Generic Constructions of Identity-Based and Certificateless KEMs

K. Bentahar, P. Farshim, J. Malone-Lee and N.P. Smart

Dept. Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom.
{bentahar, farshim, malone, nigel}@cs.bris.ac.uk

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

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 part of the algorithm is known as the *key encapsulation mechanism* (KEM) while the symmetric part – where the message is actually encrypted – is known as the *data encapsulation mechanism* (DEM). The formalisation of this basic approach originates in the work of Shoup [15]. The resulting KEM/DEM encryption paradigm has received much attention in recent years [6, 7, 15]. It is very attractive as it gives a clear separation between the various parts of the cipher allowing for modular design.

In [7] 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 which we aim to extend in this paper, by applying these techniques to two types of recently introduced, but closely related, primitives: ID-based encryption and certificateless encryption.

A secure and efficient ID-based encryption algorithm was introduced by Boneh and Franklin [4], based on pairings on elliptic curves. In [10], Lynn mentions that the encryption algorithm proposed by Boneh and Franklin is unlikely to be used, since in practice one will use a form of key encapsulation. We call such an encapsulation mechanism an ID-KEM. Lynn proceeds to mention a possible ID-KEM construction, but he gives no security model or proof.

As mentioned above, one of the contributions of this paper is to formalise the notion of key encapsulation for the ID-based setting. Having done this, we show that Lynn's construction for an ID-KEM can be used to build a fully-secure ID-based encryption scheme, when combined with an appropriate DEM. The resulting scheme is computationally more efficient than the original Boneh and Franklin construction. The security proof at first sight seems tighter for Lynn's construction, however, the proof of security relies on a stronger assumption, namely the *gap bilinear Diffie–Hellman problem*, described in Section 6, as opposed to the Bilinear Diffie–Hellman problem on which the Boneh–Franklin scheme is based.

We also present a generic construction of an ID-KEM, which is secure in a strong sense, from any ID-based encryption scheme, which is secure in a weak sense. When we instantiate this generic scheme with the BasicIdent scheme from [4] we obtain an ID-KEM which is as efficient as the Boneh–Franklin construction, and which is based on the, now standard, Bilinear Diffie–Hellman problem. We feel our construction of an ID-based encryption scheme from an ID-KEM and a standard DEM is more natural than the construction in [4], which relies on the Fujisaki–Okamoto transform [8].

Another contribution of this paper is to present a security model for key encapsulation applied to certificateless encryption. This form of encryption was introduced and developed in a series of works by Al-Riyami and Paterson [1–3]. The idea is to have the benefit of ID-based encryption (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) ID-based encryption scheme plus a special form of (weakly secure) public key scheme, such as RSA or ElGamal in certain groups, and then constructs a CL-KEM from this. The resulting scheme is secure in a strong sense.

The security model for a certificateless encryption scheme [1–3], and therefore for a CL-KEM, has two types of adversarial attack. We show that combining the CL-KEM with a standard DEM results in a secure certificateless encryption scheme. This generic approach allows one to add certificateless encryption onto an infrastructure of existing RSA and ElGamal keys, which are either not certified or whose certificates are not trusted by the sender. In order to prove our composition result we need to modify the definitions of security for a certificateless encryption schemes 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, ID-based and certificateless encryption. In Section 4 we present the analogous definitions for KEMs. In Section 5 we show a simple generalisation of the hybrid result of Cramer and Shoup [6], which allows us to combine any KEM meeting our security definitions in Section 4 with a standard DEM so as to meet the security definitions of an encryption scheme in Section 3. In Section 6 we give a brief overview of the pairings needed for some of our later discussion and an overview of the ID-based encryption scheme of Boneh and Franklin. In Section 7 we present our constructions of ID-KEMs, and we prove them secure under our definitions in Section 4. In Section 8 we compare the resulting ID-based encryption schemes with the original ID-based scheme presented in [4]. Finally, in Section 9 we present our construction of a generic CL-KEM and we prove that it is secure.

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 simply to reduce the amount of notation. Our results can be expanded to cope with this by inserting a domain parameter generation algorithm which takes as input 1^t, where t is a security parameter; the output of this algorithm would then be passed to and then passing to 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. 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 a schemes with a weaker soundness definition in the standard way [6].

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).

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 $\mathbb{R}_{\mathtt{PK}}(\cdot)$. These spaces are all parametrised by a public key, and hence by the security parameter t. A public key encryption scheme is defined by a triple of 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 and 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 a random tape $r \in \mathbb{R}_{\mathtt{PK}}(\mathfrak{pk})$, and 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 $\perp$.

