Generic Certificateless Encryption in the Standard Model

Qiong Huang and Duncan S. Wong

Dept. of Computer Science
City University of Hong Kong
Hong Kong, China
{csqhuang, duncan}@cityu.edu.hk

Abstract. Despite the large number of certificateless encryption schemes recently proposed, many of them have been found to be insecure under a practical attack called *malicious-but-passive* KGC attack, since they all follow the same key generation procedure as that of the one proposed by Al-Riyami and Paterson in ASIACRYPT 2003. The only provably secure certificateless encryption scheme against this attack is due to Libert and Quisquater (PKC 2006). However, the security can only be shown in the random oracle model. In this paper, we first show that a scheme which has a different key generation procedure from that of Al-Riyami and Paterson also suffers from the malicious-but-passive KGC attack. Our attacking techniques are different from the previous attacks and may cause greater extent of damage than the previous ones. We also propose a generic construction of certificateless encryption which can be proven secure against this attack *in the standard model*. This generic scheme is not only the first one proven secure in the standard model, but is also very efficient to instantiate. We also describe how to use short signature and hybrid encryption to construct highly efficient instantiations of this generic scheme.

1 Introduction

In traditional public key cryptography, a user selects a public/private key pair (pk, sk) and publishes pk. A *certificate*, which essentially is a signature on the user's identity and pk issued by a *certification authority* (CA), will then be employed for indicating the relationship between the user and pk. This method works under the public key infrastructure (PKI) involves a lot of additional work for managing the certificates that include revocation, storage and distribution.

In 1984, Shamir [12], introduced the notion of identity-based cryptography, aiming to alleviate the existing problems in PKI by getting rid of certificates. A user can use an email address, an IP address or any other information related to his identity, that is publicly known and unique in the whole system, as his public key. There is a trusted party, called Key Generation Center (KGC), which is in charge of the user private key generation. The advantage of an identity-based cryptosystem is that anyone can simply use the user's identity to encrypt messages. This can be done even before the user gets its private key from the KGC. However, the user must also completely trust the KGC, which can impersonate the user and decrypt any of the user's ciphertexts. This issue is generally referred to as key escrow problem in identity-based cryptography.

In 2003, Al-Riyami and Paterson [1] introduced *certificateless cryptography*, which is intended to solve the key escrow problem that is inherent in identity-based cryptography, while at the same time, eliminate the use of certificates as in the traditional PKI. In a certificateless cryptosystem, the KGC is involved to issue a user *partial* key psk_{ID} for a user with identity ID. The user independently generates a user public/private key pair $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$, and publishes upk_{ID}. A message will then encrypted under *both* upk_{ID} and the user's identity ID. To decrypt a ciphertext, the user must have the knowledge of both the user partial key psk_{ID} and secret key usk_{ID}. Knowing only one of them does not allow the recovery of the plaintext.

Related Work. Since the introduction of certificateless cryptography [1], there have been many schemes proposed [17,7,10,6,2,11]. The original definition of certificateless encryption [1] consists of seven algorithms. It has recently been simplified to five [6] and has shown to be more versatile than the original one. In this paper, we also adopt the five-algorithm simplified version for defining a certificateless encryption scheme.

Yum and Lee proposed a generic certificateless encryption scheme in [16] which has later been shown to be insecure under the model of [1] by Libert and Quisquater [10]. In [10], the authors also proposed a generic certificateless encryption scheme. However, their scheme is only proven secure in the random oracle model, which is a heuristic method for showing the security of cryptographic schemes. The security may not preserve when the random oracle is replaced by a hash function, even if the scheme is reduced to some complexity (or number-theoretic) assumption. Recently, Liu et al. [11] proposed another certificateless encryption scheme which, to the best of our knowledge, is the first one in the standard model. However, as we will show in this paper, their scheme is insecure against *malicious-but-passive* KGC attack which is introduced as follows.

In [2], Au et al. considered another strong security model for certificateless cryptography, in which the user's trust on the KGC is further relaxed. By using the term introduced in [2], the KGC of a certificateless cryptosystem can be *malicious-but-passive*. This means the KGC can be malicious so that it may not follow the scheme specification for generating system parameters and master key, while it does not actively replace a user's public key or corrupt the user's secret key. The purpose of such a malicious-but-passive KGC is to compromise a user's secret key without being detected. Since the KGC does not need to replace the user's public key or compromise the user's machine for corrupting the user's secret key, in practice, it is very difficult to find out the occurrence of this attack. Consider a user who is the president of a country or an organization. It may have a great incentive for the KGC to launch the malicious-but-passive attack as ciphertexts for the president may contain some valuable and profitable information. The attack is practical, and the new security model would become much stronger and realistic than the old ones in this aspect if the attack is properly formalized.

Under the *malicious-but-passive* KGC attacking model, certificateless cryptosystems proposed in [1,7,9] have been shown to be insecure, since they all follow the same key generation procedure as that of [1]. The only provably secure certificateless encryption scheme against malicious-but-passive KGC attack currently available is due to Libert and Quisquater [10]. The proof is given in [2]. They showed that the IND-CPA-secure certificateless encryption scheme in [10] is also IND-CPA secure under the malicious-but-passive KGC adversarial model. Then, by applying the transformation technique of [10], any IND-CPA-secure scheme can be converted to an IND-CCA2-secure one. However, the final scheme is only secure in the random oracle model. There is no concrete or generic construction of certificateless encryption which is proven secure against malicious-but-passive KGC attacks in the standard model.

Our Results. We propose the *first generic* certificateless encryption scheme which is proven secure against malicious-but-passive KGC attacks *in the standard model*. The idea of our construction is simple. It is constructed based on three well-formalized primitives and can be considered as a sequential encryption as in [17,2], with an additional signature-based mechanism to defend against attacks discussed in [10], but without relying on the assumption of random oracles. The construction is also efficient. We will describe how to use short signature [4] and hybrid encryption to implement highly efficient instantiations of this generic scheme.

We show that a recently proposed certificateless encryption scheme [11] also suffers from the malicious-but-passive KGC attack. As mentioned above, schemes in [1,7,9] are vulnerable to the malicious-but-passive KGC attack described in [2] as they all follow the same key generation procedure as that of [1]. However, the attacking technique of [2] does not apply to the scheme in [11] as the key generation procedure is different. We propose two malicious-but-passive KGC attacks against the scheme in [11]. The first attack causes the same extent of damage as the attack described in [2] against [1,7,9]. The second attack may cause greater impact to the system as the KGC is able to decrypt ciphertexts which are for any user in the system without pre-selecting a target user. Note that our attacks do not refute the security claims made in [11], since the KGC in their security model launches attacks only after honestly generating the master public/private key pair (and system parameters).

Paper Organization. In Sec. 2, we define the certificateless encryption scheme and its security. In Sec. 3, we show that the certificateless encryption scheme in [11] is vulnerable to some new *malicious-but-passive* KGC attacks. Our generic construction of secure encryption schemes and its security analysis are provided in Sec. 4. We also describe some instantiations of the generic scheme based on short signature and hybrid encryption. The paper is concluded in Sec. 5.

2 Definition and Adversarial Model

A certificateless encryption scheme [1,2] consists of five (probabilistic) polynomial-time (PPT) algorithms:

- $\mathsf{MasterKeyGen}$: On input 1^k where $k \in \mathbb{N}$ is a security parameter, it generates a master public/private key pair (mpk, msk).
- $\mathsf{PartialKeyGen}$: On input msk and a user identity $\mathsf{ID} \in \{0,1\}^*$, it generates a user partial key psk_{ID}.
- $\mathsf{UserKeyGen}$: On input mpk and a user identity ID, it generates a user public/private key pair $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$.
- Enc: On input mpk, a user identity ID, a user public key upk_{ID} and a message m, it returns a ciphertext c.
- Dec: On input a user private key usk_{ID}, a user partial key psk_{ID}, and a ciphertext c, it returns the plaintext m.

