Efficient Certificateless KEM in the Standard Model *

Revised 8 March 2009

Georg Lippold, Colin Boyd, Juan González Nieto

Information Security Institute, Queensland University Of Technology,
GPO Box 2434, Brisbane QLD 4001, Australia
{g.lippold,c.boyd,j.gonzaleznieto}@qut.edu.au

Abstract. We give a direct construction of a certificateless key encapsulation mechanism (KEM) in the standard model that is more efficient than the generic constructions proposed before by Huang and Wong [10]. We use a direct construction from Kiltz and Galindo's KEM scheme [11] to obtain a certificateless KEM in the standard model; our construction is roughly twice as efficient as the generic construction. We also address the security flaw discovered by Selvi et al. [16].

1 Introduction

Certificateless encryption introduced by Al-Riyami and Paterson [1] is a variant of identity based encryption that limits the key escrow capabilities of the key generation centre (KGC), which are inherent in identity based encryption [3]. Dent [8] published a survey of more than twenty certificateless encryption schemes that focuses on the different security models and the efficiency of the respective schemes. In certificateless cryptography schemes, there are three secrets per party:

1. The key issued by the key generation centre (Dent [8] calls it "partial private key"). We assume in the following that this key is ID-based, although it does not necessarily have to be ID-based.
2. The user generated private key x_{ID} (Dent calls it "secret value").
3. The ephemeral value chosen randomly for each session.

Key encapsulation mechanisms (KEM) provide efficient means to communicate a random key from a sender to a designated receiver. Messages used with public key encryption schemes are usually limited in length or have to belong to a specific group. Contrariwise, key encapsulation mechanisms encrypt only a key that is then usually used in a symmetric *data encapsulation mechanism* (DEM) and thus provide increased efficiency over public key encryption. The resulting scheme is then called a *hybrid encryption* scheme [7,6]. Efficient

* Research funded by the Australian Research Council through Discovery Project DP0666065

constructions for a certificateless encryption scheme in the standard model can be obtained from our scheme using the KEM-DEM construction [6,2].

PREVIOUS WORK has identified both identity based key encapsulation mechanisms (IB-KEM) [11,5] (see [12] for a comparison) and certificateless key encapsulation mechanisms (CL-KEM) [2,10]. However, the known constructions for CL-KEM schemes are all generic constructions: they involve running a public key based encryption scheme and an ID-based KEM in parallel and are thus not very efficient. In this work we propose the first direct CL-KEM construction from an efficient IB-KEM in the standard model and prove the construction secure.

THE SECURITY MODEL for our CL-KEM construction is similar to that of previous work by Bentahar et al. [2] and Huang and Wong [10]. We consider a "weak" certificateless adversary that can replace public keys, but cannot request decapsulations of a ciphertext under a replaced public key unless the corresponding *user secret value* is disclosed to the simulator. This is a realistic notion as in real life one cannot expect a user to successfully decrypt ciphertexts that do not correspond to the user's private key. For a full discussion of the security model see Section 3 on the facing page.

THE MAIN CONTRIBUTIONS of this work are:

- First efficient direct construction for a CCA secure certificateless key encapsulation mechanism proven secure in the standard model.
- Simplified proof strategy for certificateless KEM constructions.
- Direct efficient constructions for certificateless CCA secure encryption [2] and key agreement [4] follow from our construction.
- Approximately twice as efficient as the generic construction by Huang and Wong.
- Improved security model for certificateless KEM

2 Definitions

2.1 Target Collision Resistant Hash Function

Let $\mathcal{F} = (\mathsf{TCR}_s)_{s \in S}$ be a family of hash functions for security parameter k and with seed $s \in S$ where S is parametrized by the security parameter k. $\mathcal{F}$ is said to be *collision resistant* if, for a hash function $\mathsf{TCR} = \mathsf{TCR}_s$ with $s \xleftarrow{\$} S$, it is infeasible for an efficient adversary to find two distinct values $x \neq y$ such that $\mathsf{TCR}(x) = \mathsf{TCR}(y)$.

The notion of a *target collision resistant hash function*(TCR) is strictly weaker. The adversary against a target collision resistant hash function is supplied with a randomly drawn hash function $\mathsf{TCR} = \mathsf{TCR}_s$ and a randomly chosen element x. The task of the adversary is to find a y such that $\mathsf{TCR}(x) = \mathsf{TCR}(y)$. Note that the adversary may not select x, and is thus limited with respect to collision resistant hash functions. Target collision resistant hash functions are sometimes also called *universal one-way hash functions*. Naor and Yung [14] and Rompel [15] give efficient constructions for target collision resistant hash functions from

arbitrary one-way functions. In the following we assume that TCR's exist and define the advantage of any efficient polynomial time adversary $\mathcal{M}$ against a randomly chosen hash function $\mathsf{TCR} = \mathsf{TCR}_s$ as

$$\mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{M}}(k) = \Pr[y \xleftarrow{\$} \mathcal{M}(\mathsf{TCR}(\cdot), x) | \mathsf{TCR}(y) = \mathsf{TCR}(x)]$$

The hash function TCR is said to be *target collision resistant* if the advantage for all $\mathcal{M}$ against TCR is negligible in k.

