J. Cryptol. (2008) 21: 178–199
DOI: 10.1007/s00145-007-9000-z

Journal of
CRYPTOLOGY
© 2007 International Association for Cryptologic Research

Original Paper

Generic Constructions of Identity-Based and Certificateless KEMs

K. Bentahar, P. Farshim, J. Malone-Lee, and N.P. Smart
Department of Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, UK
bentahar@cs.bris.ac.uk; farshim@cs.bris.ac.uk; malone@cs.bris.ac.uk; nigel@cs.bris.ac.uk

Communicated by Mihir Bellare

Received 24 February 2005 and revised 10 October 2006
Online publication 27 July 2007

Abstract. We extend the concept of key encapsulation to the primitives of identity-based and certificateless encryption. We show that the natural combination of ID-KEMs or CL-KEMs with data encapsulation mechanisms results in encryption schemes that are secure in a strong sense. In addition, we give generic constructions of ID-KEMs and CL-KEMs that are provably secure in the random oracle model.

1. Introduction

The natural way to perform public-key encryption for large messages is to separate the encryption into two parts: one part uses public-key techniques to encrypt a one-time symmetric key; the other part uses the symmetric key to encrypt the actual message. In such a construction, the public-key part of the algorithm is known as the *key encapsulation mechanism* (KEM) while the symmetric-key part—where the message is actually encrypted—is known as the *data encapsulation mechanism* (DEM). The formalization of this basic approach originates in the work of Shoup [18]. The resulting KEM/DEM encryption paradigm has received much attention in recent years [11,12,18]. It is very attractive as it gives a clear separation between the various parts of the cipher allowing for modular design.

In [12] Dent proposes a number of generic constructions of KEMs from standard public-key encryption schemes. The KEMs themselves are secure in a strong sense. However, the encryption schemes from which they are built require only a weak notion of security. It is this line of work that we extend here by applying these techniques to two types of recently introduced, but closely related, primitives: identity-based encryption (IBE) and certificateless encryption (CL).

A secure and efficient IBE scheme was introduced by Boneh and Franklin [9], based on pairings on elliptic curves. Many others have also been proposed, for example [8, 10,19]. One of the contributions of this paper is to formalize the notion of key encapsulation for the identity-based setting. We also present a generic construction of an

ID-KEM, that is secure in a strong sense, from any IBE scheme, that is secure in a weak sense. We feel that our construction of an IBE scheme from an ID-KEM and a standard DEM is more natural than the construction in [9], which relies on the Fujisaki–Okamoto transform [13].

In addition to the work on IBE, we also present a security model for key encapsulation applied to certificateless encryption and a generic construction for such schemes. This form of encryption was introduced and developed in a series of publications by Al-Riyami and Paterson [1–3]. The idea is to have the benefit of IBE (the absence of certificates) without the drawback (key-escrow). We describe a generic construction of a certificateless variant of a KEM, which we call a CL-KEM. Our generic construction takes any (weakly secure) IBE scheme plus a special form of (weakly secure) public-key scheme, such as RSA or ElGamal in certain groups, and uses them to construct a CL-KEM. The resulting scheme is secure in a strong sense.

Since we use a generic public-key scheme in our construction one can now add certificateless encryption to an infrastructure of existing RSA and ElGamal keys, which are either not certified or whose certificates are not trusted by the sender. This possibility to take a set of already deployed public keys and to use them in a certificateless manner allows certificateless encryption to be used in situations outside the scope of existing technology.

We show that combining a CL-KEM with a standard DEM results in a secure certificateless encryption scheme. However, in order to prove our composition result we need to modify the definitions of security for a certificateless encryption scheme and CL-KEMs slightly. We will discuss this point further once we have introduced the appropriate security notions.

Our paper proceeds as follows. In Section 3 we give the security definitions for the primitives that we are interested in: standard public-key encryption, IBE and certificateless encryption. In Section 4 we show a simple generalization of the hybrid result of Cramer and Shoup [11], which allows one to combine any ID/CL-KEM meeting a particular security definitions with a standard DEM so as to meet the security definitions of an encryption scheme, as presented in Section 3. In Section 5 we present our generic construction of an ID-KEM, and prove that it is secure under our definitions. Finally, in Section 6 we present our construction of a generic CL-KEM and we prove that it is secure.

Our generic constructions of ID and CL-KEMs are all in the random oracle model [7]. It is clear, since our composition theorem is proved in the standard model, that if one has an ID-KEM which was secure in the standard model then the resulting hybrid IBE scheme would also be secure in the standard model. We leave it as an open question to find generic constructions of strongly secure ID and CL-KEMs in the standard model from weaker primitives.

We note that recently IBE schemes have been proposed that are secure in the standard model [8,15,19]. However, these schemes have extremely large key-sizes and they are not nearly as efficient as our constructions, especially when chosen ciphertext security is considered. In many of these standard model schemes to achieve chosen ciphertext security requires using a two-level hierarchical IBE scheme.

2. Conventions and Notation

In the following sections, where we give definitions, we do not explicitly define set-up algorithms which define the domain parameters for the schemes, such as underlying groups. This is to simplify and lighten the notation. Our results can be expanded to cope with this by inserting a domain parameter generation algorithm that takes as input 1^t, where t is a security parameter. The output of this algorithm would then be passed to the key-generation algorithms.

In addition, to simplify our discussion, we assume that all encryption algorithms are sound in that any ciphertext produced by the genuine encryption algorithm will always decrypt correctly. We make an analogous set of assumptions for KEMs. All our concrete constructions do indeed satisfy this condition, but our general results can be extended to cover schemes with a weaker soundness definition in the standard way [11].

If S is a set then we write $v \leftarrow S$ to denote the action of sampling from the uniform distribution on S and assigning the result to the variable v. If S contains one element s we use $v \leftarrow s$ as shorthand for $v \leftarrow \{s\}$.

We shall be concerned with probabilistic polynomial-time (PPT) algorithms. If A is such an algorithm we denote the action of running A on input I and assigning the resulting output to the variable v by $v \leftarrow A(I)$. Note that since A is probabilistic, $A(I)$ is a probability space and not a value.

If $\mathtt{E}$ is an event defined in some probability space, we denote the probability that $\mathtt{E}$ occurs by $\Pr[\mathtt{E}]$ (assuming the probability space is understood from the context).

In our proofs we will make use of the following key lemma [11].

Lemma 1. *Let* $\mathtt{U}_1$, $\mathtt{U}_2$ *and* $\mathtt{F}$ *be events defined on some probability space. Suppose that* $\Pr[\mathtt{U}_1 \wedge \neg\mathtt{F}] = \Pr[\mathtt{U}_2 \wedge \neg\mathtt{F}]$, *then*

$$|\Pr[\mathtt{U}_1] - \Pr[\mathtt{U}_2]| \leq \Pr[\mathtt{F}].$$

3. Public-Key, Identity-Based and Certificateless Encryption Schemes

3.1. *Public-Key Encryption*

In this section we recap on some basic definitions of public-key encryption schemes and introduce some additional terminology that we require.

Let the message space be denoted $\mathbb{M}_{\mathtt{PK}}(\cdot)$, the ciphertext space by $\mathbb{C}_{\mathtt{PK}}(\cdot)$, and the space from which randomness used in encryption comes from by $\mathbb{R}_{\mathtt{PK}}(\cdot)$. All of these spaces are specified as functions that, on input of a public key, return a description of a finite set. A public-key encryption scheme consists of the three PPT algorithms $(\mathbb{G}_{\mathtt{PK}}, \mathbb{E}_{\mathtt{PK}}, \mathbb{D}_{\mathtt{PK}})$.

- $\mathbb{G}_{\mathtt{PK}}(1^t)$ is the key generation algorithm. This takes as input 1^t. It outputs a public/private key pair $(\mathfrak{pk}, \mathfrak{sk})$.
- $\mathbb{E}_{\mathtt{PK}}(\mathfrak{pk}, m; r)$ is the encryption algorithm. This takes as input $\mathfrak{pk}$ and a message $m \in \mathbb{M}_{\mathtt{PK}}(\mathfrak{pk})$, plus possibly some randomness $r \in \mathbb{R}_{\mathtt{PK}}(\mathfrak{pk})$. It outputs the corresponding ciphertext $c \in \mathbb{C}_{\mathtt{PK}}(\mathfrak{pk})$.

- $\mathbb{D}_{\mathtt{PK}}(\mathfrak{sk}, c)$ is the decryption algorithm. On input of $\mathfrak{sk}$ and c this outputs the corresponding value of m or a failure symbol $\perp$.

The *de facto* definition of security for public-key encryption schemes is indistinguishability under adaptive chosen ciphertext attack (IND-CCA2) [6]. When using an indistinguishability definition for security one clearly needs to restrict the message space to consist of messages of a fixed length. However, we will require a much weaker form of security in our construction of a generic CL-KEM, namely OW-CPA^{++}. Here OW is an acronym for one-way (i.e. the adversary's task is to invert the encryption function) and the attack model CPA^{++} is an attack model in which an adversary has access to the following oracles.

- A ciphertext validity oracle that checks whether a given ciphertext is valid or not.
- A plaintext checking oracle that, on input of a message and a ciphertext, checks whether the ciphertext is an encryption of the message, for a given public key.
- A ciphertext equality oracle that, on input of two ciphertexts, checks if they are encryptions of the same message under a given public key.