Consider the following two-stage games between an adversary $A = (A_1, A_2)$ of the encryption algorithm and a challenger.

OW Adversarial Game	IND Adversarial Game
1. $(\mathfrak{pk}, \mathfrak{sk}) \leftarrow \mathbb{G}_{\mathtt{PK}}(1^t)$.	1. $(\mathfrak{pk}, \mathfrak{sk}) \leftarrow \mathbb{G}_{\mathtt{PK}}(1^t)$.
2. $s \leftarrow A_1^{\mathcal{O}_{\mathfrak{pk}}}(\mathfrak{pk})$.	2. $(s, m_0, m_1) \leftarrow A_1^{\mathcal{O}_{\mathfrak{pk}}}(\mathfrak{pk})$.
3. $m \leftarrow \mathbb{M}_{\mathtt{PK}}(\mathfrak{pk})$.	3. $b \leftarrow \{0, 1\}$.
4. $c^* \leftarrow \mathbb{E}_{\mathtt{PK}}(\mathfrak{pk}, m; r)$.	4. $c^* \leftarrow \mathbb{E}_{\mathtt{PK}}(\mathfrak{pk}, m_b; r)$.
5. $m' \leftarrow A_2^{\mathcal{O}_{\mathfrak{pk}}}(\mathfrak{pk}, c^*, s)$.	5. $b' \leftarrow A_2^{\mathcal{O}_{\mathfrak{pk}}}(\mathfrak{pk}, c^*, s, m_0, m_1)$.

In the above games s is some state information and $\mathcal{O}_{\mathfrak{pk}}$ denotes the oracles to which the adversary has access. There are various possibilities for this oracle depending on the attack model for our game:

- CPA Model: In this model the adversary does not have access to any oracles.
- CCA2 Model: In this model the oracle $\mathcal{O}_{\mathfrak{pk}}$ is a decryption oracle with respect to the public key $\mathfrak{pk}$. The adversary has access to $\mathcal{O}_{\mathfrak{pk}}$, subject to the restriction that in the second phase, once it has been given c^*, A is not allowed to call $\mathcal{O}_{\mathfrak{pk}}$ with the challenge encryption c^*.

If we let MOD denote the mode of the attack, namely either CPA and CCA2, the adversary's advantage in the first game is defined to be

$$\mathrm{Adv}_{\mathtt{PK}}^{\mathtt{OW-MOD}}(A) = \Pr[m' = m],$$

while the advantage in the second game is given by

$$\mathrm{Adv}_{\mathtt{PK}}^{\mathtt{IND-MOD}}(A) = |2\Pr[b' = b] - 1|.$$

A public key encryption algorithm is considered to be secure, in the sense of a given goal and attack model (IND-CCA2 for example) if, for all PPT bounded adversaries, the advantage in the relevant game above is a negligible function of the security parameter t.

We also define an attack notion of CPA^{++}, in this model the adversary is given access to the following oracles:

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

Dent [7] calls the attack model in which an adversary has access to only the first of these oracles a CPA$^+$ model, which motivates our naming. The CPA$^+$ model was first used in [9] to attack a version of the EPOC-2 cipher; it is sometimes referred to as a "reaction attack".

In our generic CL-KEM construction we shall only require a scheme which is OW-CPA^{++}. Such schemes are readily available, for example, the naive text-book RSA scheme is such a scheme, assuming the RSA problem is hard. As another example, one can take text-book ElGamal, on the assumption that the gap Diffie–Hellman problem [12] is hard.

Public key encryption schemes for which an explicit algorithm exists to implement a plaintext checking oracle will be called *verifiable*. Note that textbook RSA is verifiable, as is textbook ElGamal if one implements it in a group on which there is a bilinear pairing.

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

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

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

3.2 ID-Based Encryption Schemes

Here we give the security notions for an ID-based encryption scheme, as first introduced by Boneh and Franklin [4].

