpk, s'),$$

  and set the public key as $pk' = pk$.
- $\mathsf{ExtractPartialKey}'(msk', ID, pk')$: compute

$$\tilde{ID} \leftarrow H(ID\|pk'),$$

$$psk' \leftarrow \mathsf{ExtractPartialKey}(msk', \tilde{ID}),$$

  and return psk'.
- $\mathsf{SetPrivateKey}'(mpk', s', psk')$: compute

$$usk' \leftarrow \mathsf{SetPrivateKey}(mpk, s', psk'),$$

  and return usk'.
- $\mathsf{Sig}'(mpk', usk', m)$: compute

$$\sigma \leftarrow \mathsf{Sig}(mpk, usk', m)$$

  and return σ.
- $\mathsf{Ver}'(mpk', ID, pk', m, \sigma)$: compute

$$\tilde{ID} \leftarrow H(ID\|pk'),$$

  return $\mathsf{Ver}(mpk, \tilde{ID}, pk', m, \sigma)$.

$\mathcal{KD}$-$\mathcal{CLS}_{\mathsf{THF}}$: Key Dependent Certificateless Signature Based on Trapdoor Hash Family $(\mathcal{I}, \mathcal{H})$

- $\mathsf{Setup}'(1^k)$: compute $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$ and $(HK, TK) \leftarrow \mathcal{I}(1^k)$. Set $mpk' = (mpk, HK)$ and $msk' = msk$.
- $\mathsf{SetSecretValue}'(mpk')$: compute

$$s \leftarrow \mathsf{SetSecretValue}(mpk),$$

  set the secret value as $s' = s$.
- $\mathsf{SetPublicKey}'(mpk', s')$: compute

$$pk \leftarrow \mathsf{SetPublicKey}(mpk, s'),$$

  randomly select $r \leftarrow \mathcal{R}$, and set the public key as $pk' \leftarrow (pk, r)$.
- $\mathsf{ExtractPartialKey}'(msk', ID, pk')$: compute

$$\tilde{ID} \leftarrow h_{HK}(ID\|pk; r),$$

$$psk' \leftarrow \mathsf{ExtractPartialKey}(msk', \tilde{ID}),$$

  and return psk'.
- $\mathsf{SetPrivateKey}'(mpk', s', psk')$: compute

$$usk' \leftarrow \mathsf{SetPrivateKey}(mpk, s', psk'),$$

  and return usk'.
- $\mathsf{Sig}'(mpk', usk', m)$: compute

$$\sigma \leftarrow \mathsf{Sig}(mpk, usk', m)$$

  and return σ.
- $\mathsf{Ver}'(mpk', ID, pk', m, \sigma)$: compute

$$\tilde{ID} \leftarrow h_{HK}(ID\|pk; r),$$

  return $\mathsf{Ver}(mpk, \tilde{ID}, pk, m, \sigma)$.

Fig. 3. Generic Transformations from Conventional CLS to KD-CLS.

4.3. A concrete KD-CLS scheme

By applying our trapdoor hash based generic transformation to the certificateless signature scheme in [16], we can derive a KD-CLS in the standard model. Interestingly, the Setup, SetSecretValue, SetPublicKey, and ExtractPartialKey algorithms of this KD-CLS scheme are the same as those of the concrete KD-CLE scheme presented in Section 3.4, so we omit the details of these algorithms here.

- $\mathsf{SetPrivateKey}(mpk, s, psk)$: Parse psk into (d_1, d_2), and set the full user private key as

$$usk \leftarrow (\chi_1, \chi_2) = (d_1^s, d_2^s).$$

- $\mathsf{Sig}(mpk, usk, m)$: To sign a message $m \in \{0,1\}^*$, parse usk into (χ_1, χ_2), randomly selects $r_1, r_2 \leftarrow \mathbb{Z}_q^*$, compute $\tilde{ID} \leftarrow h_{p',g',y'}(ID\|X\|Y; r)$ (where $pk = (X, Y, r)$), and generate the signature as

$$\sigma \leftarrow (V, R_1, R_2) = (\chi_1 \cdot F_u(\tilde{ID})^{r_1} \cdot F_v(\tilde{m})^{r_2}, \chi_2 \cdot g^{r_1}, g^{r_2})$$

  where $\tilde{m} = H'(m)$.
- $\mathsf{Ver}(mpk, ID, pk, m, \sigma)$: Given a signature $\sigma = (V, R_1, R_2)$ for an identity ID and public key $pk = (X, Y, r)$ on a message m, a verifier first computes $\tilde{ID} \leftarrow h_{p',g',y'}(ID\|X\|Y; r)$ and $\tilde{m} = H'(m)$, and then checks whether

$$e(X, g_1) = e(Y, g) \quad \text{and} \quad e(V, g) = e(g_2, Y)e(F_u(\tilde{ID}), R_1)e(F_v(\tilde{m}), R_2).$$

The verifier accepts the signature if and only if the equations hold.