The model in which an adversary only has access to the first of these oracles was first formalized by Pointcheval and Okamoto under the name PCA (plaintext checking attack) [17]. Dent [12] calls this model the CPA^{+} model. This motivates our own naming. Schemes that are secure in the sense of OW-CPA^{++} are readily available: raw RSA under the RSA assumption, or ElGamal under the assumption that the gap Diffie–Hellman problem [16] is hard, for example.

Public-key encryption schemes for which an explicit algorithm exists to implement a plaintext checking oracle will be called *verifiable*. Clearly, a verifiable encryption scheme cannot be secure in the sense of indistinguishability of encryptions. Note that textbook RSA is verifiable, as is ElGamal if one implements it in a group on which there is a bilinear pairing.

3.2. *Identity-Based Encryption Schemes*

Here we give the definition and security notions for an IBE scheme, as first introduced by Boneh and Franklin [9]. Extending our earlier notation we denote the message, ciphertext and randomness spaces of an IBE scheme by $\mathbb{M}_{\mathtt{ID}}(\cdot)$, $\mathbb{C}_{\mathtt{ID}}(\cdot)$ and $\mathbb{R}_{\mathtt{ID}}(\cdot)$, respectively. These are functions that, on input of a master public key $M_{\mathfrak{pk}}$, return descriptions of finite sets. An IBE scheme consists of four algorithms:

- $\mathbb{G}_{\mathtt{ID}}(1^t)$ is a PPT algorithm that takes as input 1^t and returns the master public key $M_{\mathfrak{pk}}$ and the master secret key $M_{\mathfrak{sk}}$.
- $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{sk}}, \mathtt{ID}_A)$ is the private key extraction algorithm. It takes as input $M_{\mathfrak{sk}}$ and $\mathtt{ID}_A \in \{0,1\}^*$, an identifier string for A, and it returns the associated private key $D_{\mathtt{ID}_A}$. This algorithm may be deterministic or probabilistic.
- $\mathbb{E}_{\mathtt{ID}}(\mathtt{ID}_A, M_{\mathfrak{pk}}, m; r)$ is the encryption algorithm. On input of an identifier $\mathtt{ID}_A$, the master public key $M_{\mathfrak{pk}}$, a message $m \in \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$, and possibly some randomness $r \in \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})$, this algorithm outputs $c \in \mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.
- $\mathbb{D}_{\mathtt{ID}}(D_{\mathtt{ID}_A}, c)$ is the deterministic decryption algorithm. On input of the private key $D_{\mathtt{ID}_A}$ and a ciphertext c, this outputs the corresponding plaintext m or a failure symbol $\perp$.

Security notions related to IBE schemes can be found in [9]. Our generic constructions will make use of schemes which are ID-OW-CPA secure, but we shall use these to construct schemes which are ID-IND-CCA2 secure. Again for ID-IND-CCA2 schemes we need to assume a message space consisting of messages of a given fixed length.

To cope with probabilistic ciphers, we require that not too many choices for r encrypt a given message to a given ciphertext. Let $\gamma(M_{\mathfrak{pk}})$ be the least upper bound

$$|\{r \in \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}}) : \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; r) = c\}| \leq \gamma(M_{\mathfrak{pk}}) \tag{1}$$

for every $\mathtt{ID}$, $m \in \mathbb{M}_{\mathtt{PK}}(M_{\mathfrak{pk}})$ and $c \in \mathbb{C}_{\mathtt{PK}}(M_{\mathfrak{pk}})$. Our requirement is that the quantity $\gamma(M_{\mathfrak{pk}})/|\mathbb{R}_{\mathtt{PK}}(M_{\mathfrak{pk}})|$ is a negligible function of the security parameter.

Note that it is not necessarily impossible for there to be a secure cryptosystem without the above property. There could be a cryptosystem for which, for all public keys, there is one message in the message space that always encrypts to the same ciphertext, whatever random input is used for encryption. Unless the adversary is able to find this particular message however, it does not help it to break the cryptosystem.

3.3. *Certificateless Encryption Schemes*

We now describe certificateless encryption schemes as proposed by Al-Riyami and Paterson [1–3]. We expand on the details here in some length as understanding the security model, and in particular our new model of Type-I$^-$ adversaries, will be crucial to understanding the rest of the paper.

A certificateless scheme makes use of a trusted third party known as a key generation center (KGC). Unlike the trusted party in an identity-based setting, the KGC does not have access to users' private keys. The KGC uses a global secret key to compute *partial private keys* for users from their identities. Partial private keys are passed from the KGC to the users in a possibly untrusted manner. See [1] for a discussion of the transmission mechanism in more detail.

Suppose that user A with identity $\mathtt{ID}_A$ has been supplied with partial private key $D_{\mathtt{ID}_A}$ by the KGC. This user combines $D_{\mathtt{ID}_A}$ with some additional secret information—its *secret value*—to generate its full private key S_A. The secret value is not known to the KGC and therefore S_A is not known to the KGC either. User A computes its public key from its secret value; it can do this without knowing $D_{\mathtt{ID}_A}$. We denote the public key $\mathfrak{pk}_A$.

The system is not identity-based: the public key of a user cannot be derived from its identity alone. Instead, a user publishes its public key in some publicly accessible directory. Unlike a traditional PKI, it is not necessary to obtain and verify certificates for public keys in this scenario. Before presenting the formal definitions we note that the notion of certificateless encryption is closely related to that of certificate based encryption [14].

Formally, a certificateless scheme consists of seven polynomial time algorithms, where the message, ciphertext and randomness spaces are defined as before:

– $\mathbb{G}_{\mathtt{CL}}(1^t)$ is a PPT algorithm that takes as input 1^t and returns the master public key $M_{\mathfrak{pk}}$ and the master secret key $M_{\mathfrak{sk}}$.

- $\texttt{Partial-Private-Key-Extract}$ takes as input $M_{\mathfrak{sk}}$ and an identifier string for A, $\texttt{ID}_A \in \{0,1\}^*$, and returns a partial private key $D_{\texttt{ID}_A}$. This algorithm may be deterministic or probabilistic.
- $\texttt{Set-Secret-Value}$ is a PPT algorithm that takes no input (bar the system parameters) and outputs a secret value $\mathfrak{sk}_A$.
- $\texttt{Set-Public-Key}$ is a deterministic algorithm that takes as input the secret value $\mathfrak{sk}_A$ and outputs a public key $\mathfrak{pk}_A$.
- $\texttt{Set-Private-Key}$ is a deterministic algorithm that takes as input a partial private key $D_{\texttt{ID}_A}$ and a secret value $\mathfrak{sk}_A$ and returns the (full) private key S_A.
- $\mathbb{E}_{\texttt{CL}}(\mathfrak{pk}_A, \texttt{ID}_A, M_{\mathfrak{pk}}, m; r)$ is the PPT encryption algorithm. On input of a public key $\mathfrak{pk}_A$, an identifier $\texttt{ID}_A$, the master public key $M_{\mathfrak{pk}}$, a message $m \in \mathbb{M}_{\texttt{CL}}(M_{\mathfrak{pk}})$ and possibly some randomness $r \in \mathbb{R}_{\texttt{CL}}(M_{\mathfrak{pk}})$, this algorithm outputs a ciphertext $c \in \mathbb{C}_{\texttt{CL}}(M_{\mathfrak{pk}})$.
- $\mathbb{D}_{\texttt{CL}}(S_A, c)$ is the deterministic decryption algorithm. On input of a ciphertext c and the full private key S_A this algorithm outputs the corresponding value of the plaintext m or a failure symbol $\perp$.

Owing to the lack of authenticating information for public keys—certificates for example—an adversary may be able to replace users' public keys with public keys of its choice. This appears to give adversaries enormous power. However, to compute the full private key of a user, knowledge of the partial private key is necessary.

To capture the scenario above, Al-Riyami and Paterson [1–3] consider a security model in which an adversary is able to adaptively replace users' public keys with (valid) public keys of its choice. Such an adversary is called a Type-I adversary below.

Since the KGC is able to produce partial private keys, we must of course assume that the KGC does not replace users public keys itself. We do however treat other adversarial behavior of a KGC: eavesdropping on ciphertexts and making decryption queries for example. Such an adversarial KGC is referred to as a Type-II adversary below.

By assuming that a KGC does not replace users public keys itself, a user is placing a similar level of trust in a KGC that it would in a PKI certificate authority: it is always assumed that a CA does not issue certificates for individuals on public keys which it has maliciously generated itself!

Below we present a game to formally define what an adversary must do to break the scheme; X can be instantiated with I or II in the description below.

```
Type-X Adversarial Game
1. $(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}) \leftarrow \mathbb{G}_{\texttt{CL}}(1^t)$.
2. $(\texttt{ID}^*, s, m_0, m_1) \leftarrow A_1^{\mathcal{O}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}})$.
3. $b \leftarrow \{0,1\}$.
4. $c^* \leftarrow \mathbb{E}_{\texttt{CL}}(\mathfrak{pk}^*, \texttt{ID}^*, M_{\mathfrak{pk}}, m_b; r)$.
5. $b' \leftarrow A_2^{\mathcal{O}}(c^*, s)$.
```

In the above s is some state information, and m_0 and m_1 are messages of equal length. Note, that the master secret key $M_{\mathfrak{sk}}$ is only passed to the adversary in step 2 above in the case of Type-II adversaries.

When performing the encryption (step 4) in the game, the challenger uses the *current* public key $\mathfrak{pk}^*$ of the user with identifier $\texttt{ID}^*$. (Note that a Type-II adversary is unable to change users' public keys, so the notion of current public key is redundant.)