We define the message, ciphertext and randomness spaces of our ID scheme by $\mathbb{M}_{\mathtt{ID}}(\cdot)$, $\mathbb{C}_{\mathtt{ID}}(\cdot)$, $\mathbb{R}_{\mathtt{ID}}(\cdot)$. These are parametrised by the master public key $M_{\mathfrak{pk}}$, and hence by the security parameter t. An ID-based encryption scheme is specified by four polynomial time algorithms:

– $\mathbb{G}_{\mathtt{ID}}(1^t)$: A PPT algorithm which takes as input 1^t and returns the master public key $M_{\mathfrak{pk}}$ and the master secret key $M_{\mathfrak{sk}}$.
– $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{sk}}, \mathtt{ID}_A)$: A deterministic private key extraction algorithm which takes as input $M_{\mathfrak{sk}}$ and $\mathtt{ID}_A \in \{0,1\}^*$, an identifier string for A, and returns the associated private key $D_{\mathtt{ID}_A}$.
– $\mathbb{E}_{\mathtt{ID}}(\mathtt{ID}_A, M_{\mathfrak{pk}}, m; r)$: This is the PPT 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 $c \in \mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.
– $\mathbb{D}_{\mathtt{ID}}(D_{\mathtt{ID}_A}, c)$: This is the deterministic decryption algorithm. On input of the private key $D_{\mathtt{ID}_A}$ and a ciphertext c this outputs the corresponding value of the plaintext m or a failure symbol $\perp$.

Consider the following two-stage games between an adversary A of the encryption algorithm and a challenger.

ID-OW Adversarial Game

1. $(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID}}(1^t)$.
2. $(s, \mathtt{ID}^*) \leftarrow A_1^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathfrak{pk}})$.
3. $m \leftarrow \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.
4. $c^* \leftarrow \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}^*, M_{\mathfrak{pk}}, m; r)$.
5. $m' \leftarrow A_2^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathfrak{pk}}, c^*, s, \mathtt{ID}^*)$.

ID-IND Adversarial Game

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

In the above, s is some state information and $\mathcal{O}_{\mathtt{ID}}$ are oracles to which the adversary has access. There are various possibilities for these oracles depending on the attack model for our game:

- CPA Model: In this model the adversary only has access to a private key extraction oracle which on input of $\mathtt{ID} \neq \mathtt{ID}^*$ will output the corresponding value of $D_{\mathtt{ID}}$.
- CCA2 Model: In this model the adversary has access to the private key extraction oracle as above, but it also has access to a decryption oracle with respect to any identity $\mathtt{ID}$ of the adversary's choosing. The adversary has access to this decryption oracle, subject to the restriction that in the second phase A is not allowed to call the decryption oracle with the pair $(c^*, \mathtt{ID}^*)$.

If we let MOD denote the mode of attack, either CPA or CCA2, the adversary's advantage in the first game is defined to be

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-OW-MOD}}(A) = \Pr[m' = m],$$

while the advantage in the second game is given by

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-IND-MOD}}(A) = |2\Pr[b' = b] - 1|.$$

An ID-based encryption algorithm is considered to be secure, in the sense of a given goal and attack model (ID-IND-CCA2 for example) if, for all PPT adversaries, the advantage in the relevant game is a negligible function of the security parameter t.

Again, 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

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

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.

3.3 Certificateless Encryption Schemes

We now describe certificateless encryption schemes as proposed by Al-Riyami and Paterson. See [1–3] for further details.

A certificateless scheme makes use of a trusted third party known as a key generation centre (KGC). Unlike the trusted party in an ID-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.

Formally, a certificateless scheme is specified by seven polynomial time algorithms:

- $\mathbb{G}_{\mathtt{CL}}(1^t)$. A PPT algorithm which takes as input 1^t and returns the master public key $M_{\mathfrak{pk}}$ and the master secret key $M_{\mathfrak{sk}}$.
- `Partial-Private-Key-Extract`. A deterministic algorithm which takes as input $M_{\mathfrak{sk}}$ and an identifier string for A, $\mathtt{ID}_A \in \{0,1\}^*$ and returns a partial private key $D_{\mathtt{ID}_A}$.
- `Set-Secret-Value`. A PPT algorithm which takes no input (bar the system parameters) and outputs a secret value $\mathfrak{sk}_A$.
- `Set-Public-Key`. A deterministic algorithm which takes as input the secret value $\mathfrak{sk}_A$ and outputs a public key $\mathfrak{pk}_A$.
- `Set-Private-Key`. A deterministic algorithm which takes as input a partial private key $D_{\mathtt{ID}_A}$ and a secret value $\mathfrak{sk}_A$ and returns the (full) private key S_A.
- $\mathbb{E}_{\mathtt{CL}}(\mathfrak{pk}_A, \mathtt{ID}_A, M_{\mathfrak{pk}}, m; r)$. This is the PPT encryption algorithm. On input of a public key $\mathfrak{pk}_A$, an identifier $\mathtt{ID}_A$, the master public key $M_{\mathfrak{pk}}$, a message $m \in \mathbb{M}_{\mathtt{CL}}(M_{\mathfrak{pk}})$ and possibly some randomness $r \in \mathbb{R}_{\mathtt{CL}}(M_{\mathfrak{pk}})$, this algorithm outputs a ciphertext $c \in \mathbb{C}_{\mathtt{CL}}(M_{\mathfrak{pk}})$.
- $\mathbb{D}_{\mathtt{CL}}(S_A, c)$. This 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 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 behaviour 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 the 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 formally describe the two types of adversary that we have discussed.