In practice, the KGC (Key Generation Center) performs the first two algorithms: $\mathsf{MasterKeyGen}$ and $\mathsf{PartialKeyGen}$. The master public key mpk is then published and it is assumed that everyone in the system can get a legitimate copy of mpk. It is also assumed that the partial key is issued to the corresponding user via a secure channel so that no one except the intended user can get it. Every user in the system also performs $\mathsf{UserKeyGen}$ for generating its own public/private key pair and publishes the public key. The correctness requirement is defined in the conventional way. We skip the details and refer readers to [1,2] for details.

Adversarial Model. There are two types of security for a certificateless encryption scheme, Type-I security and Type-II security, along with two types of adversaries, $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. Adversary $\mathcal{A}_1$ models a malicious adversary which compromises the user private key usk_{ID} or replaces the user public key upk_{ID}, however, cannot compromise the master private key msk nor get access to the user partial key psk_{ID}. Adversary $\mathcal{A}_2$ models the malicious-but-passive KGC which controls the generation of the master public/private key pair, and that of any user partial key psk_{ID}. The following are five oracles which can be accessed by the adversaries.

– $\mathsf{CreateUser}$: On input an identity $\mathsf{ID} \in \{0,1\}^*$, if ID has not been created, the oracle runs $psk_{\mathsf{ID}} \leftarrow \mathsf{PartialKeyGen}(msk, \mathsf{ID})$ and $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}}) \leftarrow \mathsf{UserKeyGen}(mpk, \mathsf{ID})$. It then stores $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ into a list List[1]. In both cases, upk_{ID} is returned.
– $\mathsf{RevealPartialKey}$: On input an identity ID, the oracle searches List for an entry corresponding to ID. If it is not found, $\perp$ is returned; otherwise, the corresponding psk_{ID} is returned.
– $\mathsf{RevealSecretKey}$: On input an identity ID, the oracle searches List for the entry of ID. If it is not found, $\perp$ is returned; otherwise, the corresponding usk_{ID} is returned.
– $\mathsf{ReplaceKey}$: On input an identity ID along with a user public/private key pair (upk', usk'), the oracle searches List for the entry of ID. If it is not found, nothing will be carried out. If $usk' = \perp$, the oracle sets $usk' = usk_{\mathsf{ID}}$. Then, it updates $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ to $(\mathsf{ID}, psk_{\mathsf{ID}}, upk', usk')$.
– $\mathsf{Decryption}$: On input an identity ID and a ciphertext c, the oracle searches List for the entry of ID. If it is not found, $\perp$ is returned. Otherwise, it runs $m \leftarrow \mathsf{Dec}(psk_{\mathsf{ID}}, usk_{\mathsf{ID}}, c)$ and returns m. Note that the original upk_{ID} (which is returned by $\mathsf{CreateUser}$ oracle) may have been replaced by the adversary.

Remark: In the original adversarial model of certificateless encryption [1,10], it is required that the $\mathsf{Decryption}$ oracle should provide correct decryptions even after the user public key has been replaced by the adversary while the corresponding user secret key is not known. We believe that the model is hardly realistic. Hence in this paper, we only require the $\mathsf{Decryption}$ oracle to perform the decryption task by using the current user keys. This also captures the case that the user public key is replaced by the adversary, but the user secret key remains the same. It is possible that the message m recovered from the ciphertext by using the current usk_{ID} is $\perp$.

Game-I : Let $\mathcal{C}_1$ be the challenger/simulator and $k \in \mathbb{N}$ be a security parameter.

1. $\mathcal{C}_1$ runs $(mpk, msk) \leftarrow \mathsf{MasterKeyGen}(1^k)$, and then invokes $\mathcal{A}_1$ on input 1^k and mpk.
2. In this game, $\mathcal{A}_1$ can issue $\mathsf{CreateUser}$, $\mathsf{RevealPartialKey}$, $\mathsf{RevealSecretKey}$, $\mathsf{ReplaceKey}$ and $\mathsf{Decryption}$ queries.
3. $\mathcal{A}_1$ submits two equal-length messages (m_0, m_1) along with a target identity ID^*.
4. $\mathcal{C}_1$ selects a random bit $b \in \{0,1\}$, computes a challenge ciphertext c^* by running $c^* \leftarrow \mathsf{Enc}(mpk, \mathsf{ID}^*, upk_{\mathsf{ID}^*}, m_b)$, and returns c^* to $\mathcal{A}_1$, where upk_{ID^*} is the user public key currently in List for ID^*.
5. $\mathcal{A}_1$ continues to issue queries as in step 2. Finally it outputs a bit b'.

$\mathcal{A}_1$ is said to win the game if $b' = b$, and (1) $\mathcal{A}_1$ did not query $\mathsf{RevealPartialKey}$ on ID^*, (2) $\mathcal{A}_1$ did not query $\mathsf{Decryption}$ on (ID^*, c^*). We denote by $Pr[\mathcal{A}_1\ \mathsf{Succ}]$ the probability that $\mathcal{A}_1$ wins the game, and define the *advantage* of $\mathcal{A}_1$ in **Game-I** to be $\mathsf{Adv}_{\mathcal{A}_1} = \left|Pr[\mathcal{A}_1\ \mathsf{Succ}] - \frac{1}{2}\right|$.

[1] Note that the list List is shared among all these five oralces.

Game-II : Let $\mathcal{C}_2$ be the simulator/challenger and $k \in \mathbb{N}$ be a security parameter.

1. $\mathcal{C}_2$ runs $\mathcal{A}_2$ on input 1^k, which returns a master public key mpk to $\mathcal{C}_2$. Note that $\mathcal{A}_2$ cannot make any oracle query at this stage[2].
2. $\mathcal{A}_2$ can start querying oracles $\mathsf{CreateUser}$, $\mathsf{RevealSecretKey}$, $\mathsf{ReplaceKey}$ and $\mathsf{Decryption}$. Note that oracle $\mathsf{RevealPartialKey}$ is no longer needed as $\mathcal{A}_2$ knows the master private key msk, and $\mathcal{A}_2$ may not follow the specification of $\mathsf{MasterKeyGen}$ to generate (msk, mpk). One thing to notice is that when $\mathcal{A}_2$ issues a query to $\mathsf{CreateUser}$ oracle, it has to additionally provide the user partial key psk_{ID}.
3. $\mathcal{A}_2$ submits two equal-length messages (m_0, m_1) along with a target identity ID^*. $\mathcal{C}_2$ randomly selects a bit b, and computes the challenge ciphertext c^* by running $c^* \leftarrow \mathsf{Enc}(mpk, \mathsf{ID}^*, upk_{\mathsf{ID}^*}, m_b)$. It returns c^* to $\mathcal{A}_2$.
4. $\mathcal{A}_2$ continues to issue queries as in step 2. Finally, it outputs a bit b'.

$\mathcal{A}_2$ is said to win the game if $b' = b$, and (1) $\mathcal{A}_2$ did not query $\mathsf{RevealSecretKey}$ on ID^*, (2) $\mathcal{A}_2$ did not query $\mathsf{ReplaceKey}$ on $(\mathsf{ID}^*, \cdot, \cdot)$ to replace upk_{ID^*}, (3) $\mathcal{A}_2$ did not query $\mathsf{Decryption}$ on (ID^*, c^*). Similarly, we denote by $Pr[\mathcal{A}_2\ \mathsf{Succ}]$ the probability that $\mathcal{A}_2$ wins the game, and define the *advantage* of $\mathcal{A}_2$ in **Game-II** to be $\mathsf{Adv}_{\mathcal{A}_2} = \left|Pr[\mathcal{A}_2\ \mathsf{Succ}] - \frac{1}{2}\right|$.