2.2 Admissible Bilinear Pairing

Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order p. A bilinear pairings map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ between the groups $\mathbb{G}$ and $\mathbb{G}_T$ satisfies the following properties:

Bilinear We say that a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is *bilinear* if $e(g^a, g^b) = e(g, g)^{ab}$ for all $g \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$.

Non-degenerate We say that e is non-degenerate if it does not send all pairs in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_T$. Since $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order p, it follows that if $g \in \mathbb{G}$ is a generator of $\mathbb{G}$, then $e(g, g)$ is a generator of $\mathbb{G}_T$.

Computable There is an efficient algorithm to compute $e(g, h)$ for any $g, h \in \mathbb{G}$.

2.3 Decisional Bilinear Diffie-Hellman Problem

The decisional Bilinear Diffie-Hellman assumption states that given $\{g^a, g^b, g^c\} \in \mathbb{G}^3$ it is hard to distinguish $e(g, g)^{abc} \in \mathbb{G}_T$ from a random element $R \xleftarrow{\$} \mathbb{G}_T$. Let $\mathcal{Z}$ be an algorithm that takes as input a triple $\{g^a, g^b, g^c, T\} \in \mathbb{G}^3 \times \mathbb{G}_T$, and outputs a bit $b \in \{0, 1\}$ indicating $T \stackrel{?}{=} e(g, g)^{abc}$. We define the dBDH advantage of $\mathcal{Z}$ to be

$$\mathbf{Adv}^{\text{dBDH}}_{\mathcal{Z}} = \left| \Pr\left[a, b, c \xleftarrow{\$} \mathbb{Z}_p : Z(g^a, g^b, g^c, T) = \left(T \stackrel{?}{=} e(g, g)^{abc} \right) \right] - 1/2 \right|$$

3 Security Model

3.1 Types of certificateless adversaries

In certificateless cryptography it is common to distinguish between two types of adversaries:

Type I: A Type I adversary represents an outsider adversary that does not have access to the secret master key of the key generation centre (KGC).

Type II: A Type II adversary represents an insider adversary that has access to the master secret key (e.g. a malicious KGC).

The security of the scheme is then further classified by the type of decryption oracle access that the adversary has:

Strong security: The adversary has access to a *strong decryption oracle*. This means that the oracle can decrypt ciphertexts even if it does not know the private key that matches the public key used for encryption. Thus it can decrypt a ciphertext $C \in \mathcal{C}$ even if the adversary replaced the certificateless public key that was used to generate the ciphertext and does not disclose the matching private key to the decryption oracle.

Weak security: The adversary has access to a *weak secret value decryption oracle* (Weak SV Decrypt oracle). The oracle can decrypt ciphertexts only if it is given all private keys necessary for decryption. If the adversary replaced a public key, then decryption is only possible if the adversary submits the private key matching the public key along with the decryption request.

In his survey on certificateless encryption schemes, Dent [8] remarks that "the Weak [...] [security] model seems to most realistically reflect the potential abilities of an attacker." All published CL-KEM schemes [10,2] focus on the *weak security* model. We will use this model for our work as well.

3.2 Certificateless Key Encapsulation Mechanism

We use the definition by Huang and Wong [10] for a *certificateless key-encapsulation mechanism (CL-KEM)*. A certificateless KEM consists of the following algorithms:

CL-KEM IBE Setup: On input 1^k where $k \in \mathbb{N}$ is a security parameter, it generates a master public/private key pair (mpk, msk).

CL-KEM IBE KeyDerivation: On input msk and a user identity $\mathsf{ID} \in \{0,1\}^*$, it generates a user partial key / ID-based private key sk_{ID}.

CL-KEM User KeyGen: On input mpk and a user identity ID, it generates a user public/private key pair (upk, usk).

CL-KEM Key Verification: On input mpk and upk, it generates an encryption key enc_k that is used for all following encapsulations. This algorithm needs to run only if the master public key or the user public key change (which should happen less frequent than actual encapsulations take place).

CL-KEM Encapsulation: takes as input $(mpk, enc_k, \mathsf{ID})$ and outputs an encapsulation key pair $(K, C) \in \mathcal{K} \times \mathcal{E}$ where C is called the encapsulation of the key K and $\mathcal{K}$ and $\mathcal{E}$ are the key space and the encapsulation space respectively.

CL-KEM Decapsulation: takes as input $((sk_{\mathsf{ID}}, usk), \mathsf{ID}, C)$ and decapsulates C to get back a key K, or outputs the special symbol $\perp$ indicating invalid encapsulation.

3.3 The security game for CL-KEM

To model the security guarantees of a certificateless scheme correctly, we introduce the following model that merges the requirements by Dent [8] and Huang & Wong [10]. The adversary $\mathcal{M}$ has access to the following oracles:

Reveal master key: The adversary is given access to the master secret key.

Reveal ID-based key(ID): The adversary extracts the ID-based private key of party ID.

Get user public key(ID): The adversary obtains the certificateless public key for ID. If the certificateless key for the identity has not yet been generated, it is generated with the *user key gen* algorithm.

Replace public key(ID, pk): Party ID's certificateless public key is replaced with pk chosen by the adversary. All communication (encryption, encapsulation) for Party ID will use the new public key.