The adversary's advantage is defined to be

$$\mathrm{Adv}_{\mathtt{CL}}^{\mathtt{Type\text{-}X}}(A) = |2\Pr[b' = b] - 1|,$$

where X is either I, I^- or II (see below for definition of I^-). We now turn to the various oracles $\mathcal{O}$ available to the adversaries in each game.

Type-I Adversary Oracle Access: This adversary may request public keys, replace public keys with (valid) public keys of its choice, extract partial private and private keys, and make decryption queries for all identities of its choosing. We make natural restrictions on such a Type-I adversary: it is not allowed to do any of the following.

1. Extract the private key for $\mathtt{ID}^*$ at any point.
2. Request the private key for any identity if the corresponding public key has been replaced.
3. Replace the public key for $\mathtt{ID}^*$ before its challenge ciphertext has been issued *and* extract the partial private key for $\mathtt{ID}^*$ (at any point).
4. Once the challenge ciphertext c^* has been issued it must not make a decryption query on c^* under $\mathtt{ID}^*$ and the public key $\mathfrak{pk}^*$ used to encrypt m_b.

Type-I^- Adversary Oracle Access: This adversary is very similar to the Type-I adversary described above. The only difference is that, if it has replaced a public key and it subsequently requires a decryption query that involves a decryption with the corresponding secret key, it must supply this key to the decryption oracle. (Note that the decryption oracle continues to use $M_{\mathfrak{sk}}$ which is unknown to the adversary.) We propose this slightly weakened definition to allow us to prove our composition result and remark that, in any real life application, there could never be an oracle that performs decryption with an unknown secret key for an adversary.

Type-II Adversary Oracle Access: In this game the adversary has access to the master secret key $M_{\mathfrak{sk}}$ and so can create partial private keys itself. It is not allowed to replace public keys of entities at any point, but it can request public keys and make private key extraction queries for all entities of its choosing. However, it is not allowed to extract the private key for the challenge identity $\mathtt{ID}^*$ at any point. In addition, once the challenge ciphertext c^* has been issued, it cannot make a decryption query on c^* for the combination $(\mathfrak{pk}^*, \mathtt{ID}^*)$.

A certificateless system is said to be secure if, for all PPT Type-I *and* Type-II adversaries, the advantage in winning the relevant game is a negligible function of the security parameter. As mentioned above, to prove our composition result, we need to weaken the requirement of Type-I security and replace it with Type-I^-.

3.4. *Public-Key, Identity-Based and Certificateless Key Encapsulation Mechanisms*

Following Shoup [18] one can define the notion of *key encapsulation mechanisms* (KEMs) in the public-key setting as follows.

A standard KEM consists of three algorithms $(\mathbb{G}_{\mathtt{KEM}}, \mathbb{E}_{\mathtt{KEM}}, \mathbb{D}_{\mathtt{KEM}})$:

- $\mathbb{G}_{\mathtt{KEM}}(1^t)$ is the PPT key generation algorithm. This takes as input 1^t and outputs a public/private key pair $(\mathfrak{pk}, \mathfrak{sk})$.

- $\mathbb{E}_{\mathtt{KEM}}(\mathfrak{pk})$ is the PPT key encapsulation algorithm. This takes as input $\mathfrak{pk}$ and outputs an encapsulated key pair $(k, c) \in \mathbb{K}_{\mathtt{KEM}}(\mathfrak{pk}) \times \mathbb{C}_{\mathtt{KEM}}(\mathfrak{pk})$. The item c is called the encapsulation of the key k. The key k is assumed to be uniformly distributed over the key space $\mathbb{K}_{\mathtt{KEM}}(\mathfrak{pk})$.
- $\mathbb{D}_{\mathtt{KEM}}(\mathfrak{sk}, c)$ is the decapsulation algorithm. On input of $\mathfrak{sk}$ and c this outputs the corresponding value of k or an invalid encapsulation symbol $\perp$.

It is relatively straightforward to extend this definition to the identity-based and the certificateless settings. One can also easily extend the security notions of identity-based and certificateless schemes to the KEM setting. To save space we only perform this extension for certificateless schemes, the other simpler definitions can be derived by the reader.

A CL-KEM scheme is specified by seven polynomial time algorithms:

- $\mathbb{G}_{\mathtt{CL\text{-}KEM}}(1^t)$ is a PPT algorithm that takes as input 1^t and returns the master public keys $M_{\mathfrak{pk}}$ and the master secret key $M_{\mathfrak{sk}}$.
- `Partial-Private-Key-Extract` takes as input $M_{\mathfrak{sk}}$ and an identifier string for A, $\mathtt{ID}_A \in \{0, 1\}^*$ and returns a partial private key $D_{\mathtt{ID}_A}$. It may be deterministic or probabilistic.
- `Set-Secret-Value` is a PPT algorithm that takes no input (bar the system parameters) and outputs a secret value $\mathfrak{sk}_A$.
- `Set-Public-Key` is a deterministic algorithm that takes as input $\mathfrak{sk}_A$ and outputs a public key $\mathfrak{pk}_A$.
- `Set-Private-Key` is a deterministic algorithm that takes as input $D_{\mathtt{ID}_A}$ and $\mathfrak{sk}_A$ and returns S_A the (full) private key.
- $\mathbb{E}_{\mathtt{CL\text{-}KEM}}(\mathfrak{pk}_A, \mathtt{ID}_A, M_{\mathfrak{pk}})$ is the PPT encapsulation algorithm. On input of $\mathfrak{pk}_A$, $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$ this outputs a pair (k, c) where $k \in \mathbb{K}_{\mathtt{CL\text{-}KEM}}(M_{\mathfrak{pk}})$ is a key for the DEM and $c \in \mathbb{C}_{\mathtt{CL\text{-}KEM}}(M_{\mathfrak{pk}})$ is the encapsulation of that key.
- $\mathbb{D}_{\mathtt{CL\text{-}KEM}}(S_A, c)$ is the deterministic decapsulation algorithm. On input of c and S_A this outputs the corresponding k or a failure symbol $\perp$.

To define the security model for CL-KEMs we simply adapt the security model of Al-Riyami and Paterson into the KEM framework. Again there are three types of adversary against a CL-KEM: Type-I, Type-I$^-$ and Type-II. Each adversary is trying to win the following games, where the various oracle accesses allowed are identical to those defined in Section 3.3, we simply replace the word "decryption" with "decapsulation". Again the master secret $M_{\mathfrak{sk}}$ is only passed to the adversary in the case of Type-II adversaries.

```
Type-X Adversarial Game
1. $(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}) \leftarrow \mathbb{G}_{\mathtt{CL\text{-}KEM}}(1^t)$.
2. $(\mathtt{ID}^*, s) \leftarrow A_1^{\mathcal{O}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}})$.
3. $(k_0, c^*) \leftarrow \mathbb{E}_{\mathtt{CL\text{-}KEM}}(\mathfrak{pk}^*, \mathtt{ID}^*, M_{\mathfrak{pk}})$.
4. $k_1 \leftarrow \mathbb{K}_{\mathtt{CL\text{-}KEM}}(M_{\mathfrak{pk}})$.
5. $b \leftarrow \{0, 1\}$.
6. $b' \leftarrow A_2^{\mathcal{O}}(c^*, s, \mathtt{ID}^*, k_b)$.
```

When performing the encapsulation, in line three of both games, the challenger uses the *current* public key $\mathfrak{pk}^*$ of the entity with identifier $\mathtt{ID}^*$. The adversary's advantage in such a game is defined to be

$$\mathrm{Adv}_{\mathtt{CL\text{-}KEM}}^{\mathtt{Type\text{-}X}}(A) = |2\Pr[b' = b] - 1|$$

where X is either I, I$^-$ or II. A CL-KEM is considered to be secure, in the sense of IND-CCA2, if for all PPT adversaries A, the advantage is a negligible function of t.

4. Combining KEMs, ID-KEMs and CL-KEMs with DEMs

In order to apply our ID-KEM and CL-KEM constructions we will need to compose them with data encapsulation mechanisms. We shall show that combining a secure DEM with a fully secure KEM will result in a fully secure hybrid scheme. The security definition that we require of a DEM is a variant of the find-then-guess game as defined by Bellare et al. [5]. An adversary must determine which of two messages has been encrypted under a random key. After being given the challenge encryption, the adversary is given access to a decryption oracle for the key in question. We denote this notion FG-CCA henceforth.

This result will allow us to construct ID-IND-CCA2 secure IBE schemes from ID-IND-CCA2 secure ID-KEMs and FG-CCA secure DEMs. Similarly, it will allow us to construct certificateless encryption schemes secure against Type-I$^-$ and Type-II adversaries.

We assume that the key space output by the KEMs corresponds to the key space required by the DEM. Our construction follows that of Cramer and Shoup ([11], §7.3) and consists of the natural concatenation of the key encapsulation followed by the data encapsulation of the message under the key encapsulated by the first component. We denote such a ciphertext $C = (c_1, c_2)$ henceforth, where c_1 encapsulates the key and c_2 encapsulates the data. We refer to such a construction as *hybrid*.

The following theorem is a natural generalization of Theorem 5 of [11]. Note that, unlike the equivalent result in [11], we are implicitly assuming that, for all keys, all encapsulations decapsulate correctly. It would be straightforward to generalize the result. However, the soundness condition that we are assuming applies to all the primitives that we consider in this paper.