Definition 1. *A certificateless encryption scheme* CLE *is said to be* Type-I secure *(resp.* Type-II secure*) if there is no probabilistic polynomial-time adversary* $\mathcal{A}_1$ *(resp.* $\mathcal{A}_2$*) which wins* **Game-I** *(resp.* **Game-II***) with non-negligible advantage.* CLE *is said to be* $\mathsf{IND\text{-}CCA2}$ secure *if it is both* Type-I secure *and* Type-II secure*.*

3 Malicious-but-Passive KGC Attack

We describe two new malicious-but-passive KGC attacking techniques (under **Game-II**) to compromise schemes that follow the key generation procedure described in [11]. The techniques are different from that in [2], which is used to compromise schemes based on another type of key generation procedures [1,7,9].

We briefly describe the certificateless encryption scheme proposed in [11] to a certain extent that our attacking technique can be understood without referring to the complete description of the original scheme. In the $\mathsf{MasterKeyGen}$ of [11], the KGC first generates a pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ such that each group has order p. Then, a generator g of $\mathbb{G}_1$ is selected. This is followed by the selection of a set of random elements in $\mathbb{G}_1$. The parameters we are going to use in the attack below are $g_2, g'_1, h_1, u' \in_R \mathbb{G}_1$ and $\hat{U} = \{\hat{u}_i\}$ where $\hat{u}_i \in_R \mathbb{G}_1$, for $i = 1, \cdots, n$, and some $n \in \mathbb{Z}$. We skip the description of the remaining steps of $\mathsf{MasterKeyGen}$ and also the entire $\mathsf{PartialKeyGen}$. In $\mathsf{UserKeyGen}$, the user public key upk_{ID^*} for a user with identity ID^* is denoted by $(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}) \in \mathbb{G}_1^2$. We do not need to look into how these two elements are generated. Our first attack is described as follows.

(Attack 1) The malicious-but-passive KGC (that is $\mathcal{A}_2$ in **Game-II**) arbitrarily selects a target identity ID^*. It computes $\mathfrak{u} = H_u(\mathsf{ID}^*)$, where $H_u : \{0,1\}^* \rightarrow \{0,1\}^n$ is a collision-resistant

[2] One exception is that if a scheme is analyzed under the random oracle model, $\mathcal{A}_2$ can query the random oracle. In this paper, we do not consider this exception as the security of our scheme proposed in the subsequent section (Sec. 4) will be shown in the standard model.

hash function pre-defined for this scheme. Let $\mathfrak{u}[i]$ be the i-th bit of $\mathfrak{u}$. Define $\mathcal{U} \subseteq \{1, \cdots, n\}$ to be the set of indices such that $\mathfrak{u}[i] = 1$. The KGC then randomly selects $s \in_R \mathbb{Z}_p$, and sets $g_2 = (U^*)^s$, where $U^* = u' \prod_{i \in \mathcal{U}} \hat{u}_i$. Other parameters in the master public/private key pair are generated normally by the KGC. In the challenge phase of **Game-II**, the KGC submits two distinct equal-length messages, (m_0, m_1), and ID^* as the target identity. The challenger $\mathcal{C}_2$ randomly selects a bit b, computes the challenge ciphertext $C^* = (\hat{C}^*, \mathsf{com}^*, \mathsf{tag}^*)$ according to the encryption algorithm of [11]. Let the challenge ciphertext $\hat{C}^* = (C_1^*, C_2^*, C_3^*, C_4^*)$. According to the specification of the encryption algorithm in [11], we have

$$C_1^* = e(\mathsf{pk}^{(2)}, g_2)^t M, \qquad C_2^* = g^t, \qquad C_3^* = (U^*)^t, \qquad C_4^* = ((g_1')^{\mathsf{com}^*} h_1)^t$$

where $t \in_R \mathbb{Z}_p$ and $M = m_b \| \mathsf{dec}$ for some binary string dec. The KGC can get the plaintext of $\hat{C}^*$ by computing the following

$$\begin{aligned}\frac{C_1^*}{e(\mathsf{pk}^{(2)}, (C_3^*)^s)} &= \frac{e(\mathsf{pk}^{(2)}, g_2^t)M}{e(\mathsf{pk}^{(2)}, (U^*)^{st})} \\ &= \frac{e(\mathsf{pk}^{(2)}, g_2^t)M}{e(\mathsf{pk}^{(2)}, (g_2)^t)} = M = m_b \| \mathsf{dec}\end{aligned}$$

By comparing m_b with m_0 and m_1, the KGC can easily find out the message corresponding to C^*. This attack causes the same extent of damage as that described in [2] against [1,7,9]. Both attacks require the KGC to *pre-select* a target identity. The KGC is not able to compromise two users in the system under the **Game-II**. Specific to this certificateless encryption scheme described in [11], there is a more powerful malicious-but-passive KGC attacking technique which allows the KGC to decrypt any ciphertext in the system regardless which user is the corresponding decryptor. Therefore, the KGC does not need to pre-select a target identity.

(Attack 2) Note that the message M is 'masked' in C_1^* by $e(\mathsf{pk}^{(2)}, g_2)^t$. Instead of selecting g_2 randomly from $\mathbb{G}_1$, suppose the malicious-but-passive KGC randomly picks $\beta \in \mathbb{Z}_p$ and sets $g_2 = g^\beta$. We can see that the KGC can remove the mask of any ciphertext by simply computing the mask value as $e(\mathsf{pk}^{(2)}, C_2^*)^\beta$.

As a further remark, the certificateless signature scheme described in [11] also suffers from a malicious-but-passive KGC attacking technique which is very similar to the first one described above. Precisely, the KGC can forge any signature of a pre-selected target identity ID^*, as it shares the same key generation procedure. We skip the details here and emphasize that the attacks above do not refute the security claims made in [11] as their security model does not consider/capture the malicious-but-passive KGC attacks.

A Design Philosophy: To design a certificateless scheme, no matter it is an encryption scheme or a signature scheme, for security against malicious-but-passive KGC attacks, we have a great concern on its security if the scheme requires the user to make use of the parameters generated by the KGC when generating its own user public/private key pair (via $\mathsf{UserKeyGen}$). The attacks above illustrate how subtle an attack can be if the malicious-but-passive KGC has certain control on the parameters used for user key pair generation. In the first attack above, the KGC can simply modify the generation of g_2 which is one of the many parameters generated/controlled by the KGC. The user has no way to tell if g_2 is generated accordingly or maliciously. In fact, there are many other ways for the KGC to break the scheme in [11], as there are many parameters generated by or under control of the KGC. One can easily come up with more attacks against the one in [11] in addition to the two attacks described above. In addition, even if the KGC only generates a set of group parameters, for example, a bilinear pairing operation and its associated groups, we cannot guarantee that any malicious-but-passive KGC attack cannot be launched. The reason is that the groups generated and the bilinear operation chosen by the malicious KGC may not be 'generic' [13]. There may exist some trapdoor such that only the one who generates the group parameters, in our case, it is the malicious KGC, can perform some operations efficiently. Therefore, those schemes which require the user to use group parameters generated by the KGC may either be broken by some newly discovered malicious-but-passive KGC attacking techniques, or have their security left unproven.

In our generic scheme proposed below, we design our scheme such that the user partial key and user secret key are generated and used totally independently, while retaining high efficiency.

4 Our Scheme

In this section, we propose a generic certificateless encryption scheme CLE and show that it is secure under the adversarial model defined in Sec. 2. In particular, this generic scheme is the first one proven secure against the malicious-but-passive KGC attacks in the standard model.

Let $\mathsf{IBE} = (\mathsf{KG}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ be an $\mathsf{IND\text{-}ID\text{-}CCA2}$ secure identity-based encryption scheme, $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ an $\mathsf{IND\text{-}CCA2}$ secure public key encryption scheme, and $\mathsf{S} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Vrfy})$ a strong one-time signature scheme. For formal definitions of IBE, PKE and S, please refer to Appendix A. In the following, we propose a generic certificateless encryption scheme CLE, which is based on these three primitives.