The following corollary can be obtained directly from Theorems 5 and 6, and the results in [18,16].

Corollary 2. *The KD-CLS scheme presented above is KD-Type-I-CMIA and KD-Type-II-CMIA secure under the Non-pairing-based Generalized Bilinear Diffie–Hellman (NGBDH) assumption in the group $\mathbb{G}_1$, the Discrete Log assumption in the group defined by (p', q', g'), and the Collision Resistant assumption on the hash function H'.*

5. Conclusion

We introduced the concept of Key Dependent Certificateless Cryptography (KD-CLC) in this paper. Compared with conventional certificateless cryptosystems, KD-CLC can achieve stronger security, and more importantly, KGC trust level 3. We then showed two generic ways (with and without random oracle, respectively) to transform conventional certificateless encryption and signature schemes into their key dependent counterparts. As a final remark, whether the binding technique introduced by Al-Riyami and Paterson can be proved to be a secure generic transformation remains an interesting open problem.

Acknowledgements

We thank the anonymous reviewers for their helpful comments and suggestions.

References

[1] Sattam S. Al-Riyami, Kenneth G. Paterson, Certificateless public key cryptography, in: Advances in Cryptology - ASIACRYPT 2003, pp. 452–473. Full paper available at: http://eprint.iacr.org/2003/126.
[2] Man Ho Au, Jing Chen, Joseph K. Liu, Yi Mu, Duncan S. Wong, Guomin Yang, Malicious KGC attack in certificateless cryptography, in: ASIACCS 2007.
[3] Joonsang Baek, Reihaneh Safavi-Naini, Willy Susilo, Certificateless public key encryption without pairing, in: Information Security Conference, ISC 2005, pp. 134–148.
[4] Kamel Bentahar, Pooya Farshim, John Malone-Lee, Nigel P. Smart, Generic constructions of identity-based and certificateless KEMs, J. Cryptology 21 (2) (2008) 178–199.
[5] Alexander W. Dent, A survey of certificateless encryption schemes and security models, Int. J. Inf. Sec. 7 (5) (2008) 349–377.
[6] Alexander W. Dent, A brief introduction to certificateless encryption schemes and their infrastructures, in: Proc. EuroPKI, 2009, pp. 1–16.
[7] Alexander W. Dent, Benoît Libert, Kenneth G. Paterson, Certificateless encryption schemes strongly secure in the standard model, in: Public Key Cryptography 2008, pp. 344–359.
[8] Marc Girault, Self-certified public keys, in: Advances in Cryptology - EUROCRYPT, 1991, pp. 490–497.
[9] Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang, Xiaotie Deng, Certificateless signature: a new security model and an improved generic construction, Des. Codes Cryptography 42 (2) (2007) 109–126.
[10] Qiong Huang, Duncan S. Wong, Generic certificateless encryption in the standard model, in: IWSEC 2007, pp. 278–291.
[11] Qiong Huang, Duncan S. Wong, Generic certificateless key encapsulation mechanism, in: ACISP 2007, pp. 215–229.
[12] Xinyi Huang, Yi Mu, Willy Susilo, Duncan S. Wong, Wei Wu, Certificateless signature revisited, in: ACISP 2007, pp. 308–322.
[13] Hugo Krawczyk, Tal Rabin, Chameleon signatures, in: NDSS 2000, pp. 143–154.
[14] Junzuo Lai, Weidong Kou, Self-generated-certificate public key encryption without pairing, in: Public Key Cryptography, 2007, pp 476–489.
[15] Benoît Libert, Jean-Jacques Quisquater, On constructing certificateless cryptosystems from identity based encryption, in: Public Key Cryptography 2006, pp. 474–490.
[16] Joseph K. Liu, Man Ho Au, Willy Susilo, Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model, in: ASIACCS 2007, pp. 273–283.
[17] Adi Shamir, Identity-based cryptosystems and signature schemes, in CRYPTO 1984, pp. 47–53.
[18] Adi Shamir, Yael Tauman, Improved online/offline signature schemes, in: Advances in Cryptology - CRYPTO, 2001, pp. 355–367.