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

In the above s is some state information.

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

The adversary's advantage is defined to be

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

where X is either I, I$^-$ or II (see below for definition of I$^-$). We now turn to the various oracle accesses of the adversaries in each game.

Type-I Adversary Oracle Access: This adversary may request public keys, replace public keys with 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, 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 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, 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 must weaken the requirement of Type-I security and replace it with Type-I$^-$.

4 Public Key, Identity-Based and Certificateless Key Encapsulation Mechanisms

In this section we recall the basic definitions of Key Encapsulation Mechanisms (KEMs). We then extend this concept to the ID-based and certificateless situations. We let $\mathbb{K}_{\mathtt{KEM}}(\mathfrak{pk}), \mathbb{K}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}), \mathbb{K}_{\mathtt{CL-KEM}}(M_{\mathfrak{pk}})$ denote the space of keys output by our various KEMs, and we let $\mathbb{C}_{\mathtt{KEM}}(\mathfrak{pk}), \mathbb{C}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}), \mathbb{C}_{\mathtt{CL-KEM}}(M_{\mathfrak{pk}})$ denote the respective space of encapsulations. All of these spaces are parametrised by a public key and so are indirectly parametrised by a security parameter t.

4.1 Public-Key Key Encapsulation Mechanisms

A standard KEM – one in the traditional public key setting – is defined by a triple of probabilistic polynomial time (PPT) algorithms $(\mathbb{G}_{\mathtt{KEM}}, \mathbb{E}_{\mathtt{KEM}}, \mathbb{D}_{\mathtt{KEM}})$:

- $\mathbb{G}_{\mathtt{KEM}}(1^t)$ is the (randomised) 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 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 $\perp$.

Consider the following two-stage games between an adversary A of the KEM and a challenger.

OW Adversarial Game
1. $(\mathfrak{pk}, \mathfrak{sk}) \leftarrow \mathbb{G}_{\mathtt{KEM}}(1^t)$.
2. $s \leftarrow A_1^{\mathcal{O}_{\mathfrak{pk}}}(\mathfrak{pk})$.
3. $(k, c^*) \leftarrow \mathbb{E}_{\mathtt{KEM}}(\mathfrak{pk})$.
4. $k' \leftarrow A_2^{\mathcal{O}_{\mathfrak{pk}}}(\mathfrak{pk}, c^*, s)$.

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

Here s is some state information and $\mathcal{O}_{\mathfrak{pk}}$ is a decapsulation oracle with respect to the public key $\mathfrak{pk}$. In the CPA attack model the adversary is not allowed any access to $\mathcal{O}_{\mathfrak{pk}}$, while in the CCA2 attack model it does have access to $\mathcal{O}_{\mathfrak{pk}}$, subject to the restriction that in the second phase A is not allowed to call $\mathcal{O}_{\mathfrak{pk}}$ with the challenge encapsulation c^*.

We let MOD denote either CPA or CCA2. The adversary's advantage in the first game is defined to be

$$\mathrm{Adv}_{\mathtt{KEM}}^{\mathtt{OW-MOD}}(A) = \Pr[k' = k],$$

while the advantage in the second game is given by

$$\mathrm{Adv}_{\mathtt{KEM}}^{\mathtt{IND-MOD}}(A) = |2\Pr[b' = b] - 1|.$$

A KEM is considered to be secure, with respect to a given goal and attack model (IND-CCA2 for example) if, for all PPT adversaries, the advantage in the relevant game above is a negligible function of the security parameter t.