- $\mathsf{MasterKeyGen}$: The KGC runs $(mpk, msk) \leftarrow \mathsf{IBE.KG}(1^k)$, publishes mpk and keeps msk secret.
- $\mathsf{PartialKeyGen}$: On input an identity ID, the KGC runs $psk_{\mathsf{ID}} \leftarrow \mathsf{IBE.Extract}(msk, \mathsf{ID})$ and returns psk_{ID}.
- $\mathsf{UserKeyGen}$: The user (with identity ID) runs $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}}) \leftarrow \mathsf{PKE.KG}(1^k)$, publishes $(\mathsf{ID}, upk_{\mathsf{ID}})$ and stores usk_{ID}.
- Enc: To encrypt a message m for user ID, the encryptor computes the following and returns c:

$$\begin{aligned}(vk, sk) &\leftarrow \mathsf{S.KG}(1^k)\\ c_1 &\leftarrow \mathsf{IBE.Enc}(mpk, \mathsf{ID}, m\|vk)\\ c_2 &\leftarrow \mathsf{PKE.Enc}(upk_{\mathsf{ID}}, c_1)\\ \sigma &\leftarrow \mathsf{S.Sign}(sk, c_2)\\ c &\leftarrow (c_2, \sigma, vk)\end{aligned}$$

- Dec: On input an identity ID and a ciphertext $c = (c_2, \sigma, vk)$, if $0 \leftarrow \mathsf{S.Vrfy}(vk, \sigma, c_2)$, $\perp$ is returned. Otherwise, the decryptor computes the following:

$$c_1 \leftarrow \mathsf{PK.Dec}(usk_{\mathsf{ID}}, c_2)$$

$$m \| vk' \leftarrow \mathsf{IBE.Dec}(psk_{\mathsf{ID}}, \mathsf{ID}, c_1)$$

If $vk' \neq vk$, the decryptor outputs $\perp$; otherwise, it outputs m.

Theorem 1. *The certificateless encryption scheme* CLE *is* Type-I secure*, provided that the underlying identity-based encryption scheme* IBE *is* IND-ID-CCA2 *secure, and the one-time signature scheme* S *is* strongly unforgeable*.*

Proof. Suppose that $\mathcal{A}_1$ is a PPT adversary that tries to break the Type-I security of CLE. Let the challenge ciphertext that $\mathcal{A}_1$ receives be $c^* = (c_2^*, \sigma^*, vk^*)$. We say a ciphertext $c = (c_2, \sigma, vk)$ is valid with respect to identity ID if the decryption oracle would not output $\perp$ on input (ID, c). We denote by Forge_1 the event that vk^* appears in a decryption query $(\mathsf{ID}, c = (c_2, \sigma, vk^*))$ issued by $\mathcal{A}_1$ such that c is valid with respect to ID and $(c_2, \sigma) \neq (c_2^*, \sigma^*)$. Then we have:

Lemma 1. $Pr[\mathsf{Forge}_1]$ *is* negligible *in the security parameter k.*

Lemma 2. $\left|Pr[\mathcal{A}_1 \ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_1}] + \frac{1}{2}Pr[\mathsf{Forge}_1] - \frac{1}{2}\right|$ *is* negligible *(in k).*

Please refer to Appendix B for the proofs. Therefore, by these two lemmas, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}_1} &= \left|Pr[\mathcal{A}_1 \ \mathsf{Succ}] - \frac{1}{2}\right| \\
&= \left|Pr[\mathcal{A}_1 \ \mathsf{Succ} \wedge \mathsf{Forge}_1] - \frac{1}{2}Pr[\mathsf{Forge}_1] + \frac{1}{2}Pr[\mathsf{Forge}_1] + Pr[\mathcal{A}_1 \ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_1}] - \frac{1}{2}\right| \\
&\leq \left|Pr[\mathcal{A}_1 \ \mathsf{Succ} \wedge \mathsf{Forge}_1] - \frac{1}{2}Pr[\mathsf{Forge}_1]\right| + \left|Pr[\mathcal{A}_1 \ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_1}] + Pr[\mathsf{Forge}_1] \cdot \frac{1}{2} - \frac{1}{2}\right| \\
&\leq \frac{1}{2}Pr[\mathsf{Forge}_1] + \left|Pr[\mathcal{A}_1 \ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_1}] + \frac{1}{2}Pr[\mathsf{Forge}_1] - \frac{1}{2}\right|
\end{aligned}$$

which is also *negligible* in k. This completes the proof of Theorem 1. □

Theorem 2. *The certificateless encryption scheme* CLE *is* Type-II secure *if the underlying public key encryption scheme* PKE *is* IND-CCA2 *secure and the one-time signature scheme* S *is* strongly unforgeable*.*

Proof. Suppose that $\mathcal{A}_2$ is a PPT adversary that tries to break the Type-II security of CLE. Let $c^* = (c_2^*, \sigma^*, vk^*)$ be the challenge ciphertext that $\mathcal{A}_2$ receives, and let Forge_2 be the event that vk^* appears in a decryption query $(\mathsf{ID}, c = (c_2, \sigma, vk))$ issued by $\mathcal{A}_2$ (i.e., $vk^* = vk$) such that c is valid with respect to ID and $(c_2, \sigma) \neq (c_2^*, \sigma^*)$. We have the following two lemmas:

Lemma 3. $Pr[\mathsf{Forge}_2]$ *is* negligible *in k.*

This proof is similar to that of Lemma 1 and is omitted here.

Lemma 4. $\left|Pr[\mathcal{A}_2 \ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}Pr[\mathsf{Forge}_2] - \frac{1}{2}\right|$ *is* negligible *in k.*

Proof. We construct a PPT algorithm $\mathcal{C}_2$ to break the IND-CCA2 security of PKE by using $\mathcal{A}_2$ as a subroutine. Given the challenge public key pk and a decryption oracle $\bar{\mathcal{O}}_{\mathsf{D}}$, $\mathcal{C}_2$ runs $\mathcal{A}_2$ on input 1^k, which returns a master public key mpk. Assume that $\mathcal{A}_2$ issues at most q distinct CreateUser queries. Then, $\mathcal{C}_2$ randomly selects $i \in \{1, 2, \cdots, q\}$, runs $(vk^*, sk^*) \leftarrow \mathsf{S.KG}(1^k)$, stores (vk^*, sk^*) which will be used in the generation of the challenge ciphertext of $\mathcal{A}_2$, and simulates all the oracles for $\mathcal{A}_2$ as follows:

– $\mathsf{CreateUser}$: On input an identity ID and the user partial key psk_{ID}, we assume that this is the j-th distinct $\mathsf{CreateUser}$ query. If $j \neq i$, $\mathcal{C}_2$ runs $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}}) \leftarrow \mathsf{PKE.KG}(1^k)$, stores $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}}, 0)$ into a list List, and returns upk_{ID}. If $j = i$, $\mathcal{C}_2$ simply stores $(\mathsf{ID}, psk_{\mathsf{ID}}, pk, \perp, 0)$ into List and returns pk.
– $\mathsf{RevealSecretKey}$: On input an identity ID, $\mathcal{C}_2$ searches List for the entry of ID. If there is no such an entry, $\mathcal{C}_2$ returns $\perp$. If $upk_{\mathsf{ID}} = pk$, $\mathcal{C}_2$ outputs a random bit and aborts. Otherwise, it returns the corresponding usk_{ID}.
– $\mathsf{ReplaceKey}$: On input $(\mathsf{ID}, upk', usk')$, $\mathcal{C}_2$ searches List for the entry of ID. If there is no such an entry, $\mathcal{C}_2$ does nothing. If $upk_{\mathsf{ID}} = pk$, $\mathcal{C}_2$ aborts and outputs a random bit. If $usk' = \perp$, $\mathcal{C}_2$ sets $usk' = usk_{\mathsf{ID}}$. Then, it updates $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ to be $(\mathsf{ID}, psk_{\mathsf{ID}}, upk', usk')$.
– $\mathsf{Decryption}$: On input $(\mathsf{ID}, c = (c_2, \sigma, vk))$, $\mathcal{C}_2$ searches List for the entry of ID. If it is not found, $\mathcal{C}_2$ returns $\perp$. If ID is not the i-th distinct query made by $\mathcal{A}_2$ to $\mathsf{CreateUser}$, $\mathcal{C}_2$ simulates the $\mathsf{Decryption}$ oracle using its knowledge of psk_{ID} and usk_{ID}. Otherwise ($upk_{\mathsf{ID}} = pk$), after the validity check of σ on c_2 with respect to vk, $\mathcal{C}_2$ makes a $\mathsf{Decryption}$ query to oracle $\bar{\mathcal{O}}_{\mathsf{D}}$ which returns with c_1. Then it completes the rest using its knowledge of psk_{ID}. Note that if event Forge_2 occurs during the simulation of $\mathsf{Decryption}$ oracle, namely, σ is a valid signature on c_2 with respect to vk and $vk = vk^*$, $\mathcal{C}_2$ outputs a random bit and aborts.