Reveal secret value(ID): The adversary extracts the secret value $\beta_{\mathsf{ID}}, \gamma_{\mathsf{ID}}$ that corresponds to the certificateless public key for party ID. If the adversary issued a *replace public key* query for ID before, $\perp$ is returned.

Decapsulate(ID, C): The adversary learns the decapsulation of C under ID or $\perp$ if C is invalid or if the adversary replaced the public key of ID.

Decapsulate(ID, C, x): The adversary learns the decapsulation of C under ID using the secret value x. The special symbol $\perp$ will be returned if C is invalid.

Get challenge key encapsulation(ID^*): The adversary requests a challenge key encapsulation and thus marks the transition from $\mathbf{Oracles}_1$ to $\mathbf{Oracles}_2$ in Experiment 1. The simulator returns a challenge key encapsulation as described in Experiment 1.

The security game for a CL-KEM scheme is associated with the following experiment:

$$
\begin{aligned}
&\mathbf{Experiment\ Challenge}^{cl-kem-cca}_{\text{CL-KEM}\mathcal{M}}(k):\\
&\quad (mpk, msk) \xleftarrow{\$} \mathbf{CL\text{-}KEM\ IBE\ Setup}(k)\\
&\quad (\mathsf{ID}^*, state) \xleftarrow{\$} \mathcal{M}^{\mathbf{Oracles}_1}(find, mpk)\\
&\quad K_0^* \xleftarrow{\$} \mathcal{K}; (C^*, K_1^*) \xleftarrow{\$} \mathbf{CL\text{-}KEM\ Enc}(pk, \mathsf{ID}^*)\\
&\quad \gamma \xleftarrow{\$} \{0,1\}; K^* = K_\gamma^*\\
&\quad \gamma' \xleftarrow{\$} \mathcal{M}^{\mathbf{Oracles}_2}(\mathtt{guess}, K^*, C^*, state)\\
&\quad \text{Return } \gamma == \gamma'
\end{aligned}
\tag{1}
$$

The advantage an adversary $\mathcal{M}$ has against a CL-KEM scheme is therefore expressed by

$$
\mathbf{Adv}^{\text{CL-KEM}}_{\mathcal{M}}(k) = \left|\Pr\left[\mathbf{Experiment\ Challenge}^{cl-kem-cca}_{\text{CL-KEM}\mathcal{M}}(k)\right] - 1/2\right|
$$

For a *Type I* adversary $\mathcal{M}$, $\mathbf{Oracles}_1$ and $\mathbf{Oracles}_2$ mean access to all oracles listed above with the following limitations:

1. No *reveal master key* queries.
2. C^* must not be submitted to a *decapsulate* oracle under ID^*.
3. Not both (*reveal secret value* OR *replace public key*) AND *reveal ID-based key* oracles may be asked for ID^*.

For a *Type 2* adversary $\mathcal{M}$, $\mathbf{Oracles}_1$ and $\mathbf{Oracles}_2$ are subject to the following limitations:

1. $\mathbf{Oracles}_1$ and $\mathbf{Oracles}_2$ now includes *reveal master key* as allowed query,
2. C^* must not be submitted to a *decapsulate* oracle under ID^*.
3. *reveal secret value* must never be asked for ID^*,
4. $\mathbf{Oracles}_1$ must not include *replace public key* for ID^*.

4 The CL-KEM scheme

We describe the phases of our certificateless key encapsulation mechanism in this section. Our protocol consists of six phases: *setup*, *identity based key derivation*, *user key generation*, *key verification*,*key encapsulation*, and *key decapsulation*. The algorithms *setup*, and *identity based key derivation* are exactly the same as in Kiltz and Galindo's KEM [12]. In the following, we first recapitulate the parameters needed for the Kiltz-Galindo KEM and continue then to describe the differences needed to obtain a certificateless KEM. We will use *bilinear pairings* and *Waters hash* in the scheme, which we describe shortly.

4.1 Waters' Hash

To prove our scheme, we use Waters' hash function $H : \{0,1\}^n \rightarrow \mathbb{G}$ as described in Waters' identity based encryption scheme [17]. On input of an integer n, the randomized hash key generator $\mathsf{HGen}(\mathbb{G})$ chooses $n+1$ random group elements $h_0, h_1, \ldots, h_n \in \mathbb{G}$ and returns $h = (h_0, h_1, \ldots, h_n)$ as the public description of the hash function. The hash function $H : \{0,1\}^n \rightarrow \mathbb{G}^*$ is evaluated on a string $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_n) \in \{0,1\}^n$ as the product $H(\mathsf{ID}) = h_0 \prod_{i=1}^{n} h_i^{\mathsf{ID}_i}$.

4.2 CL-KEM Algorithms

Setup On input of the security parameter k, the key generation center picks suitable bilinear pairing parameters $(e(\cdot,\cdot), p, \mathbb{G}, \mathbb{G}_T, g)$ and uses $\mathsf{HGen}(\mathbb{G})$ to obtain a suitable Waters' hash function. The KGC also publishes system parameters $(u_1, u_2, z) \in \mathbb{G}$. See Algorithm **CL-KEM IBE Setup** in Figure 1 on the next page for details.