Theorem 1. *Let A be a PPT adversary against the hybrid IBE scheme* (*resp. the hybrid certificateless scheme*) *in the sense of ID-IND-CCA*2 (*resp. Type-I$^-$ and Type-II*) *adversaries*, *then there exists PPT adversaries B_1 and B_2*, *whose running times are essentially that of A*, *such that*

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID\text{-}IND\text{-}CCA}}(A) \le 2\mathrm{Adv}_{\mathtt{ID\text{-}KEM}}^{\mathtt{ID\text{-}IND\text{-}CCA}}(B_1) + \mathrm{Adv}_{\mathtt{DEM}}^{\mathtt{FG\text{-}CCA}}(B_2),$$

$$\mathrm{Adv}_{\mathtt{CL}}^{\mathtt{Type\text{-}I^-}}(A) \le 2\mathrm{Adv}_{\mathtt{CL\text{-}KEM}}^{\mathtt{Type\text{-}I^-}}(B_1) + \mathrm{Adv}_{\mathtt{DEM}}^{\mathtt{FG\text{-}CCA}}(B_2),$$

$$\mathrm{Adv}_{\mathtt{CL}}^{\mathtt{Type\text{-}II}}(A) \le 2\mathrm{Adv}_{\mathtt{CL\text{-}KEM}}^{\mathtt{Type\text{-}II}}(B_1) + \mathrm{Adv}_{\mathtt{DEM}}^{\mathtt{FG\text{-}CCA}}(B_2).$$

Before proceeding with the proof we note that the proof follows essentially that of [11]; the main difficulty occurs in Lemma 3. It is here that we must restrict ourselves to Type-I$^-$ adversaries in the certificateless setting. Since a Type-I secure CL-KEM is clearly Type-I$^-$ secure the above result allows us to combine a Type-I secure CL-KEM with a secure DEM, so as to obtain a Type-I$^-$ secure CL encryption scheme.

Proof. Our proof strategy is as follows. We define a sequence $\texttt{Game}_0$, $\texttt{Game}_1$, $\texttt{Game}_2$ of modified attack games in which A runs. The only difference between games is how the environment responds to A's oracle queries.

We fix some notation that we will use throughout. Let $C^* = (c_1^*, c_2^*)$ be the challenge ciphertext presented to A by its challenge encryption oracle—the oracle that encrypts either m_0 or m_1 according to a bit b. Let k^* denote the symmetric key used by the challenge encryption oracle in the generation of the challenge ciphertext, or alternatively, the decapsulation of c_1^* using the secret keys associated to $\texttt{ID}^*$—the identity chosen by the adversary on which it wishes to be challenged. For any $i = 0, 1, 2$, we let $\texttt{S}_i$ be the event that $b' = b$ in game $\texttt{Game}_i$, where b is the bit chosen by A's challenge encryption oracle. This probability is taken over the random choices of A and those of A's oracles.

Let $\texttt{Game}_0$ be the genuine attack game played by A. So by definition we have

$$|\Pr[\texttt{S}_0] - 1/2| = \frac{1}{2}\mathrm{Adv}_{*}^{\mathrm{PMOD}}(A).$$

Game $\texttt{Game}_0$ is now modified so that whenever an identity $\texttt{ID}$ and (c_1, c_2) is presented to the decryption oracle after the invocation of the challenge encryption oracle, if $\texttt{ID} = \texttt{ID}^*$ and $c_1 = c_1^*$, and in the case of a Type-I$^-$ adversary, the public key of $\texttt{ID}^*$ has not been replaced, then the decryption oracle does not use the genuine decryption procedure for the hybrid scheme, instead it uses the key k^* to decapsulate c_2 and returns the result to the adversary. This modification to $\texttt{Game}_0$ gives us the game $\texttt{Game}_1$. Games $\texttt{Game}_0$ and $\texttt{Game}_1$ are identical, under the soundness condition that we discussed above, and so $\Pr[\texttt{S}_1] = \Pr[\texttt{S}_0]$.

We now modify $\texttt{Game}_1$ by replacing k^* with a random key k' from $\mathbb{K}_{\texttt{DEM}}(t_1, t_2)$. With this modification we have the game $\texttt{Game}_2$. The result then follows from the following two lemmas. □

Lemma 2. *There is a PPT algorithm B_1, whose running time is essentially the same as that of A, such that*

$$|\Pr[\texttt{S}_2] - \Pr[\texttt{S}_1]| = \mathrm{Adv}_{*\text{-}\mathrm{KEM}}^{\mathrm{PMOD}}(B_1),$$

where MOD is IND-CCA2, Type-I$^-$ or Type-II and $*$ is ID or CL as appropriate.

Proof. To prove this we demonstrate how to construct an adversary B_1 of the KEM to violate the assumed security against adaptive chosen ciphertext (resp. Type-I$^-$/Type-II) attack.

Adversary B_1 is constructed by running adversary A. We respond to A's queries as follows.

– When A calls any oracle, bar its decryption or challenge encryption oracles, B_1 simply relays these queries to its own equivalent oracle.

- To respond to A's decryption oracle query for an identity $\mathtt{ID}$ and a ciphertext (c_1, c_2) before A has queried its challenge encryption oracle, B_1 proceeds as follows. It first obtains k by calling its own decapsulation oracle with c_1. If $k = \perp$ then B_1 replies to A with $\perp$. Otherwise it proceeds to use k to decrypt c_2 and relays the result to A.
- When A calls its challenge encryption oracle with identity $\mathtt{ID}^*$ and messages (m_0, m_1), B_1 first calls its own challenge encryption oracle with $\mathtt{ID}^*$ to obtain $(k^\dagger, c_1^*)$. It then chooses a bit d at random and computes $c_2^* \leftarrow \mathbb{E}_{\mathtt{DEM}}(k^\dagger, m_d)$. Finally, it responds to A with (c_1^*, c_2^*).
- To respond to A's decryption oracle query for an identity $\mathtt{ID}$ and a ciphertext (c_1, c_2) after A has queried its challenge encryption oracle, B_1 proceeds as follows.
  - If $(\mathtt{ID}, c_1) \neq (\mathtt{ID}^*, c_1^*)$ then it uses the same procedure that it used before A's call to its challenge encryption oracle.
  - In the case of a Type-I$^-$ adversary against a certificateless encryption scheme, if $(\mathtt{ID}, c_1) = (\mathtt{ID}^*, c_1^*)$ and the public key has been replaced, then B_1 responds by calling the decapsulation oracle provided to it by A with input $(\mathtt{ID}^*, c_1^*)$ to obtain k. It then uses k to decrypt c_2 and relays the response to A.
  - Otherwise, B_1 uses $k^\dagger$ to decrypt c_2 and relays the result to A.

At the end of the simulation, A outputs a bit d'. If $d' = d$, B_1 outputs 1, otherwise it outputs 0.

Let b be the internal bit of B_1's challenge oracle which B_1 seeks to determine and let b' be the bit output by B_1. By construction we see that when $b = 1$, so $k^\dagger$ is the key encapsulated within c_1^*, A is run exactly as it would be run in $\mathtt{Game}_1$. This means that

$$\Pr[\mathtt{S}_1] = \Pr[d' = d | b = 1] = \Pr[b' = 1 | b = 1], \tag{2}$$

where d is A's challenge bit and d' is A's guess. Also, when $b = 0$, so a random k' is used in the generation of the challenge ciphertext, A is run exactly as it would be in $\mathtt{Game}_2$. This means that

$$\Pr[\mathtt{S}_2] = \Pr[d' = d | b = 0] = \Pr[b' = 1 | b = 0]. \tag{3}$$

The result follows from (2), (3) and the definitions of security for KEMs when one observes that

$$\mathrm{Adv}^{\mathtt{PMOD}}_{\mathtt{*\text{-}KEM}}(B_1) = |2\Pr[b' = b] - 1| = |\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]|. \quad \square$$

Lemma 3. *There is a PPT algorithm B_2, whose running time is essentially the same as that of A, such that*

$$|\Pr[\mathtt{S}_2] - 1/2| = \frac{1}{2}\mathrm{Adv}^{\mathtt{FG\text{-}CCA}}_{\mathtt{DEM}}(B_2).$$

Proof. To construct such a B_2 we simply run A as it would be run in game $\mathtt{Game}_2$. We run the ID/CL-KEM's key generation step so we can respond to A's queries before it calls its challenge encryption oracle. When A calls its challenge encryption oracle with identity $\mathtt{ID}^*$ and messages (m_0, m_1) we simply relay (m_0, m_1) to the challenge encryption oracle of B_2 to obtain c_2^*. We then run the key encapsulation mechanism to obtain (k, c_1) we discard k and set $c_1^* = c_1$. Finally we return (c_1^*, c_2^*) to A. We continue to respond to A's queries as before except if it a makes decryption query $\mathtt{ID}^*$, (c_1^*, c_2) for some c_2. In this instance there are two cases:

– If we are dealing with a Type-I$^-$ adversary A of a certificateless encryption scheme, and the public key of $\mathtt{ID}^*$ has been replaced, then B_2 decapsulates (ID^*, c_1^*) using the provided secret key to obtain k, decrypts c_2 and relays the response to A.
– Otherwise we query B_2's decryption oracle with c_2 and relay the response to A.