At some point $\mathcal{A}_2$ submits two equal-length messages (m_0, m_1) along with a target identity ID^*. If $usk_{\mathsf{ID}^*} \neq pk$, $\mathcal{C}_2$ outputs a random bit and aborts. Otherwise, it computes $(c_1)_0 \leftarrow \mathsf{IBE.Enc}(mpk, \mathsf{ID}^*, m_0 \| vk^*)$ and $(c_1)_1 \leftarrow \mathsf{IBE.Enc}(mpk, \mathsf{ID}^*, m_1 \| vk^*)$, submits $(c_1)_0$ and $(c_1)_1$ to its challenger, and is returned c_2^* which is a ciphertext of $(c_1)_b$, where $b \in \{0,1\}$. It then computes $\sigma^* \leftarrow \mathsf{S.Sign}(sk^*, c_2^*)$ and returns $c^* = (c_2^*, \sigma^*, vk^*)$ to $\mathcal{A}_2$ as the challenge ciphertext. $\mathcal{C}_2$ continues to simulate all the oracles for $\mathcal{A}_2$ as above. Finally it outputs the bit b' output by $\mathcal{A}_2$.

Obviously, the probability that $\mathcal{C}_2$ doesn't abort in simulating $\mathsf{ReplaceKey}$ and $\mathsf{RevealSecretKey}$ oracles is at least $1/q$. If $\mathcal{C}_2$ doesn't abort in simulating the two oracles, the probability that it wins its own game is $\frac{1}{2}Pr[\mathsf{Forge}_2] + Pr[\mathcal{A}_2 \text{ Succ} \wedge \overline{\mathsf{Forge}_2}]$. Thus, we get that the advantage that $\mathcal{C}_2$ wins its game is

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{C}_2} &= \left| Pr[\mathcal{C}_2 \text{ Succ}] - \frac{1}{2} \right| \\
&\geq \left| \frac{1}{2} \cdot \left(1 - \frac{1}{q}\right) + \left(Pr[\mathcal{A}_2 \text{ Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}Pr[\mathsf{Forge}_2]\right) \cdot \frac{1}{q} - \frac{1}{2} \right| \\
&= \left| Pr[\mathcal{A}_2 \text{ Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}Pr[\mathsf{Forge}_2] - \frac{1}{2} \right| \cdot \frac{1}{q}
\end{aligned}$$

Guaranteed by the $\mathsf{IND\text{-}CCA2}$ security of PKE, we have that $\mathsf{Adv}_{\mathcal{C}_2}$ is negligible in k. Thus, $\left|Pr[\mathcal{A}_2 \text{ Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}Pr[\mathsf{Forge}_2] - \frac{1}{2}\right|$ is negligible as well since $1/q$ is polynomial in k. □

By the two lemmas above, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}_2} &= \left| Pr[\mathcal{A}_2 \text{ Succ}] - \frac{1}{2} \right| \\
&\leq \left| Pr[\mathcal{A}_2 \text{ Succ} \wedge \mathsf{Forge}_2] - \frac{1}{2}Pr[\mathsf{Forge}_2] \right| + \left| Pr[\mathcal{A}_2 \text{ Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}Pr[\mathsf{Forge}_2] - \frac{1}{2} \right| \\
&\leq \frac{1}{2}Pr[\mathsf{Forge}_2] + \left| Pr[\mathcal{A}_2 \text{ Succ} \wedge \overline{\mathsf{Forge}_2}] + \frac{1}{2}Pr[\mathsf{Forge}_2] - \frac{1}{2} \right|
\end{aligned}$$

which is negligible as well. This completes the proof of Theorem 2. □

The following corollary is obtained immediately from Theorem 1 and Theorem 2.

Corollary 1. *The certificateless encryption scheme* CLE *described above is* $\mathsf{IND\text{-}CCA2}$ secure.

4.1 Discussion

In practice, for high performance in the encryption process, we usually use the *hybrid encryption* method which combines a public key encryption and a symmetric encryption to encrypt the message instead of using the public key encryption directly to encrypt bulk data. To apply this to our generic scheme CLE, we first generate a random symmetric key key for a secure symmetric encryption scheme SE, then use SE to encrypt $m\|vk$ under key, and finally encrypt key using CLE. The decryption algorithm is modified accordingly. One of the key advantages of applying the hybrid encryption onto CLE (besides efficiency) is that the message space will not be restricted by the size of the verification key vk. We elaborate more on the size of vk below.

The (strong) one-time signature in our generic scheme CLE provides a certain assurance on that the encryptor did encrypt the message itself (or the '*well-formedness*' of a ciphertext). Since most of the one-time signature schemes in the literature follow the 'one-way function' paradigm [5], the verification key and the signature are both of large size. An immediate consequence is that the message encrypted (i.e., $m\|vk$) and the resulting ciphertext (i.e., $c = (c_2, \sigma, vk)$) of our scheme also suffer from the large size. We describe two methods which can significantly reduce the size in the actual implementation of CLE.

One simple solution is to replace the strong one-time signature scheme with a conventional signature scheme which is strongly unforgeable under adaptive chosen message attack. This does not weaken the security of CLE because any strongly unforgeable signature scheme is also a strong one-time signature scheme. A good candidate is the short signature proposed by Boneh and Boyen [4] as the verification key and the signature of the scheme in [4] are both small in size. The tradeoff is that the resulting scheme requires more computation than that of a strong one-time signature scheme.

Another solution is to first map the one-time verification key into a much shorter string using a *target collision-resistant* hash function H[3], and then encrypt $m\|\mathsf{H}(vk)$ rather than $m\|vk$. The decryption algorithm is changed accordingly. That is, the user checks if the second part of the plaintext (decrypted from the ciphertext) is the hash of vk. To analyze the security of this modified scheme, we need to show that it is negligible for the adversary to issue a $\mathsf{Decryption}$ query on input ID and $c = (c_2, \sigma, vk)$ such that $vk \neq vk^*$ but $\mathsf{H}(vk) = \mathsf{H}(vk^*)$, where vk^* is the verification key in the challenge ciphertext. It is guaranteed by the collision-resistance property of H. This method reduces the impact on the message size by the size of the verification key while adding only slightly on the computation cost. However, the ciphertext is not much shorter than that of the original scheme.

Yet, we observe that the verification key vk could be removed from the ciphertext without any influence on the security of the resulting encryption scheme. In this way, there is no need to use the asymmetric version of one-time signature. Instead, we could use a (strong) one-time message authentication code (MAC) [14,15,3,8]. The new scheme CLE' enjoys better efficiency and much shorter ciphertext than CLE. The definition of strong one-time MAC and the new construction will be described in the next part.

[3] Please refer to Appendix A for the definition of target collision-resistant hash functions.