Identity-based Key Derivation To generate an ID-based key for an identity $\mathsf{ID} \in \{0,1\}^n$, the key generation centre follows the Algorithm **CL-KEM IBE KeyDerivation** in Figure 1 on the facing page.

User Key Generation To obtain a certificateless KEM, we introduce the new algorithm *user key generation* into the Kiltz-Galindo KEM. The user generates a certificateless key pair from the system parameters as outlined by Algorithm **CL-KEM User Keygen** in Figure 1 on the next page. After key generation, the user publishes *upk* and keeps *usk* private.

CL-KEM IBE Setup(k) :

$u_1, u_2 \xleftarrow{\$} \mathbb{G}^*$

$a, b \xleftarrow{\$} \mathbb{Z}_p{}^*; z \leftarrow e(g^a, g^b)$

$H \xleftarrow{\$} HGen(\mathbb{G})$

$mpk \leftarrow (u_1, u_2, g^a, g^b, z, H)$

$msk \leftarrow \alpha = g^{ab}$

Return(mpk, msk)

CL-KEM IBE KeyDer(msk, ID) :

$s \xleftarrow{\$} \mathbb{Z}_p{}^*$

$sk_{\mathsf{ID}} \leftarrow (\alpha \cdot H(\mathsf{ID})^s, g^s)$

Return(sk_{ID})

CL-KEM User Keygen(mpk, ID) :

$(u_1, u_2, g^a, g^b, z, H) \leftarrow mpk$

$\beta_{\mathsf{ID}}, \gamma_{\mathsf{ID}} \xleftarrow{\$} \mathbb{Z}_p{}^*$

$upk \leftarrow (g^{\beta_{\mathsf{ID}}}, g^{\gamma_{\mathsf{ID}}}, g^{a\beta_{\mathsf{ID}}}, g^{b\gamma_{\mathsf{ID}}})$

$usk \leftarrow \beta_{\mathsf{ID}} \cdot \gamma_{\mathsf{ID}}$

Return(upk, usk)

CL-KEM KeyVer(mpk, upk) :

$(g^{\beta_{\mathsf{ID}}}, g^{\gamma_{\mathsf{ID}}}, g^{a\beta_{\mathsf{ID}}}, g^{b\gamma_{\mathsf{ID}}}) \leftarrow upk$

$(u_1, u_2, g^a, g^b, z, H) \leftarrow mpk$

Check if $e(g^a, g^{\beta_{\mathsf{ID}}}) \stackrel{?}{=} e(g, g^{a\beta_{\mathsf{ID}}})$

and $e(g^b, g^{\gamma_{\mathsf{ID}}}) \stackrel{?}{=} e(g, g^{b\gamma_{\mathsf{ID}}})$

TRUE : $enc_k \leftarrow e(g^{a\beta_{\mathsf{ID}}}, g^{b\gamma_{\mathsf{ID}}})$

FALSE : $enc_k \xleftarrow{\$} \mathbb{G}_T$

Return(enc_k)

CL-KEM Enc$(enc_k, mpk, \mathsf{ID}, M)$:

$(u_1, u_2, g^a, g^b, z, H) \leftarrow mpk$

$r \xleftarrow{\$} \mathbb{Z}_p{}^*$

$c_1 \leftarrow g^r$

$c_2 \leftarrow H(\mathsf{ID})^r, t \leftarrow \mathsf{TCR}(c_1)$

$c_3 \leftarrow (u_1^t \cdot u_2)^r$

$K \leftarrow enc_k{}^r \in \mathbb{G}_T$

$C \leftarrow (c_1, c_2, c_3) \in \mathbb{G}^3$

Return(K, C)

CL-KEM Dec$(sk_{\mathsf{ID}}, usk, C)$:

$c_1, c_2, c_3 \leftarrow C$

$d_1, d_2 \leftarrow sk_{\mathsf{ID}}$

$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p{}^*$

$t \leftarrow TCR(c_1)$

$K \leftarrow \left(\dfrac{e(c_1, d_1 \cdot (u_1^t u_2)^{r_1} \cdot H(\mathsf{ID})^{r_2})}{e(c_2, d_2 \cdot g^{r_2}) e(g^{r_1}, c_3)} \right)^{usk}$

Return(K)

Fig. 1. Our CCA secure CL-KEM.

Certificateless Key Verification We add the new *key verification* algorithm to the certificateless KEM. This algorithm makes sure that the master public key is part of the encryption, and addresses the security issue discovered in [16]. If the master public key is not part of the encryption, key decapsulation will result in a key that is different from the key generated during encapsulation. The output of the key verification algorithm is the encapsulation base key enc_k, that is then used for key encapsulation.

Certificateless Key Encapsulation We modify the Kiltz-Galindo encapsulation mechanism by using enc_k instead of z for encryption. Thus we get a very efficient encapsulation mechanism, outlined by Algorithm **CL-KEM Enc** in Figure 1 on the preceding page. The key K is used for encryption, C is the certificateless encapsulation of K.

Certificateless Key Decapsulation Decapsulation is also very efficient as it needs only one additional exponentiation over the Kiltz-Galindo KEM decapsulation algorithm. The Algorithm **CL-KEM Dec** in Figure 1 on the previous page describes the decapsulation.

This concludes the description of the certificateless KEM construction.