In this simulation A is run by B_2 in exactly the same manner as the former would be run in game $\mathtt{Game}_2$; moreover, $\Pr[\mathtt{S}_2]$ corresponds exactly to the probability that B_2 correctly determines the hidden bit of its challenge encryption oracle since B_2 outputs whatever A outputs. The result follows. □

5. ID-KEM Constructions

In this section we describe our generic construction of a fully secure identity-based KEM from an ID-OW-CPA secure IBE scheme. We let H_1 and H_2 denote cryptographic hash functions, with the following domains and co-domains

$$H_1 : \{0,1\}^* \to \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}}), \quad \text{and}$$
$$H_2 : \{0,1\}^* \to \mathbb{K}_{\mathtt{DEM}}(t)$$

where $\mathbb{K}_{\mathtt{DEM}}(t)$ is the key space of the DEM we will be using.

We take a generic probabilistic IBE scheme, we shall denote the associated encryption algorithm by $\mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; r)$ and the associated decryption algorithm by $\mathbb{D}_{\mathtt{ID}}(D_{\mathtt{ID}}, c)$, where $D_{\mathtt{ID}}$ is the output from the extraction algorithm $\mathbb{X}_{\mathtt{ID\text{-}KEM}}(M_{\mathfrak{sk}}, \mathtt{ID})$. We assume the message space of $\mathbb{E}_{\mathtt{ID}}$ is finite and is given by $\mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$ and the space of randomness is given by $\mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.

In the following construction we make no assumption on how such an identity-based scheme is constructed, only that it exists.

```
$\mathbb{E}_{\mathtt{ID\text{-}KEM}}(\mathtt{ID}, M_{\mathfrak{pk}})$
  – $m \leftarrow \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.
  – $r \leftarrow H_1(m)$.
  – $c \leftarrow \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; r)$.
  – $k \leftarrow H_2(m)$.
  – Return $(k, c)$.

$\mathbb{D}_{\mathtt{ID\text{-}KEM}}(D_{\mathtt{ID}}, c)$
  – $m \leftarrow \mathbb{D}_{\mathtt{ID}}(D_{\mathtt{ID}}, c)$.
  – If $m = \perp$ then return $\perp$.
  – $r \leftarrow H_1(m)$.
  – If $c \neq \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; r)$ then return $\perp$.
  – $k \leftarrow H_2(m)$.
  – Return $k$.
```

Before proceeding we compare our construction with the FullIdent scheme of Boneh and Franklin [9] and the Fujisaki–Okamoto transformation that it uses [13]. Our first remark is that we can instantiate the ID-OW-CPA scheme in our construction using the BasicIdent scheme described by Boneh and Franklin [9] as it has the security property that we require. The computational requirements for our construction and FullIdent are then identical. However, we believe that our approach has considerable advantages.

Firstly, as an artifact of the Fujisaki–Okamoto transformation, the random input for the FullIdent scheme depends on the actual message being encrypted. In particular, this means that the expensive elliptic curve operations can only be performed once one knows what message will be encrypted. Our method, using an ID-KEM, has the property that a key can be encapsulated at any point. This could be of particular benefit if one is going to send the same message to multiple parties, a problem studied in [4].

The second advantage of our approach is also related to the use of the Fujisaki–Okamoto transformation. A popular application for IBE is encrypted email and one very popular form of email is web-based email for roaming users. Suppose that we wanted IBE encryption for web-based email. In this instance it would make sense for user secret keys to be stored in a secure server. Using our construction, since the KEM and the DEM are independent, the user can send the (relatively short) KEM to the server for decryption without having to send the entire message. This is not the case for any scheme derived from the Fujisaki–Okamoto transform because there the actual message is necessary for integrity checking.

The final point to note is that our construction makes use of an IBE scheme that is ID-OW-CPA (such as BasicIdent from [9]). We remark that this scheme is also ID-IND-CPA.

Theorem 2. *If $\mathbb{E}_{\mathtt{ID}}$ is an ID-OW-CPA secure IBE scheme and H_1 and H_2 are modeled as random oracles then the above construction of an ID-KEM is secure against adaptive chosen ciphertext attack.*

Specifically, if A is a PPT algorithm that breaks the ID-KEM by using a chosen ciphertext attack, then there exists a PPT algorithm B, with

$$\mathrm{Adv}_{\mathtt{ID\text{-}KEM}}^{\mathtt{ID\text{-}IND\text{-}CCA2}}(A) \leq 2(q_1 + q_2 + q_D) \cdot \mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID\text{-}OW\text{-}CPA}}(B) + \frac{2q_D\gamma(M_{\mathfrak{pk}})}{|\mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})|},$$

where q_1, q_2 and q_D are the number of queries made by A to H_1, H_2 and the decryption oracle respectively, and $\gamma(M_{\mathfrak{pk}})$ is as in (1).

Proof. The proof of this result follows by adapting the proof of Theorem 5 of [12] to the identity-based setting. The only essential difference is that one needs to answer the key extraction queries of the adversary A, however this is done by appealing to the key extraction oracle provided to B. □

6. CL-KEM Construction

In this section we give our generic construction of a CL-KEM.

- Let $(\mathbb{G}_{\mathtt{PK}}, \mathbb{E}_{\mathtt{PK}}, \mathbb{D}_{\mathtt{PK}})$ be a OW-CPA^{++} secure public-key encryption algorithm that is verifiable, for example textbook RSA.
- Let $(\mathbb{G}_{\mathtt{ID}}, \mathbb{X}_{\mathtt{ID}}, \mathbb{E}_{\mathtt{ID}}, \mathbb{D}_{\mathtt{ID}})$ be an ID-OW-CCA2 secure IBE algorithm.

We define our seven algorithms as follows. The algorithm $\mathbb{G}_{\mathtt{CL\text{-}KEM}}$ is defined to be equal to $\mathbb{G}_{\mathtt{ID}}$. The algorithm `Partial-Private-Key-Extract` returns $D_{\mathtt{ID}_A}$: the output from $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{sk}}, \mathtt{ID}_A)$. The values returned by `Set-Secret-Value` and `Set-Public-Key` are simply the outputs $\mathfrak{pk}_A$ and $\mathfrak{sk}_A$ from $\mathbb{G}_{\mathrm{PK}}$. The algorithm `Set-Private-Key` returns the pair $S_A = (D_{\mathtt{ID}_A}, \mathfrak{sk}_A)$. Finally, using a hash function H, encapsulation and decapsulation work as follows.

```
E_CL-KEM(pk_A, ID_A, M_pk):                D_CL-KEM(S_A, c):
– $m_1 \leftarrow \mathbb{M}_{\mathrm{PK}}(\mathfrak{pk})$.                     – $(c_1, c_2) \leftarrow c$.
– $r_1 \leftarrow \mathbb{R}_{\mathrm{PK}}(\mathfrak{pk})$.                     – $(D_{\mathtt{ID}_A}, \mathfrak{sk}_A) \leftarrow S_A$.
– $m_2 \leftarrow \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.                  – $m_1 \leftarrow \mathbb{D}_{\mathrm{PK}}(\mathfrak{sk}_A, c_1)$.
– $r_2 \leftarrow \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.                  – If $m_1 = \perp$ then return $\perp$.
– $c_1 \leftarrow \mathbb{E}_{\mathrm{PK}}(\mathfrak{pk}_A, m_1; r_1)$.          – $m_2 \leftarrow \mathbb{D}_{\mathtt{ID}}(D_{\mathtt{ID}_A}, c_2)$.
– $c_2 \leftarrow \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}_A, M_{\mathfrak{pk}}, m_2; r_2)$.  – If $m_2 = \perp$ then return $\perp$.
– $k \leftarrow H(c_1, \mathfrak{pk}_A, m_1, m_2)$.        – $k \leftarrow H(c_1, \mathfrak{pk}_A, m_1, m_2)$.
– $c \leftarrow (c_1, c_2)$.
```

Note that we require our IBE scheme in Theorem 3 below to be ID-OW-CCA2 secure. We use this rather strong requirement to simplify our presentation and note that an ID-OW-CPA scheme can be enhanced to offer ID-OW-CCA2 security by making the random input the hash of the message. This is done at the cost of a re-encryption during verification. The proof for such a construction is almost identical to the proof of Theorem 2, the only difference being the definition of the adversarial goal for the algorithm A.

Before stating our results, we compare the performance of our construction with that of Al-Riyami and Paterson [3]. The major advantage of our approach is that we decouple the identity based part of the scheme from the public-key part of the scheme. This means that our scheme can be used within an existing PKI. This could also lead to performance advantages. Suppose that we implement the public-key part of the scheme with raw RSA (which satisfies our security requirements) and the scheme derived from BasicIdent [9] by making the random input the hash of the message (as discussed above). This would mean that in encryption we replace two pairing computations with one RSA encryption operation. The only overhead is an extra RSA decryption operation in the decryption phase.

We also remark that, unlike our approach, the scheme of [3] makes use of the Fujisaki–Okamoto transformation [13] and so the comments that we make in Section 5 also apply here.

6.1. *Security Proof: Type-I Adversary*

In this section we shall prove that Type-I security of our generic CL-KEM construction rests both on the security of the identity-based component of the scheme and on the security of the public-key component.