4.2 A More Efficient Scheme

A *message authentication code* MAC is a pair of polynomial-time algorithms $(\mathsf{Mac}, \mathsf{Vrfy})$ such that:

- Mac takes as input a key $mk \in \mathcal{K}_M$ (the key space of MAC) and a message m, and outputs a tag σ, where k is the security parameter and m is in some implicit message space. We denote this by $\sigma \leftarrow \mathsf{Mac}_{mk}(m)$. Without loss of generality, we can also simply assume here that the key space $\mathcal{K}_M$ of MAC is $\{0,1\}^k$.
- Vrfy takes as input a key mk, a message m and a tag σ and outputs a bit $b \in \{0,1\}$ where the 1-value of b indicates 'accept' and 0-value indicates 'reject'. We denote this by $b \leftarrow \mathsf{Vrfy}_{mk}(m, \sigma)$.

For the security of a message authentication code, we consider the following game:

1. A random key $mk \in \{0,1\}^k$ is chosen;
2. $\mathcal{A}_M(1^k)$ is allowed to submit a message m and is then returned $\sigma \leftarrow \mathsf{Mac}_{mk}(m)$.
3. Finally, $\mathcal{A}_M$ outputs (m^*, σ^*).

We say that $\mathcal{A}_M$ wins if $1 \leftarrow \mathsf{Vrfy}_{mk}(m^*, \sigma^*)$ and $(m^*, \sigma^*) \neq (m, \sigma)$ (assuming that $\mathcal{A}_M$ did issue a query for a tag on input m in step 2).

Definition 2. *A message authentication code* MAC *is said to be* strong one-time *secure, if for any PPT adversary* $\mathcal{A}_M$*, the probability that* $\mathcal{A}_M$ *wins the the above game is* negligible *in* k*.*

Now we begin to describe the more efficient scheme. Let IBE and PKE be as in Sec. 4, and let $\mathsf{MAC} = (\mathsf{Mac}, \mathsf{Vrfy})$ be a *strong one-time* message authentication code and $\mathcal{K}_\mathsf{M}$ be the key space of MAC. We construct a new certificateless encryption scheme CLE' from IBE, PKE and MAC, as below:

- $\mathsf{MasterKeyGen}$: The KGC runs $(mpk, msk) \leftarrow \mathsf{IBE.KG}(1^k)$, publishes mpk and keeps msk secret.
- $\mathsf{PartialKeyGen}$: On input an identity ID, the KGC runs $psk_\mathsf{ID} \leftarrow \mathsf{IBE.Extract}(msk, \mathsf{ID})$ and returns psk_ID.
- $\mathsf{UserKeyGen}$: The user (with identity ID) runs $(upk_\mathsf{ID}, usk_\mathsf{ID}) \leftarrow \mathsf{PKE.KG}(1^k)$, publishes $(\mathsf{ID}, upk_\mathsf{ID})$ and stores usk_ID.
- Enc: To encrypt a message m for user ID, the encryptor computes the following and returns c:

$$\begin{aligned}
mk &\leftarrow \mathcal{K}_M \\
c_1 &\leftarrow \mathsf{IBE.Enc}(mpk, \mathsf{ID}, m \| mk) \\
c_2 &\leftarrow \mathsf{PKE.Enc}(upk_\mathsf{ID}, c_1) \\
\sigma &\leftarrow \mathsf{MAC.Mac}_{mk}(c_2) \\
c &\leftarrow (c_2, \sigma)
\end{aligned}$$

– Dec: On input an identity ID and a ciphertext $c = (c_2, \sigma)$, the decryptor computes the following:

$$\begin{aligned} c_1 &\leftarrow \mathsf{PK.Dec}(usk_{\mathsf{ID}}, c_2) \\ m\|mk &\leftarrow \mathsf{IBE.Dec}(psk_{\mathsf{ID}}, \mathsf{ID}, c_1) \\ b &\leftarrow \mathsf{MAC.Vrfy}_{mk}(c_2) \end{aligned}$$

If $b = 0$, the decryptor outputs $\perp$; otherwise, it outputs m.

Let (ID^*, c^*) be the challenge identity-ciphertext pair of the adversary, where $c^* = (c_2^*, \sigma^*)$, and let c_1^* be the plaintext of c_2^* under PKE. Consider a query $(\mathsf{ID}, c = (c_2, \sigma))$ submitted by the adversary to the $\mathsf{Decryption}$ oracle, and let c_1 be the plaintext of c_2 under PKE. Obviously we have that $(\mathsf{ID}, c) \neq (\mathsf{ID}^*, c^*)$. We focus on the case in which $\mathsf{ID} = \mathsf{ID}^*$. Now the event Forge_1 (resp. Forge_2) is defined as the event that the key mk extracted from c_2 is the same as that extracted from c_2^* and $1 \leftarrow \mathsf{MAC.Vrfy}_{mk}(\sigma)$ in **Game I** (resp. **Game II**). Similar to the proofs of Lemma 1 and 3, we can show that the probability that the event Forge_1 (resp. Forge_2) occurs is negligible in k. This is guaranteed by the *strong one-time security* of MAC. The other parts of the security proof remain the same except some minor modifications. One can easily come up with the security proof of CLE', for its high similarity with CLE.

Remark: As we can see from the above construction, the verification key is removed from the ciphertext and now is hidden in c_2, and the symmetric signing key is not generated by a (time-consuming) key generation algorithm any more but randomly selected. Therefore, the scheme CLE' enjoys *much shorter* ciphertext and *much higher* efficiency than CLE. However, unlike CLE, the main drawback of this new scheme is that the validity of the ciphertext cannot be verified *before* the decryption any longer. One has to decrypt the ciphertext, even if it is invalid.

Remark Again: To make the new scheme be of practical use, we may use CBC-MAC with 128-bit AES as the underlying block cipher again. Besides, there are many other good candidates for MAC, such as the strong one-time MACs with information-theoretic security in [14,15].

5 Conclusion

In this paper, we consider the security of certificateless encryption schemes in the presence of a *malicious-but-passive* KGC, and propose the *first generic* certificateless encryption scheme in *the standard model*. The scheme is efficient. We also describe how to apply short signature and hybrid encryption to implement an efficient instantiation of our generic scheme.

On the study of malicious-but-passive KGC attacks, we show that although the scheme in [11] does not have the same key generation procedure as that of [1], there are two new attacks which can compromise the Type-II security of their scheme. In particular, our second attack allows the KGC to decrypt any ciphertext without pre-selecting a target user.

However, it still remains an open problem on how to construct a certificateless encryption scheme secure with respect to an even stronger security model which combines the strongest one described in [1,10] and the malicious-but-passive KGC model described in [2] and also in this paper. The model in [1,10] requires the decryption oracle to provide correct decryption even after the user public key has been replaced.

References

1. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *Proc. ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer-Verlag, 2003.
2. M. H. Au, J. Chen, J. K. Liu, Y. Mu, D. S. Wong, and G. Yang. Malicious KGC attacks in certificateless cryptography. To appear in ACM ASIACCS 2007, also at `http://eprint.iacr.org/2006/255`.
3. M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, December 2000. Extended abstract in Proc. of Advances in Cryptology - Crypto 94, volume 839 of Lecture Notes in Computer Science, pages 341–358, Y. Desmedt ed, Springer-Verlag, 1994.
4. D. Boneh and X. Boyen. Short signatures without random oracles. In *Proc. EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, 2004.
5. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308, Apr. 1988.
6. 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 Australian Conference, ACISP 2006*, pages 235–246. Springer-Verlag, 2006. Lecture Notes in Computer Science vol. 4058.
7. 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*, volume 3810 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 2005.
8. ISO/IEC 9797. Data cryptographic techniques - data integrity mechanism using a cryptographic check function employing a block cipher algorithm, 1989.
9. X. Li, K. Chen, and L. Sun. Certificateless signature and proxy signature schemes from bilinear pairings. *Lithuanian Mathematical Journal*, 45(1):76–83, 2005.
10. 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*, volume 3958 of *Lecture Notes in Computer Science*, pages 474–490. Springer-Verlag, 2006.
11. J. K. Liu, M. H. Au, and W. Susilo. Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model. To appear in ACM ASIACCS 2007. Full paper: `http://eprint.iacr.org/2006/373`.
12. A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1984.
13. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proc. EUROCRYPT 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer-Verlag, 1997.
14. D. R. Stinson. Universal hashing and authentication codes. *Designs, Codes, and Cryptography*, 4(4):369–380, 1994.
15. M. N. Wegman and J. L. Carter. New hash functions and their user in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.
16. D. H. Yum and P. J. Lee. Generic construction of certificateless encryption. In *ICCSA 2004*, volume 3043 of *Lecture Notes in Computer Science*, pages 802–811. Springer-Verlag, 2004.
17. D. H. Yum and P. J. Lee. Generic construction of certificateless signature. In *Information Security and Privacy: 9th Australian Conference, ACISP 2004*, volume 3108 of *Lecture Notes in Computer Science*, pages 200–211. Springer-Verlag, 2004.