5 Efficiency comparison

When compared to the only other CL-KEM in the standard model by Huang and Wong [10], we note that both key generation and encapsulation are twice as efficient, we save one exponentiation during decapsulation, key size is smaller and ciphertext size is approximately halved. On the other hand, we introduce the new *key verification* algorithm, which adds 5 pairings. These can be seen as three fixed-base pairings: regular pairings are $e(g^a, g^{\beta_{\mathsf{ID}}})$, $e(g^b, g^{\gamma_{\mathsf{ID}}})$ and $e(g^{a\beta_{\mathsf{ID}}}, g)$, fixed base from here are $e(g^{b\gamma_{\mathsf{ID}}}, g)$ (keeping g fixed) and $e(g^{a\beta_{\mathsf{ID}}}, g^{b\gamma_{\mathsf{ID}}})$ (keeping $g^{a\beta_{\mathsf{ID}}}$ fixed). We suspect that speedups that are available for fixed point multiplicaion in elliptic curve cryptography will carry over to the pairings case, using a memory tradeoff. See Hankerson, Menezes and Vanstone [9, Chapter 3.3.2] for details in the elliptic curve case. For a detailed comparison of our scheme with the CL-KEM construction in the standard model by Huang and Wong [10] see Table 1 on the facing page.

6 Proof of security for the CL-KEM

Theorem 1 *Assume* TCR *is a target collision resistant hash function. Under the decisional Bilinear Diffie-Hellman assumption relative to the generator G, the CL-KEM from Section 4 on page 6 is secure against chosen ciphertext attacks.*

Proving the protocol is easier if we do not treat *Type I* and *Type II* adversaries separately. Essentially, there are two strategies for dealing with an adversary:

Scheme	KeyGen	Enc	Dec	Keysize	Ciphertext
	#pairings + #[multi,regular,fixed-base]-exp			pk	overhead
IB-KEM [12]	0 + [0,2,0]	0 + [1,3,1]	3 + [1,0,2]	n+4	$3l$
+ PKE [13]	0 + [0,4,0]	0 + [0,4,0]	0 + [0,2,0]	4	$2l$
= CL-KEM [10]	0 + [0,6,0]	0 + [1,7,1]	3 + [1,2,2]	n+8	$5l$
Ours	0 + [0,3,0]	0 + [1,3,1]	3 + [1,1,2]	n+4	$3l$

We instantiate the Huang & Wong [10] scheme with the most efficient CCA2 secure PKE scheme by Kurosawa & Desmedt [13] and the most efficient CCA2 secure ID-based KEM by Kiltz & Galindo [12] and compare it to our direct construction from the Kiltz & Galindo KEM.

Table 1. Comparison of the Huang-Wong scheme with our scheme

- Embed the challenge into the ID-based part. Then the adversary may learn the secret value or replace the certificateless public key. This is generally not applicable for *Type II* adversaries.
- Embed the challenge into the CL-based part. Then the adversary may learn the ID-based secret key. This is applicable for both *Type I* and *Type II* adversaries.

For Type I adversaries that want to learn the CL-key, we use the proof from Kiltz and Galindo [12] unmodified and hand over the *user secret value* x_{ID} to the adversary. The original proof does still hold in this setting.

For Type II adversaries and Type I adversaries that want to learn the ID-based key, we have to modify the proof. The simulator $\mathcal{B}$ gets the dBDH challenge (g, g^a, g^b, g^c, T) from its challenger. Given that the adversary $\mathcal{M}$ has an advantage in the CL-KEM game, $\mathcal{B}$ uses the adversary $\mathcal{M}$ to get an advantage in solving the dBDH challenge. This strategy simplifies proving the security of the scheme: a well known proof in the ID-based setting is expanded only with what is necessary for the certificateless setting. As it turns out, the proof for the CL-part of the scheme is easier to understand as it does not have to deal with artificial aborts.

We rewrite the proof by Kiltz and Galindo to get a proof for the CL-KEM scheme for Type II adversaries. As in Kiltz & Galindo's paper, the main idea is again that the simulator knows a back door for the hash function H. Knowing the back door for H allows the simulator to let H "vanish" for the target identity. To achieve this, we have to embed the challenge slightly differently from the original proof by Kiltz and Galindo [12]. We also use a game based approach. The simulator $\mathcal{B}$ starts with knowing the discrete logarithms of g^a, g^b, g^c and "forgets" the discrete logarithms during modifications of the game.

Game 0.(Forget b) The simulator $\mathcal{B}$ picks $(a, b, c) \xleftarrow{\$} {\mathbb{Z}_p}^*$, computes g^c and $t^* = \mathsf{TCR}(g^c)$ and additionally picks $d, \delta \xleftarrow{\$} {\mathbb{Z}_p}^*$. The *CL-KEM IBE Setup* algorithm is modified as follows:

CL-KEM IBE Setup(k) :

$$\begin{aligned}
&\gamma \xleftarrow{\$} \mathbb{Z}_p, u_1 = g^a, u_2 = (g^a)^{-t^*} g^d, \alpha = g^b; z \leftarrow e(g,\alpha) = e(g,g^b) \\
&H \xleftarrow{\$} HGen(\mathbb{G}) \\
&mpk \leftarrow (u_1, u_2, g^{1/\delta}, g^{b\delta}, z, H); msk \leftarrow \alpha \\
&\text{Return}(mpk, msk)
\end{aligned} \tag{2}$$

We assume that the adversary $\mathcal{M}$ makes no more than q_0 queries for distinct identities. One of these identities will be used to create the challenge ciphertext. We enumerate these queries. The simulator $\mathcal{B}$ guesses the index of the target identity ID^* that the adversary will use in the test query by selecting $q^* \xleftarrow{\$} \mathbb{Z}_{q_0}$. We also assume that the adversary does not make more than q decapsulation queries. B sets the target identity's certificateless public key to $e(g^a, g^b) = z^a$. Both the KGC public key and the master secret key $\alpha = g^b$ can be given to the adversary at the start of the game.