4.2 ID-Based Key Encapsulation Mechanisms

An ID-KEM scheme is specified by four polynomial time algorithms:

- $\mathbb{G}_{\mathtt{ID-KEM}}(1^t)$. A PPT algorithm which takes as input 1^t and returns the master public key $M_{\mathfrak{pk}}$ and the master secret key $M_{\mathfrak{sk}}$.
- $\mathbb{X}_{\mathtt{ID-KEM}}(M_{\mathfrak{sk}}, \mathtt{ID}_A)$. A deterministic algorithm which takes as input $M_{\mathfrak{sk}}$ and an identifier string for A, $\mathtt{ID}_A \in \{0,1\}^*$, and returns the associated private key $D_{\mathtt{ID}_A}$.
- $\mathbb{E}_{\mathtt{ID-KEM}}(\mathtt{ID}_A, M_{\mathfrak{pk}})$. This is the PPT encapsulation algorithm. On input of $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$ this outputs a pair (k, c) where $k \in \mathbb{K}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}})$ is a key and $c \in \mathbb{C}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}})$ is the encapsulation of that key.
- $\mathbb{D}_{\mathtt{ID-KEM}}(D_{\mathtt{ID}_A}, c)$. This is the deterministic decapsulation algorithm. On input of c and $D_{\mathtt{ID}_A}$ this outputs k or a failure symbol $\perp$.

Consider the following two-stage games between an adversary A of the ID-KEM and a challenger.

ID-OW Adversarial Game
1. $(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID-KEM}}(1^t)$.
2. $(s, \mathtt{ID}^*) \leftarrow A_1^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathfrak{pk}})$.
3. $(k, c^*) \leftarrow \mathbb{E}_{\mathtt{ID-KEM}}(\mathtt{ID}^*, M_{\mathfrak{pk}})$.
4. $k' \leftarrow A_2^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathfrak{pk}}, c^*, s, \mathtt{ID}^*)$.

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

In the above s is some state information and $\mathcal{O}_{\mathtt{ID}}$ denotes oracles to which the adversary has access. There are two possibilities for these oracles depending on the attack model for our game:

- CPA Model: In this model the adversary only has access to a private key extraction oracle which, on input of $\mathtt{ID} \neq \mathtt{ID}^*$, will output the corresponding value of $D_{\mathtt{ID}}$.
- CCA2 Model: In this model the adversary has access to the private key extraction oracle as above, but it also has access to a decapsulation oracle with respect to any identity $\mathtt{ID}$ of the adversary's choosing. The adversary has access to this decapsulation oracle, subject to the restriction that in the second phase A is not allowed to call $\mathcal{O}_{\mathtt{ID}}$ with the pair $(c^*, \mathtt{ID}^*)$.

The adversary's advantage in the first game is defined to be

$$\mathrm{Adv}_{\mathtt{ID-KEM}}^{\mathtt{ID-OW-MOD}}(A) = \Pr[k' = k].$$

While the advantage in the second game is given by

$$\mathrm{Adv}_{\mathtt{ID-KEM}}^{\mathtt{ID-IND-MOD}}(A) = |2\Pr[b' = b] - 1|.$$

An ID-KEM is considered to be secure, in the sense of a given goal and attack model (ID-IND-CCA2 for example) if for all PPT adversaries A, the advantage in the relevant game above is a negligible function of the security parameter t.

4.3 CL-KEM Definition

We now adapt the KEM definition of Section 4.1 to the case of the certificateless systems of Section 3.3.

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

- $\mathbb{G}_{\mathtt{CL-KEM}}(1^t)$. A PPT algorithm which 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`. A deterministic algorithm which takes as input $M_{\mathfrak{sk}}$ and an identifier string for A, $\mathtt{ID}_A \in \{0,1\}^*$ and returns a partial private key $D_{\mathtt{ID}_A}$.
- `Set-Secret-Value`. A PPT algorithm which takes no input (bar the system parameters) and outputs a secret value $\mathfrak{sk}_A$.
- `Set-Public-Key`. A deterministic algorithm which takes as input $\mathfrak{sk}_A$ and outputs a public key $\mathfrak{pk}_A$.
- `Set-Private-Key`. A deterministic algorithm which takes as input $D_{\mathtt{ID}_A}$ and $\mathfrak{sk}_A$ and returns S_A the (full) private key.
- $\mathbb{E}_{\mathtt{CL-KEM}}(\mathfrak{pk}_A, \mathtt{ID}_A, M_{\mathfrak{pk}})$. This 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 $k \in \mathbb{K}_{\mathtt{CL-KEM}}(M_{\mathfrak{pk}})$ is a key and $c \in \mathbb{C}_{\mathtt{CL-KEM}}(M_{\mathfrak{pk}})$ is the encapsulation of that key.
- $\mathbb{D}_{\mathtt{CL-KEM}}(S_A, c)$. This is the deterministic decapsulation algorithm. On input of c and S_A this outputs 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, called a Type-I, Type-I$^-$ and a Type-II adversary. Each adversary is trying to win one of the following games, where the various oracle accesses allowed are identical to those defined in Section 3.3, where we simply replace the word "decryption" with "decapsulation".