Theorem 3. *Our generic CL-KEM construction is secure against Type-I adversaries in the random oracle model, assuming the IBE scheme is secure in the sense of ID-OW-CCA2 and the public-key scheme is secure in the sense of OW-CPA^{++}.*

In particular, let A denote a PPT Type-I adversary A against our generic CL-KEM which makes at most q_H calls to the random oracle H, requests up to q_{PK} public keys, makes at most q_R replacements of public keys, makes at most q_{SK} private key extractions, makes at most q_{PX} partial-private key extractions and at most q_D decapsulation *queries. These queries are subject to the restrictions imposed in the Type-I game definition given above.*

Then there exists two PPT adversaries: B_1 *against the ID-OW-CCA*2 *security of the IBE system which makes at most* $q_H + q_D$ *calls to the random oracle* H, *at most* q_D *calls to its decryption oracle and at most* $q_{PX} + q_{SK}$ *calls to its private key extraction*; *and* B_2 *against the OW-CPA*$^{++}$ *security of the public-key scheme that makes at most* $q_H + q_D$ *calls to the random oracle* H. *Both* B_1 *and* B_2 *run for essentially the same time as* A *and they are such that*

$$\mathrm{Adv}^{\texttt{Type-I}}_{\texttt{CL-KEM}}(A) \le 2(q_H + q_D)\mathrm{Adv}^{\texttt{ID-OW-CCA2}}_{\texttt{ID}}(B_1) + 2(q_{PK} + q_D + 1)\mathrm{Adv}^{\texttt{OW-CPA}^{++}}_{\texttt{PK}}(B_2).$$

Proof. Let A denote a Type-I adversary against our CL-KEM as specified in the statement of the theorem.

We let $\texttt{ID}^*$ denote the challenge identity chosen by A after its first stage. We shall denote the target encapsulation by $c^* = (c_1^*, c_2^*)$ which encapsulates the key k_1^*. Let m_1^* denote the message encrypted in c_1^* under $\mathfrak{pk}^*$, the public key of $\texttt{ID}^*$ at the time the challenge ciphertext is created, and let m_2^* denote the message encrypted in c_2^* under $\texttt{ID}^*$. Let k_0^* denote a random key selected from the co-domain of H, and let b denote a random bit; both outside the view of the adversary. In the second stage of the game we let $k^* = k_b^*$ denote the key given to the adversary and we let b' denote the bit returned by the adversary.

Security is proved using two games $\texttt{Game}_0$ and $\texttt{Game}_1$. In each game $\texttt{Game}_i$ we let $\texttt{S}_i$ denote the event that $b' = b$. We let $\texttt{Game}_0$ denote the original attack game and so by definition

$$\mathrm{Adv}^{\texttt{Type-I}}_{\texttt{CL-KEM}}(A) = |2\Pr[\texttt{S}_0] - 1|. \tag{4}$$

$\texttt{Game}_1$: In $\texttt{Game}_1$ we replace the public key request oracles, the private key extraction oracles and the hash function H by the following oracle simulations.

- **Public Key Request/Private Key Extraction:** We keep a list $L_X = \{(\texttt{ID}, \mathfrak{pk}, \mathfrak{sk})\}$ of length at most $q_{PK} + q_{SK} + q_{PX} + q_R + q_D$. When either oracle is called on an identity $\texttt{ID}$ we check whether this identity already appears on the list, if so we respond with either $\mathfrak{pk}$ or $\mathfrak{sk}$ as appropriate. Otherwise we call $\mathbb{G}_{\mathfrak{pk}}$ to obtain a new pair $(\mathfrak{pk}, \mathfrak{sk})$ insert $(\texttt{ID}, \mathfrak{pk}, \mathfrak{sk})$ into the list and then return the appropriate value of $\mathfrak{pk}$ or $\mathfrak{sk}$.
- **Public Key Replacement:** If A wishes to replace the public key for user $\texttt{ID}$ with $\mathfrak{pk}'$ then we search the L_X list for an entry corresponding to $\texttt{ID}$ and replace this entry with $(\texttt{ID}, \mathfrak{pk}', \perp)$. If no such entry exists then we add $(\texttt{ID}, \mathfrak{pk}', \perp)$ to the list L_X.
- **Partial Private Key Extraction:** The challenger answers these queries using the genuine partial private key extraction algorithm.
- **Hash Function:** We keep a list $L_H = \{(k, c_1, \mathfrak{pk}, m_1, m_2)\}$ of length at most $q_H + q_D$. If this oracle is called with input $(c_1, \mathfrak{pk}, m_1, m_2)$ we perform the following steps:
  - If $(k, c_1, \mathfrak{pk}, m_1, m_2)$ is on L_H then we respond with k.
  - If there is some $(k, c_1, \mathfrak{pk}, \perp, m_2)$ on L_H such that c_1 is the encryption of m_1 under the key $\mathfrak{pk}$ we update this entry to make it $(k, c_1, \mathfrak{pk}, m_1, m_2)$ and we

respond with k. Note, this requires the property that the public-key scheme is verifiable.

- Otherwise we generate k at random from the co-domain of H, we place $(k, c_1, \mathfrak{pk}, m_1, m_2)$ onto L_H and respond with k.

– **Decapsulation Queries:** On input of $c = (c_1, c_2)$ and $\mathtt{ID}$, the simulator for algorithm A can decapsulate c_2 to obtain m_2, since it knows the master key for the identity-based scheme. Then, by making a call to the public key extraction oracle we can obtain the public/private key pair $(\mathfrak{pk}, \mathfrak{sk})$ from the list L_X corresponding to the identity $\mathtt{ID}$. There are two cases:

- $\mathfrak{sk} \neq \perp$, in which case the public key has not been replaced. We can then use $\mathfrak{sk}$ to decrypt c_1 to obtain m_1. The simulator for hash function H can then be called on the input $(c_1, \mathfrak{pk}, m_1, m_2)$ so as to obtain the encapsulated key k.
- $\mathfrak{sk} = \perp$, in which case the public key has been replaced. We then search L_H to find an entry $(k, c_1, \mathfrak{pk}, m_1, m_2)$ such that c_1 is the encryption of m_1 under the key $\mathfrak{pk}$. Note, this requires the property that the public-key scheme is verifiable. If such an entry exists then we return k. Otherwise we generate k at random from the co-domain of H, place $(k, c_1, \mathfrak{pk}, \perp, m_2)$ onto L_H and return k.

Since A is working in the random oracle model then the two games are identical. We have

$$\Pr[\mathtt{S}_0] = \Pr[\mathtt{S}_1]. \tag{5}$$

Before proceeding, we define three events.

`Replace`: The event that A replaces the public key for $\mathtt{ID}^*$ before the challenge ciphertext is issued.
`Extract`: The event that A extracts the partial private key for $\mathtt{ID}^*$.
`Ask`: The event that the simulator for H is called with input $(c_1^*, \mathfrak{pk}^*, m_1^*, m_2^*)$.

We immediately have the following.

$$\begin{aligned}\Pr[\mathtt{S}_1] &= \Pr[\mathtt{S}_1 \wedge \mathtt{Replace}] + \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Replace}] \\ &= \Pr[\mathtt{S}_1|\mathtt{Replace}]\Pr[\mathtt{Replace}] \\ &\quad + \Pr[\mathtt{S}_1|\neg\mathtt{Replace}](1 - \Pr[\mathtt{Replace}]). \end{aligned} \tag{6}$$

Also,

$$\Pr[\mathtt{S}_1|\mathtt{Replace}] = \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Extract}|\mathtt{Replace}]. \tag{7}$$

The last equality above follows from the fact that, by definition of a Type-I adversary, if `Replace` occurs then `Extract` is forbidden.

Now,

$$\begin{aligned}\Pr[\mathtt{S}_1 \wedge \neg\mathtt{Extract}|\mathtt{Replace}] &= \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Extract} \wedge \mathtt{Ask}|\mathtt{Replace}] \\ &\quad + \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Extract} \wedge \neg\mathtt{Ask}|\mathtt{Replace}] \\ &\leq \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Extract} \wedge \mathtt{Ask}|\mathtt{Replace}] + \frac{1}{2}. \end{aligned} \tag{8}$$

The final inequality follows from the fact that, if the query $(c_1^*, \mathfrak{pk}^*, m_1^*, m_2^*)$ is never made to the simulator for H, then A can have no advantage.
We also have

$$\begin{aligned}\Pr[\mathtt{S}_1|\neg\mathtt{Replace}] &= \Pr[\mathtt{S}_1 \wedge \mathtt{Ask}|\neg\mathtt{Replace}] + \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Ask}|\neg\mathtt{Replace}] \\ &\le \Pr[\mathtt{S}_1 \wedge \mathtt{Ask}|\neg\mathtt{Replace}] + \frac{1}{2}. \end{aligned} \tag{9}$$

Again, the last inequality follows from the fact that, if the query $(c_1^*, \mathfrak{pk}^*, m_1^*, m_2^*)$ is never made to the simulator for H, then A can have no advantage.

We now describe an algorithm B_1 to break the assumed ID-OW-CCA2 security of the IBE scheme used in the construction. This algorithm runs A in a similar manner to how A is run in $\mathtt{Game}_1$. The first difference is how we respond to A's decapsulation queries in the construction of B_1. To do this we must introduce an additional list L'_H. This list is initially empty, it is updated by the new decapsulation oracle as described below.

- **Decapsulation Queries:** Suppose that we are responding to a query $\mathtt{ID}$, (c_1, c_2). If $\mathtt{ID} \neq \mathtt{ID}^*$ or $c_2 \neq c_2^*$ we respond as in $\mathtt{Game}_1$ except that, rather than using knowledge of the master key for the identity-based scheme which we no longer have, we decapsulate c_2 using the decapsulation oracle provided to B_1. Otherwise, we make a call to the public key request oracle to obtain the public key $\mathfrak{pk}$ from the list L_X corresponding to the identity $\mathtt{ID}$. We then search L'_H for an entry $(k, c_1, \mathfrak{pk})$. If such exists we respond with k. Otherwise we choose k at random from the co-domain of H, add $(k, c_1, \mathfrak{pk})$ to L'_H and respond with k.