FIND PHASE. During its execution, $\mathcal{M}$ makes a number of *reveal master key*, *reveal ID-based key*, *reveal secret value*, *replace public key*, and *decapsulate* requests. The simulator deals with the adversary's queries in the following way:

Get master key: $\mathcal{B}$ returns α.

Get user public key(ID)**:** If these requests target an identity that has not been initialized before, there are two possibilities: If it is the q^*th distinct query, the simulator picks $\epsilon \xleftarrow{\$} \mathbb{Z}_p*$ and returns $(g^{a\epsilon}, g^{1/\epsilon}, g^{a\epsilon/\delta}, g^{b\delta/\epsilon})$. Otherwise, the simulator generates a new certificateless keys on the fly as specified by **CL-KEM User Keygen**, publishes the ID's certificateless public key in the directory of certificateless public keys and records the certificateless private key $usk = \beta_{\mathsf{ID}} \cdot \gamma_{\mathsf{ID}}$ along with the ID in a table (later referred to as the *table of certificateless private keys*).

Replace user public key$(\mathsf{ID}, upk_{\mathsf{ID}})$**:** The simulator inserts the new certificateless public key upk'_{ID} into the table of certificateless public keys and inserts $\perp$ into the *table of certificateless private keys* at position ID.

Reveal ID-based key(ID)**: (only Type I)** As the simulator knows $\alpha = g^b$ these queries can always be answered throughout the game for Type I adversaries. For Type II adversaries, α can be passed to the adversary at the start of the game. Then it is not necessary to provide this functionality to the adversary (the adversary may compute the keys on its own).

Decapsulation(C, ID)**:** The simulator returns the decapsulation of C under ID query using the entry from the *table of certificateless private keys* or $\perp$ if the certificateless public key was replaced by the adversary or C is an invalid encapsulation.

Decapsulation(C, ID, x)**:** The simulator returns the decapsulation of C under ID query using x as the user secret value or $\perp$ if C is an invalid encryption.

Eventually, the adversary returns a target identity ID^*. The simulator chooses a random key K_0^* and runs the encapsulation algorithm to create a key K_1^*

together with the challenge ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$. The challenge ciphertext is computed as

$$c_1^* \leftarrow g^c, t^* \leftarrow \mathsf{TCR}(g^c), c_2^* = H(\mathsf{ID}^*)^c, c_3^* = (u_1^{t^*} u_2)^c$$

Then, the simulator chooses a random bit b and the challenge ciphertext C^* is returned together with the key $K^* = K_b^*$ to the adversary.

Guess Phase. The adversary continues to query the oracles provided by the simulator under the condition that he may not request a decapsulation of C^* under ID^* and may not request the *user secret value* x_{ID^*}. Finally, the adversary returns a bit b'. If $b' = b$ then the simulator returns 1, else he returns 0. This completes the description of the simulator. Let X_i denote the event that the adversary $\mathcal{M}$ wins game i. Thus we have for the advantage of the adversary against the CL-KEM scheme: $\mathbf{Adv}_{CL-KEM,\mathcal{M}}^{cl-kem-cca} = |\Pr[X_0] - 1/2|$.

Game 1.(Eliminate hash collisions) The simulator fixed $c_1^* = g^c$ and $t^* = \mathsf{TCR}(g^c)$ at the start of the game and aborts if a decapsulation query is made for any ciphertext $C = (c_1, c_2, c_3)$ for that $\mathsf{TCR}(c_1) = t^*$ and $c_1 \neq c_1^*$. Otherwise, Game 0 and Game 1 are identical. This event happens only with negligible probability as otherwise $\mathcal{M}$ could be used as an efficient adversary against TCR. Thus we have

$$|\Pr[X_1] - \Pr[X_0]| \leq \mathbf{Adv}_{\mathsf{TCR},\mathcal{M}}^{\text{hash-tcr}}(k)$$

Game 2.(Change of hash keys) The game continues as in Game 1 except that the simulator changes the way the hash keys $\boldsymbol{h} = (h_0, h_1, \ldots, h_n)$ are generated. Set $m = 2q$ (where q is the upper bound on the decapsulation queries) and randomly choose

$$\begin{gathered} x_0, x_1, \ldots, x_n \xleftarrow{\$} \{0, \ldots, p-1\}; \quad y_0', y_1, \ldots, y_n \xleftarrow{\$} \{0, \ldots, m-1\} \\ k \xleftarrow{\$} \{0, \ldots, n\} \end{gathered} \tag{3}$$

and set $y_0 \leftarrow p - km + y_0'$.