A Definition Revisions

A.1 Identity-Based Encryption

An identity-based encryption scheme consists of four (probabilistic) polynomial-time algorithms, KG, $\mathsf{Extract}$, Enc and Dec, which are for master key generation, user private key extraction, encrypting a message and decrypting a ciphertext, respectively. The standard and strongest notion of security of an ID-based encryption scheme is $\mathsf{IND\text{-}ID\text{-}CCA2}$, which is defined by the following game between a challenger and a PPT adversary $\mathcal{A}$:

1. The challenger runs $\mathsf{KG}(1^k)$ to obtain a master public/private key pair (mpk, msk), and sends mpk to $\mathcal{A}$.
2. $\mathcal{A}$ can issue two types of queries:
   - $\mathsf{Extraction}$: $\mathcal{A}$ submits an identity $\mathsf{ID} \in \{0,1\}^*$, and is returned the corresponding private key $sk_{\mathsf{ID}} \leftarrow \mathsf{Extract}(msk, \mathsf{ID})$.
   - $\mathsf{Decryption}$: $\mathcal{A}$ submits an identity ID and a ciphertext c of its choice, and is returned $m \leftarrow \mathsf{Dec}(sk_{\mathsf{ID}}, \mathsf{ID}, c)$.

   This process can be repeated for polynomially many times.
3. $\mathcal{A}$ submits two equal-length messages, (m_0, m_1) along with an identity ID^* to the challenger, which then selects a bit $b \in \{0,1\}$ at random, computes $c^* \leftarrow \mathsf{Enc}(mpk, \mathsf{ID}^*, m_b)$, and returns c^* to $\mathcal{A}$.
4. $\mathcal{A}$ continues to issue queries of its choice as in step 2, for polynomially many times.
5. Finally, $\mathcal{A}$ outputs a bit b'.

We say $\mathcal{A}$ wins the game if $b' = b$, and

1. $\mathcal{A}$ didn't issue an $\mathsf{Extraction}$ query on input ID^*;
2. $\mathcal{A}$ didn't issue a $\mathsf{Decryption}$ query on input (ID^*, c^*).

We denote by $Pr[\mathcal{A}\ \mathsf{Succ}]$ the probability that $\mathcal{A}$ wins the game, and define $\mathcal{A}$'s *advantage* in the game to be $\mathsf{Adv}_{\mathcal{A}} = \left|Pr[\mathcal{A}\ \mathsf{Succ}] - \frac{1}{2}\right|$.

A.2 Public Key Encryption

A public key encryption scheme consists of three (probabilistic) polynomial-time algorithms, KG, Enc and Dec, which are for key generation, encrypting a message and decrypting a ciphertext, respectively. The standard notion of security of a public key encryption scheme is $\mathsf{IND\text{-}CCA2}$, which is defined by the following game between a challenger and a PPT adversary $\mathcal{A}$:

1. The challenger runs $\mathsf{KG}(1^k)$ to obtain a public/private key pair (pk, sk), and sends pk to $\mathcal{A}$.
2. $\mathcal{A}$ requests to decrypt a ciphertext c of its choice, under the public key pk, and is returned $m \leftarrow \mathsf{Dec}(sk, c)$. This process can be repeated for polynomially many times.
3. $\mathcal{A}$ submits two equal-length messages, (m_0, m_1) to the challenger, which then selects a bit $b \in \{0,1\}$ at random, computes $c^* \leftarrow \mathsf{Enc}(pk, m_b)$, and returns c^* to $\mathcal{A}$.
4. $\mathcal{A}$ continues to request to decrypt ciphertexts of its choice.
5. Finally, $\mathcal{A}$ outputs a bit b'.

We say $\mathcal{A}$ wins the game if $b' = b$, and $\mathcal{A}$ didn't request to decrypt c^*. We denote by $Pr[\mathcal{A}\ \mathsf{Succ}]$ the probability that $\mathcal{A}$ wins the game, and define $\mathcal{A}$'s *advantage* in the game to be $\mathsf{Adv}_{\mathcal{A}} = \left|Pr[\mathcal{A}\ \mathsf{Succ}] - \frac{1}{2}\right|$.

A.3 Strong One-Time Signature

A signature scheme consists of three (probabilistic) polynomial-time algorithms, KG, Sign and Vrfy, which are for key generation, signing a message, and verifying a signature, respectively. The standard notion of the security of a signature scheme is *existential unforgeability under a* *chosen message attack.* A strong one-time signature is a signature scheme with strong existential unforgeability under a chosen one message attack, which is a security notion stronger than the standard one, and defined by the following game between a challenger and a PPT forger $\mathcal{F}$:

1. The challenger computes a one-time key pair (vk, sk) by running $(vk, sk) \leftarrow \mathsf{KG}(1^k)$, and then runs $\mathcal{F}$ on input vk.
2. $\mathcal{F}$ may issue a signing query on a single message chosen by itself, $m \in \{0,1\}^*$, under the verification key vk, and is returned a valid signature $\sigma \leftarrow \mathsf{Sign}(sk, m)$.
3. Finally, $\mathcal{F}$ outputs a message/signature pair, (m^*, σ^*), and wins the game if $(m^*, \sigma^*) \neq (m, \sigma)$ and $1 \leftarrow \mathsf{Vrfy}(vk, \sigma^*, m^*)$.

We denote by $Pr[\mathcal{F}\ \mathsf{Succ}]$ the probability that $\mathcal{F}$ wins the game.

A.4 Target Collision-Resistant Hash Functions

Let $\mathsf{H}_s : \{0,1\}^* \rightarrow \{0,1\}^\ell$ be a family of keyed hash functions for each k-bit key s, where $\ell = \ell(k) = \text{poly}(k)$ is a polynomial in k. The *target collision-resistance* is defined by the following game played by a PPT adversary $\mathcal{H}$:

1. A key s is randomly selected from $\{0,1\}^k$, and c^* is randomly from $\{0,1\}^*$.
2. On input s and c^*, $\mathcal{H}$ outputs c.

$\mathcal{H}$ wins the game if $c \neq c^*$ and $\mathsf{H}_s(c) = \mathsf{H}_s(c^*)$. We denote by $\mathcal{H}\ \mathsf{Succ}$ the event that $\mathcal{H}$ wins the game, and define the advantage of $\mathcal{H}$ in the game as $\mathsf{Adv}_{\mathcal{H}} = Pr[\mathcal{H}\ \mathsf{Succ}]$.