To generate the challenge ciphertext for A we proceed as usual to compute c_1^*, we obtain c_2^* by relaying $\mathtt{ID}^*$ output by A to B_1's challenge oracle and we choose k^* at random from the co-domain of H.

Finally, at the end of A's execution, we choose a random input $(c_1, \mathfrak{pk}, m_1, m_2)$ from L_H and output m_2 as B_1's attempt to recover the plaintext within c_2^*.

Now, A is run by B_1 up until the event $\mathtt{Ask}$ occurs in exactly the same manner as A is run in $\mathtt{Game}_1$; moreover, if the event $\mathtt{Ask}$ occurs, B_1 succeeds to recover the plaintext within c_2^* with probability at least $1/(q_H + q_D)$ since there are at most $(q_H + q_D)$ entries in L_H. This tells us that

$$\Pr[\mathtt{S}_1 \wedge \neg\mathtt{Extract} \wedge \mathtt{Ask}|\mathtt{Replace}] \le (q_H + q_D)\mathrm{Adv}_{\mathtt{ID}}^{\text{ID-OW-CCA2}}(B_1). \tag{10}$$

To complete the proof we describe an adversary B_2 of the public-key scheme used in our construction. The adversary B_2 is given a public key $\mathfrak{pk}^*$ for which it wishes to recover a message from a ciphertext. To construct B_2 we run A in similar manner to how A is run in $\mathtt{Game}_1$. The oracles that are modified are described below.

- **Public Key Request/Private Key Extraction:** At the very beginning of the simulation we choose i uniformly at random from $[1, \ldots, q_{PK} + q_D + 1]$. We maintain a list $L_X = \{(\mathtt{ID}, \mathfrak{pk}, \mathfrak{sk})\}$ of length at most $q_{PK} + q_{SK} + q_{PX} + q_R + q_D + 1$. We have two cases when responding to a query $\mathtt{ID}$.
  - If we are responding to the i-th public key request, made either by A directly or by the decapsulation oracle, or made by the challenge encryption oracle, we respond with $\mathfrak{pk}^*$ and add $(\mathtt{ID}, \mathfrak{pk}^*, \perp)$ to L_X.

  - Otherwise, we check whether this identity already appears on the list, if so we respond with either $\mathfrak{pk}$ or $\mathfrak{sk}$ as appropriate and, if not, we call $\mathbb{G}_{\mathfrak{pk}}$ to obtain a new pair $(\mathfrak{pk}, \mathfrak{sk})$ insert $(\texttt{ID}, \mathfrak{pk}, \mathfrak{sk})$ into the list and then return the appropriate value of $\mathfrak{pk}$ or $\mathfrak{sk}$.
- **Hash Function:** We modify how the hash function operates after the challenge ciphertext has been issued. Suppose that we are responding to a query $(c_1^*, \mathfrak{pk}^*, m_1, m_2)$—where c_1^* is the first component of the challenge encapsulation—before proceeding as in $\texttt{Game}_1$ we check whether or not c_1^* is the encryption under $\mathfrak{pk}^*$ of m_1. If so we output m_1 and terminate the simulation.

To generate the challenge ciphertext for A first call the public key request oracle. If we do not receive $\mathfrak{pk}^*$ in response we abort the simulation. If we do receive $\mathfrak{pk}^*$ we proceed as usual to compute c_2^*, we obtain c_1^* by calling B_2's challenge encryption and k^* at random from the co-domain of H.

Now, when we are generating the challenge ciphertext we obtain $\mathfrak{pk}^*$ from the public key request oracle with probability at least $1/(q_{PK} + q_D + 1)$. Assuming this is so, A is run by B_2 in exactly the same way that A is run in $\texttt{Game}_1$ up until the event $\texttt{Ask}$ occurs and, moreover, if the event $\texttt{Ask}$ occurs, B_2 succeeds. We conclude that

$$\Pr[\texttt{S}_1 \wedge \texttt{Ask} | \neg\texttt{Replace}] \le (q_{PK} + q_D + 1)\mathrm{Adv}_{\texttt{PK}}^{\texttt{OW-CPA}^{++}}(B_2). \tag{11}$$

The result now follows from (4), (5), (6), (7), (8), (10), (9) and (11). □

6.2. *Security Proof: Type-II Adversary*

In this section we shall prove that Type-II security of our generic CL-KEM construction rests solely on the security of the public-key component of the scheme.

Theorem 4. *Our generic CL-KEM construction is secure against Type-II adversaries in the random oracle model, assuming the public-key encryption system is secure in the sense of OW-CPA*$^{++}$.

In particular, let A denote a PPT Type-II adversaries A against our generic CL-KEM which makes at most q_H calls to the random oracle H, at most q_{SK} calls to its private key extraction oracle, it may request up to q_{PK} public keys and make at most q_D decapsulation queries all for identities of its choice, subject to the usual restrictions.

Then there exists a PPT adversary B_2 against the OW-CPA^{++} *security of the public-key system, whose running time is essentially the same as that of A and which makes at most $q_H + q_D$ calls to the random oracle H, such that we have*

$$\mathrm{Adv}_{\texttt{CL-KEM}}^{\texttt{Type-II}}(A) \le 2(q_H + q_D)(q_{PK} + q_{SK} + q_D)\mathrm{Adv}_{\texttt{PK}}^{\texttt{OW-CPA}^{++}}(B_2).$$

Proof. Let A denote a Type-II adversary against our CL-KEM as specified in the statement of the theorem.

Security is proved via three games $\texttt{Game}_0$, $\texttt{Game}_1$ and $\texttt{Game}_2$. We define $\texttt{ID}^*$, $\mathfrak{pk}^*$, $c^* = (c_1^*, c_2^*)$, m_1^*, m_2^*, k_0^*, k_1^*, b, b', $\texttt{S}_i$ and $\texttt{Ask}$ exactly as in Theorem 3.

We let $\texttt{Game}_0$ denote the original attack game and so

$$\mathrm{Adv}_{\texttt{CL-KEM}}^{\texttt{Type-II}}(A) = |2\Pr[\texttt{S}_0] - 1|. \tag{12}$$

$\mathtt{Game}_1$: In $\mathtt{Game}_1$ we replace the public key request oracles, the private key extraction oracles and the hash function H by the following oracle simulations.

– **Public Key Request/Private Key Extraction:** We keep a list $L_X = \{(\mathtt{ID}, \mathfrak{pk}, \mathfrak{sk})\}$ of length at most $q_{PK} + q_{SK} + q_D$. When either oracle is called on an identity $\mathtt{ID}$ we check whether this identity already appears on the list, if so we respond with either $\mathfrak{pk}$ or $\mathfrak{sk}$ as appropriate. Otherwise we call $\mathbb{G}_{\mathfrak{pk}}$ to obtain a new pair $(\mathfrak{pk}, \mathfrak{sk})$ insert $(\mathtt{ID}, \mathfrak{pk}, \mathfrak{sk})$ into the list and then return the appropriate value of $\mathfrak{pk}$ or $\mathfrak{sk}$.
– **Hash Function:** We keep a list $L_H = \{(k, c_1, \mathfrak{pk}, m_1, m_2)\}$ of length at most $q_H + q_D$. If this oracle is called with input $(c, \mathfrak{pk}, m_1, m_2)$ we see whether this pair already appears on the list, if so we respond with the appropriate value of k. Otherwise we generate k at random from the co-domain of H, we place $(k, c, \mathfrak{pk}, m_1, m_2)$ into the list and return k.
– **Decapsulation Queries:** On input of $c = (c_1, c_2)$ and $\mathtt{ID}$, the simulator for algorithm A can decapsulate c_2 to obtain m_2, since it knows the master key for the identity-based scheme. Then, by making a call to the public key extraction oracle we can obtain the public/private key pair $(\mathfrak{pk}, \mathfrak{sk})$ from the list L_X corresponding to the identity $\mathtt{ID}$. Using $\mathfrak{sk}$ we can then decrypt c_1 to obtain m_1. The hash function H can then be called on the input $(c, \mathfrak{pk}, m_1, m_2)$ so as to obtain the encapsulated key k, modifying the list L_H as above.

Since A is working in the random oracle model then the two games are identical. We have

$$\begin{aligned}\Pr[\mathtt{S}_0] &= \Pr[\mathtt{S}_1] = \Pr[\mathtt{S}_1 \wedge \mathtt{Ask}] + \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Ask}] \\ &\le \Pr[\mathtt{S}_1|\mathtt{Ask}] + \Pr[\mathtt{S}_1|\neg\mathtt{Ask}] \\ &\le \Pr[\mathtt{S}_1|\mathtt{Ask}] + \frac{1}{2}. \end{aligned} \tag{13}$$

This last equality holds since H is a random oracle; if A does not make the critical query then it is able to determine whether or not k_b^* is the encapsulated key with probability at most $1/2$.