$\mathcal{B}$ redefines the public hash keys $\boldsymbol{h} = \{h_0, \ldots, h_n\}$ as $h_i = g^{x_i} u_1^{y_i} = g^{x_i}(g^a)^{y_i}$ for $0 \leq i \leq n$. Thus, the public hash function H evaluated at identity $\mathsf{ID} \in \{0,1\}^n$ is given by

$$H(\mathsf{ID}) = h_0 \prod_{i=1}^{n} h_i^{\mathsf{ID}_i} = g^{x(\mathsf{ID})} u_1^{y(\mathsf{ID})} = g^{x(\mathsf{ID})}(g^a)^{y(\mathsf{ID})}$$

with $x(\mathsf{ID}) = x_0 + \sum_{i=1}^{n} \mathsf{ID}_i x_i$ and $y(\mathsf{ID}) = y_0 + \sum_{i=1}^{n} \mathsf{ID}_i y_i$ (where $x()$ and $y()$ are only known to the simulator). As this does not change the distribution of the hash keys, the probability of success for the adversary does not change:

$$\Pr[X_2] = \Pr[X_1]$$

Game 3.(Abort for wrong challenge identity) The simulation proceeds as in Game 2. Once the simulator is being asked the *challenge ciphertext* query, it

checks the ID^* is the q^*th distinct identity and aborts otherwise. The simulator also aborts if $y(\mathsf{ID}^*) \neq 0$.

As we do not need to change the key derivation oracle during the sequence of games (as Kiltz and Galindo do), we can simplify the proof significantly. We especially do not have to deal with artificial aborts, as the abort probability for the simulator can be estimated directly using results from Kiltz and Galindo [12, Section A.2]. From Equation 3 on the preceding page we have that

$$y(\mathsf{ID}^*) = 0 = p - km + y'_0 + \sum_{i=1}^{n} \mathsf{ID}_i^* y_i$$

and from the distribution of the y_i we get that

$$0 \leq y'_0 + \sum_{i=1}^{n} \mathsf{ID}_i^* y_i < (n+1)m$$

Thus if $y(\mathsf{ID}^*) = 0 \bmod m$, then there is a unique $0 \leq k < n+1$ such that $y(\mathsf{ID}^*) = 0$ over the integers. Since k is uniformly and independently distributed over the integers, we get:

$$\Pr[y(\mathsf{ID}^*) = 0] = \Pr[y(\mathsf{ID}^*) = 0 \bmod p] \geq \Pr[y(\mathsf{ID}^*) = 0 \bmod m]/(n+1)$$

Thus for a fixed k and $b \in \mathbb{Z}_m$ we have that $\Pr[y(\mathsf{ID}) = b \bmod m] = 1/m$. So we conclude with

$$\Pr[y(\mathsf{ID}^*) = 0] \geq \frac{1}{n+1} \Pr[y(\mathsf{ID}^*) = 0 \bmod m] = \frac{1}{n+1} \cdot \frac{1}{m} = \frac{1}{m(n+1)}$$

Thus, the probability that Game 3 succeeds is given by the probability that $y(\mathsf{ID}^*) = 0$ and that ID^* is the q^*th distinct identity. As there are at most q_0 distinct ID queries by the adversary we have

$$\Pr[X_3] \geq \Pr[X_2]/(q_0 m(n+1))$$

Game 4.(Change of decapsulation oracle / Forget a) The simulator knows all *user secret keys* except for those the adversary replaced with a *replace certificateless public key* request. Regarding decapsulation queries, the simulator does not have to answer requests for identities that were issued a *replace certificateless public key* query unless the adversary supplies the *user secret key* matching the replaced certificateless public key. As the simulator can derive *ID-based private keys* from the master parameters, answering decapsulation queries for all identities except ID^* is easy, as all secret information to do this is readily available using the standard **CL-KEM Dec** algorithm as described in Figure 1 on page 7.

The simulator established in Game 3 that $y(\mathsf{ID}^*) = 0$. This enables the simulator to answer decapsulation queries for ID^* in the following way: instead of answering the decapsulation as in **CL-KEM Dec** in Figure 1 on page 7, the simulator computes the decapsulations for ID^* as follows: with $u_1 = g^a, u_2 = (g^a)^{-t^*} g^d$ and $c_1 = g^r$ we have

$$c_3 = (u_1^t u_2)^r = ((g^a)^t g^{-t^* a} g^d)^r = ((g^{a(t-t^*)} g^d)^r = (c_1^a)^{t-t^*} \cdot c_1^d.$$

To decapsulate the correct key K, we would like to compute $e(g^a, g^b)^r$. Thus knowing g^b and computing $c_1^a = (g^r)^a = g^{ra}$ will allow us to compute K by computing $e(g^{ra}, g^b) = e(g, g)^{rab}$:

$$\left(c_3/c_1^d\right)^{\frac{1}{t-t^*}} = \left((c_1^a)^{t-t^*} \cdot c_1^d/c_1^d\right)^{\frac{1}{t-t^*}} = (c_1^a)^{\frac{t-t^*}{t-t^*}} = c_1^a = g^{ra}$$