B Proofs of Lemma 1 and Lemma 2

B.1 Proof of Lemma 1

Proof. We can construct a PPT algorithm $\mathcal{F}$ to break the strong unforgeability of the underlying one-time signature scheme by using $\mathcal{A}_1$ as a subroutine. Given the target verification key vk^* and a one-time signing oracle $\mathcal{O}_{\mathsf{S}}$, $\mathcal{F}$ works as follows:

1. $\mathcal{F}$ randomly generates the master public/private key pair by running $(mpk, msk) \leftarrow \mathsf{IBE.KG}(1^k)$, and runs $\mathcal{A}_1$ on input mpk.
2. $\mathcal{F}$ simulates oracles RevealSecretKey, RevealPartialKey, ReplaceKey and Decryption for $\mathcal{A}_1$, by its knowledge of msk. When to answer a Decryption query $(\mathsf{ID}, c = (c_2, \sigma, vk))$, $\mathcal{F}$ first checks if the input ciphertext c is valid with regard to ID and $vk = vk^*$. If so, $\mathcal{F}$ outputs (c_2, σ) and halts; otherwise, it then behaves as the real decryptor.
3. At some point, $\mathcal{A}_1$ submits two equal-length messages (m_0, m_1) along with a target identity ID^*. $\mathcal{F}$ randomly chooses a bit b, runs $c_1^* \leftarrow \mathsf{IBE.Enc}(mpk, \mathsf{ID}^*, m_b \| vk^*)$ and $c_2^* \leftarrow \mathsf{PKE.Enc}(upk_{\mathsf{ID}^*}, c_1^*)$, and then issues a signing query to $\mathcal{O}_{\mathsf{S}}$ to get a one-time signature σ^* on c_2^*. $\mathcal{F}$ returns (c_2^*, σ^*, vk^*) to $\mathcal{A}_1$.
4. $\mathcal{A}_1$ continues to issue queries. Again, if a Decryption query $(\mathsf{ID}, c = (c_2, \sigma, vk))$ is valid with respect to ID and $vk = vk^*$, $\mathcal{F}$ outputs (c_2, σ) as its forgery for S.
5. If event Forge_1 doesn't occur before $\mathcal{A}_1$ produces its final output b', $\mathcal{F}$ aborts and fails.

It is readily to see that $\mathcal{F}$ perfectly simulates all the oracles for $\mathcal{A}_1$. So event Forge_1 occurs with the same probability as it would in the real environment. On the other hand, if Forge_1 occurs, $\mathcal{F}$ also succeeds in outputting a valid forgery for the one-time signature scheme S. Therefore, guaranteed by the strong unforgeability of S, we have that $Pr[\mathsf{Forge}_1]$ is negligible in k. □

B.2 Proof of Lemma 2

Proof. We now construct a PPT algorithm $\mathcal{C}_1$ to break the IND-ID-CCA2 security of IBE, using $\mathcal{A}_1$ as a subroutine. Given the master public key mpk, an Extract oracle $\mathcal{O}_{\mathsf{E}}$ and a Decryption oracle $\mathcal{O}_{\mathsf{D}}$, $\mathcal{C}_1$ runs $\mathcal{A}_1$ on input mpk. Since the generation of the verification key (in a ciphertext) is independent of the message, $\mathcal{C}_1$ can randomly generate the one-time key pair, which will be used in the generation of the challenge ciphertext c^*, at the onset of the simulation. Namely, $\mathcal{C}_1$ runs $(vk^*, sk^*) \leftarrow \mathsf{S.KG}(1^k)$, and stores (vk^*, sk^*). To answer $\mathcal{A}_1$'s oracle queries, $\mathcal{C}_1$ maintains a list $\mathsf{List} = \{(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})\}$ and works as follows:

- CreateUser: On input an identity ID, if ID was queried before, $\mathcal{C}_1$ returns the previous answer; otherwise, it runs $(upk_{\mathsf{ID}}, usk_{\mathsf{ID}}) \leftarrow \mathsf{PKE.KG}(1^k)$. It stores $(\mathsf{ID}, \perp, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ into List and returns upk_{ID}.
- RevealSecretKey: On input an ID, $\mathcal{C}_1$ looks up List for the entry of ID. If there is no such an entry, $\perp$ is returned; otherwise, the corresponding usk_{ID} is returned.
- RevealPartialKey: On input an identity ID, $\mathcal{C}_1$ searches List for the entry of ID. If there is no such an entry, $\perp$ is returned. If $psk_{\mathsf{ID}} = \perp$, it issues a query to oracle $\mathcal{O}_{\mathsf{E}}$ on input ID to get the partial private key psk_{ID} and stores it. $\mathcal{C}_1$ returns psk_{ID} to $\mathcal{A}_1$.
- ReplaceKey: On input $(\mathsf{ID}, upk', usk')$, $\mathcal{C}_1$ searches List for the entry of ID. If there is no such an entry, $\mathcal{C}_1$ does nothing else. If $usk' = \perp$, it sets $usk' = usk_{\mathsf{ID}}$. Then, it updates $(\mathsf{ID}, psk_{\mathsf{ID}}, upk_{\mathsf{ID}}, usk_{\mathsf{ID}})$ to $(\mathsf{ID}, psk_{\mathsf{ID}}, upk', usk')$.
- Decryption: On input an identity ID and a ciphertext $c = (c_2, \sigma, vk)$, $\mathcal{C}_1$ first searches List for the entry of ID. If there is no such an entry, $\perp$ is returned. Else, $\mathcal{C}_1$ works as a real Decryption oracle with the only exception that if event Forge_1 occurs, it simply outputs a random bit and aborts.

At some point, $\mathcal{A}_1$ submits to $\mathcal{C}_1$ two equal-length messages (m_0, m_1) along with a target identity ID^*. $\mathcal{C}_1$ submits $(m_0\|vk^*,\ m_1\|vk^*)$ and ID^* to its own challenger which would then choose a bit b, encrypt m_b with respect to identity ID^* and return the ciphertext c_1^*. $\mathcal{C}_1$ then computes $c_2^* \leftarrow \mathsf{PKE.Enc}(upk_{\mathsf{ID}^*}, c_1^*)$ and $\sigma^* \leftarrow \mathsf{S.Sign}(sk^*, c_2^*)$. It returns $c^* = (c_2^*, \sigma^*, vk^*)$ to $\mathcal{A}_1$ as the challenge ciphertext. $\mathcal{A}_1$ then continues to issue queries, and $\mathcal{C}_1$ answers as above. Note that, if $\mathcal{A}_1$ issues a Decryption query on input $(\mathsf{ID}^*, c = (c_2, \sigma, vk))$ where $c \neq c^*$, since $\mathcal{C}_1$ doesn't know the partial key psk_{ID^*}, it could not decrypt c by itself. Instead, after the decryption of c_2, $\mathcal{C}_1$ issues a Decryption query to its oracle $\mathcal{O}_{\mathsf{D}}$ on input c_1, and is returned $m\|vk'$. If $vk = vk'$, $\mathcal{C}_1$ returns m. Finally, $\mathcal{C}_1$ outputs the bit b' output by $\mathcal{A}_1$.

It's readily seen that $\mathcal{C}_1$ perfectly simulates all the oracles for $\mathcal{A}_1$. If $\mathcal{A}_1$ succeeds in guessing the bit b, then $\mathcal{C}_1$ also succeeds in outputting the correct bit. (If c_2^* is a ciphertext of $m_{b'}$ in CLE setting, then $c_1^* \leftarrow \mathsf{PKE.Dec}(usk_{\mathsf{ID}^*}, c_2^*)$ is a ciphertext of $m_{b'}$ in IBE setting.) Therefore, the advantage that $\mathcal{C}_1$ breaks the IND-IN-CCA2 security of IBE is

$$\mathsf{Adv}_{\mathcal{C}_1} = \left| Pr[\mathcal{C}_1\ \mathsf{Succ}] - \frac{1}{2} \right|$$

$$= \left| Pr[\mathcal{A}_1 \ \mathsf{Succ} \wedge \overline{\mathsf{Forge}_1}] + Pr[\mathsf{Forge}_1] \cdot \frac{1}{2} - \frac{1}{2} \right|$$

By the $\mathsf{IND\text{-}ID\text{-}CCA2}$ security of IBE, we have that $\mathsf{Adv}_{\mathcal{C}_1}$ is negligible. The lemma is proved. □