$\mathtt{Game}_2$: In this game a random value j is chosen from $[1, \ldots, q_{PK} + q_{SK} + q_D]$. Without loss of generality we can assume that the adversary makes the call to the public key request oracle for the challenge identity $\mathtt{ID}^*$. In $\mathtt{Game}_2$ we abort the game if the j-th element of the L_X list is not on the identity $\mathtt{ID}^*$. Let $\mathtt{F}_2$ denote the event that $\mathtt{Game}_2$ does not abort, then clearly $\Pr[\mathtt{F}_2] \ge 1/(q_{PK} + q_{SK} + q_D)$. In addition we have $\Pr[\mathtt{S}_2|\mathtt{Ask} \wedge \mathtt{F}_2] = \Pr[\mathtt{S}_1|\mathtt{Ask}]$. Which gives us

$$\Pr[\mathtt{S}_2|\mathtt{Ask}] = \Pr[\mathtt{S}_1|\mathtt{Ask}] \cdot \Pr[\mathtt{F}_2] \ge \frac{\Pr[\mathtt{S}_1|\mathtt{Ask}]}{q_{PK} + q_{SK} + q_D}.$$

We claim that $\Pr[\mathtt{S}_2|\mathtt{Ask}] = (q_H + q_D)\mathrm{Adv}_{\mathrm{PK}}^{\mathtt{OW\text{-}CPA}^{++}}(B_2)$ for an algorithm B_2. Algorithm B_2 takes as input a public key $\mathfrak{pk}^*$, it has access to a challenge oracle $\mathcal{O}_{\mathbb{E}_{\mathrm{PK}}}()$, which it can call only once. The challenge oracle will produce a ciphertext c_1^*, and the goal of B_2 is to deduce the corresponding value of m_1^*.

Algorithm B_2 runs as follows

```
1. $(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID}}(1^t)$.
2. $(\mathtt{ID}^*, s) \leftarrow A_1^{\mathcal{O}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}})$.
3. $c_1^* \leftarrow \mathcal{O}_{\mathbb{E}_{\mathtt{PK}}}()$.
4. $m_2 \leftarrow \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}}), r \leftarrow \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.
5. $c_2^* \leftarrow \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}^*, M_{\mathfrak{pk}}, m_2; r)$.
6. $k^* \leftarrow \mathbb{K}_{\mathtt{CL}}(M_{\mathfrak{pk}})$.
7. $c^* \leftarrow (c_1^*, c_2^*)$.
8. $b'' \leftarrow A_2^{\mathcal{O}}(c^*, k^*, s, \mathtt{ID}^*)$.
9. Output $b''$.
```

Algorithm B_2 answers the oracle calls of the algorithm A just as the oracles do in $\mathtt{Game}_2$. Except we make the following alterations:

- **Public Key Request/Private Key Extraction:** The j-th entry of the L_X list is replaced by $(\mathtt{ID}^*, \mathfrak{pk}^*, \perp)$, where $\mathtt{ID}^*$ is the challenge identity output by A_1 and $\mathfrak{pk}^*$ is the input public key for algorithm B_2.
- **Decapsulation Queries:** We need to modify this when called with input (c_1, c_2) and $\mathtt{ID}^*$ as we no longer know $\mathfrak{sk}^*$. We then perform the following steps:
  - We decrypt c_2 so as to obtain m_2.
  - If c_1 is not a valid ciphertext, which can be determined via the ciphertext validity oracle provided to the CPA^{++} adversary B_2, we return $\perp$.
  - If $(k, c_1, \mathfrak{pk}, m_1, m_2)$ is in L_H then we check whether c_1 is a valid encryption of m_1, using the plaintext/ciphertext checking oracle provided to the CPA^{++} adversary B_2. If so we output k.
  - Otherwise we check whether $(k, c, \mathfrak{pk}, \perp, m_2)$ is in L_H, for a ciphertext c which encrypts the same message as c_1, if so we output k. This uses the ciphertext equality oracle provided to the CPA^{++} adversary B_2.
  - Else we pick k at random, insert $(k, c_1, \mathfrak{pk}, \perp, m_2)$ in L_H and return k.
- **Hash Function:** Here we modify the oracle to make it compatible with the above decapsulation oracle. If H is called with input $(c_1, \mathfrak{pk}, m_1, m_2)$ then we respond as follows:
  - If $(k, c_1, \mathfrak{pk}, m_1, m_2)$ is in L_H then we output k.
  - If $(k, c_1, \mathfrak{pk}, \perp, m_2)$ is in L_H and c is a valid encryption of m_1 then we output k and update the entry to read $(k, c_1, \mathfrak{pk}, m_1, m_2)$. This uses the plaintext/ciphertext checking oracle provided to the CPA^{++} adversary B_2.
  - Else we pick k at random, place $(k, c_1, \mathfrak{pk}, m_1, m_2)$ into L_H and return k.

With these simulations algorithm A cannot notice the difference between running in $\mathtt{Game}_2$ and running as a subroutine for algorithm B_2. When algorithm B_2 terminates it selects a random element from the L_H $(k, c_1, \mathfrak{pk}, m_1, m_2)$, such that $m_1 \neq \perp$ and returns m_1. There are at most $q_H + q_D$ elements in L_H and therefore, from (12) and (13) we have

$$\begin{aligned}\mathrm{Adv}_{\mathtt{PK}}^{\mathtt{OW\text{-}CPA}}(B_2) &= \frac{\Pr[\mathtt{S}_2|\mathtt{Ask}]}{q_H + q_D}\\ &\geq \frac{\Pr[\mathtt{S}_1|\mathtt{Ask}]}{(q_H + q_D)(q_{PK} + q_{SK} + q_D)}\end{aligned}$$

$$\geq \frac{\Pr[\mathrm{S}_0]-\frac{1}{2}}{(q_H+q_D)(q_{PK}+q_{SK}+q_D)}$$
$$=\frac{\mathrm{Adv}^{\texttt{Type-II}}_{\texttt{CL-KEM}}(A)}{2(q_H+q_D)(q_{PK}+q_{SK}+q_D)}.$$

The result follows. □

Acknowledgements

The authors would like to offer their thanks to Mihir Bellare, Alex Dent, Kenny Paterson and the anonymous referees for providing detailed and insightful comments on earlier versions of this work.

References

[1] S.S. Al-Riyami. Cryptographic Schemes Based on Elliptic Curve Pairings. Ph.D. Thesis, University of London, 2004.

[2] S.S. Al-Riyami and K.G. Paterson. CBE from CL-PKE: A Generic Construction and Efficient Schemes. In *Public Key Cryptography—PKC 2005*, LNCS 3386, pp. 398–415. Springer, Berlin, 2005.

[3] S.S. Al-Riyami and K.G. Paterson. Certificateless Public Key Cryptography. In *Advances in Cryptology—ASIACRYPT 2003*, LNCS 2894, pp. 452–473. Springer, Berlin, 2003.

[4] M. Barbosa and P. Farshim. Efficient Identity-Based Key Encapsulation to Multiple Parties. In *Cryptography and Coding 2005*, LNCS 3796, pp. 428–441. Springer, Berlin, 2005.

[5] M. Bellare, A. Desai, E. Jokipii and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. In *38th IEEE Symposium on Foundations of Computer Science—FOCS*, pp. 94–403, 1997.

[6] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations Among Notions of Security for Public Key Encryption Schemes. In *Advances in Cryptology—CRYPTO '98*, LNCS 1462, pp. 26–45. Springer, Berlin, 1998.

[7] M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In *1st ACM Conference on Computer and Communications Security*, pp. 62–73. ACM, New York, 1993.

[8] D. Boneh and X. Boyen. Efficient Selective-ID Secure IBE without Random Oracles. In *Advances in Cryptology—EUROCRYPT 200*, LNCS 3027, pp. 223–238. Springer, Berlin, 2004.

[9] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.* **32**:586–615, 2003.

[10] L. Chen and Z. Cheng. Security Proof of Sakai–Kasahara's IBE Scheme. In *Proceedings of Cryptography and Coding 2005*, LNCS 3796, pp. 442–459. Springer, Berlin, 2005.

[11] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure Against Adaptive Chosen Ciphertext Attack. *SIAM J. Comput.* **33**:167–226, 2003.

[12] A. Dent. A Designer's Guide to KEMs. In *Cryptography and Coding, 2003*, LNCS 2898, pp. 133–151. Springer, Berlin, 2003.

[13] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Advances in Cryptology—CRYPTO '99*, LNCS 1666, pp. 537–554. Springer, Berlin, 1999.

[14] C. Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. In *Advances in Cryptology—EUROCRYPT 2003*, LNCS 2656, pp. 272–293. Springer, Berlin, 2003.

[15] C. Gentry. Practical Identity-Based Encryption without Random Oracles. In *Advances in Cryptology—EUROCRYPT 2006*, LNCS 4004, pp. 445–464. Springer, Berlin, 2006.

[16] T. Okamoto and D. Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In *Public Key Cryptography—PKC 2001*, LNCS 1992, pp. 104–118. Springer, Berlin, 2001.

[17] D. Pointcheval and T. Okamoto. REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform. In *Topics in Cryptography—CT-RSA 2001*, LNCS 2020, pp. 159–175. Springer, Berlin, 2001.
[18] V. Shoup. Using Hash Functions as a Hedge against Chosen Ciphertext Attack. In *Advances in Cryptology—EUROCRYPT 2000*, LNCS 1807, pp. 275–288. Springer, Berlin, 2000.
[19] B. Waters. Efficient Identity-Based Encryption without Random Oracles. In *Advances in Cryptology—EUROCRYPT 2005*, LNCS 3494, pp. 114–127. Springer, Berlin, 2005.