As $K = e(g^a, g^b)^r = e(g, g)^{abr}$, knowing $t = TCR(g^r)$ we can recompute K with

$$K = e\left(g^b, \left(c_3/c_1^d\right)^{\frac{1}{t-t^*}}\right) = e(g^b, g^{ar}) = e(g, g)^{abr}$$

As this behaviour does not alter the adversary's view of the game we have

$$\Pr[X_4] = \Pr[X_3]$$

Game 5.(Modify the challenge / Forget c) The simulator changes its answer to the *get challenge key encapsulation* query. Game 3 established that $y(\mathsf{ID}^*) = 0 \bmod p$, thus the challenger can compute the challenge ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$ as

$$c_1^* = g^c, c_2^* = (g^c)^{x(\mathsf{ID}^*)}, c_3^* = (g^c)^d, K = T$$

where g^c and T are given by the challenger before the game starts. Now the answer of the adversary to the challenge ciphertext is directly related to the challenge, and thus the simulator has an advantage in solving the dBDH challenge if the adversary has an advantage in winning the game:

$$\mathbf{Adv}_{CL-KEM,\mathcal{M}}^{cl-kem-cca} = \left|\Pr[X_0] - \frac{1}{2}\right| \leq \left|\frac{1}{q_0 m(n+1)}\mathbf{Adv}_{\mathcal{M}}^{\text{dBDH}}(k) + \mathbf{Adv}_{\mathsf{TCR},\mathcal{M}}^{\text{hash-tcr}}(k) - \frac{1}{2}\right|$$

7 Conclusion

We show how to construct an efficient CL-KEM scheme from an existing ID-based KEM scheme in the standard model. Our construction requires only one additional exponentiation during the construction of the certificateless key and one additional exponentiation during the decapsulation compared to the original ID-based KEM scheme and is thus more efficient than any generic construction that has been published before. By modifying the Kiltz-Galindo KEM scheme [12] which is one of the most efficient ID-based KEM schemes in the standard model, we obtain the most efficient CL-KEM scheme in the standard model today.

References

1. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public Key Cryptography. In Laih, C.S., ed.: ASIACRYPT. Volume 2894 of Lecture Notes in Computer Science., Springer (2003) 452–473 Online available at http://eprint.iacr.org/2003/126.pdf. 1

2. Bentahar, K., Farshim, P., Malone-Lee, J., Smart, N.P.: Generic Constructions of Identity-Based and Certificateless KEMs. J. Cryptology **21**(2) (2008) 178–199 2, 4
3. Boneh, D., Franklin, M.: Identity based encryption from the Weil pairing. SIAM Journal of Computing **32**(3) (2003) 586–615 Online available at http://crypto.stanford.edu/~dabo/papers/bfibe.pdf. 1
4. Boyd, C., Cliff, Y., González Nieto, J.M., Paterson, K.G.: Efficient one-round key exchange in the standard model. In Mu, Y., Susilo, W., Seberry, J., eds.: ACISP. Volume 5107 of Lecture Notes in Computer Science., Springer (2008) 69–83 2
5. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In Atluri, V., Meadows, C., Juels, A., eds.: ACM Conference on Computer and Communications Security, ACM (2005) 320–329 2
6. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. **33**(1) (2004) 167–226 1, 2
7. Dent, A.W.: A Designer's Guide to KEMs. In Paterson, K.G., ed.: Cryptography and Coding. Volume 2898 of Lecture Notes in Computer Science., Springer (2003) 133–151 1
8. Dent, A.W.: A survey of certificateless encryption schemes and security models. International Journal of Information Security **7**(5) (October 2008) 349–377 1, 4
9. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2003) 8
10. Huang, Q., Wong, D.S.: Generic Certificateless Key Encapsulation Mechanism. In Pieprzyk, J., Ghodosi, H., Dawson, E., eds.: ACISP. Volume 4586 of Lecture Notes in Computer Science., Springer (2007) 215–229 1, 2, 4, 8, 9
11. Kiltz, E., Galindo, D.: Direct Chosen-Ciphertext Secure Identity-Based Key Encapsulation Without Random Oracles. In Batten, L.M., Safavi-Naini, R., eds.: ACISP. Volume 4058 of Lecture Notes in Computer Science., Springer (2006) 336–347 1, 2
12. Kiltz, E., Galindo, D.: Direct Chosen-Ciphertext Secure Identity-Based Key Encapsulation without Random Oracles. Cryptology ePrint Archive, Report 2006/034 (2006) http://eprint.iacr.org/2006/034. 2, 6, 9, 12, 13
13. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In Franklin, M.K., ed.: CRYPTO. Volume 3152 of Lecture Notes in Computer Science., Springer (2004) 426–442 9
14. Naor, M., Yung, M.: Universal One-Way Hash Functions and their Cryptographic Applications. In: STOC, ACM (1989) 33–43 2
15. Rompel, J.: One-Way Functions are Necessary and Sufficient for Secure Signatures. In: STOC, ACM (1990) 387–394 2
16. Selvi, S.S.D., Vivek, S.S., Rangan, C.P.: Certificateless kem and hybrid signcryption schemes revisited. Cryptology ePrint Archive, Report 2009/462 (2009) http://eprint.iacr.org/2009/462. 1
17. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of Lecture Notes in Computer Science., Springer (2005) 114–127 6