Type-I Adversarial Game

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

Type-II Adversarial Game

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

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

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

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 in both games. We note that constructing CL-KEMs which are secure in the Type-I sense is relatively easy, and that such a CL-KEM is automatically Type-I$^-$ secure. Hence, we shall only be using Type-I$^-$ secure CL-KEMs in our security proof for hybrid CL encryption.

5 Combining KEMs, ID-KEMs, 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 (DEMs). In addition, when we compare our ID-KEM/DEM construction with that of the original Boneh–Franklin ID-based encryption scheme, we will need to know the exact security guarantees we can obtain from our construction. Hence, in this section we first recap on the definition of DEMs and then we generalise the hybrid construction of [6, Section 7] to the situation of ID-KEMs and CL-KEMs.

5.1 One-Time Symmetric Encryption

A one-time symmetric encryption scheme is a pair of deterministic polynomial time secret key ($\mathtt{SK}$) algorithms, $\mathbb{E}_{\mathtt{SK}}$ and $\mathbb{D}_{\mathtt{SK}}$, where key, message and ciphertext spaces are given by $\mathbb{K}_{\mathtt{SK}}(t)$, $\mathbb{M}_{\mathtt{SK}}(t)$, $\mathbb{C}_{\mathtt{SK}}(t)$ for some security parameter t.

- $\mathbb{E}_{\mathtt{SK}}(k, m)$. On input of $k \in \mathbb{K}_{\mathtt{SK}}(t)$ and $m \in \mathbb{M}_{\mathtt{SK}}(t)$ this outputs a value $c \in \mathbb{C}_{\mathtt{SK}}(t)$.

– $\mathbb{D}_{\mathtt{SK}}(k, c)$. This performs the inverse operation, or outputs $\perp$ if c is not the encryption of a message m under the key k.

We will assume $\mathbb{M}_{\mathtt{SK}}(t) = \{0,1\}^*$ and that the scheme is sound: for all m we have $\mathbb{D}_{\mathtt{SK}}(k, \mathbb{E}_{\mathtt{SK}}(k, m)) = m$. We assume that the key length $|k|$ is a polynomial function of the security parameter t.

Security of one-time symmetric encryption schemes is defined via the following game:

1. $(s, m_0, m_1) \leftarrow A_1(1^t)$.
2. $b \leftarrow \{0, 1\}$.
3. $k \leftarrow \mathbb{K}_{\mathtt{SK}}(t)$.
4. $c^* \leftarrow \mathbb{E}_{\mathtt{SK}}(k, m_b)$.
5. $b' \leftarrow A_2^{\mathcal{O}_k}(c^*, s, m_0, m_1)$.

In the above s is state information.

We let $\mathcal{O}_k$ denote an oracle to which the adversary has access. There two various possibilities for this oracle depending on the attack model for our game:

– PA Model: In this passive attack model the adversary has no access to any oracles.
– CCA Model: In this model the oracle $\mathcal{O}_k$ is a decryption oracle for the key k chosen by the challenger in the third step of the above game. This oracle is only available in the second stage of A's game and it is not allowed to be called on the challenge ciphertext c^*.

If we let MOD denote either PA or CCA, the adversary's advantage in the game, called Find-Guess or FG, is defined to be

$$\mathrm{Adv}_{\mathtt{SK}}^{\mathtt{FG-MOD}}(A) = |2\Pr[b' = b] - 1|.$$

A one-time symmetric encryption scheme is considered to be secure, in the sense of a given attack model if, for all PPT adversaries, the advantage in the above game is a negligible function of the security parameter t.

For our purposes we will require a one-time symmetric encryption scheme that is secure in the sense of IND-CCA. We call such a *data encapsulation mechanism* (DEM). The construction of DEMs from PA secure symmetric encryption schemes and MACs is discussed in [6].

5.2 Hybrid Constructions

We prove secure our hybrid constructions, which allow one to construct ID-IND-CCA2 secure ID-based encryption schemes from ID-IND-CCA2 secure ID-KEMs and DEMs, and also how to construct certificateless encryption schemes secure against Type-I$^-$ and Type-II adversaries in a similar manner.

We assume that the key space output by the KEMs corresponds to the key space required by the DEM. Our construction follows that in [6, Section 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, and we refer to such a construction as *hybrid*.

In our proofs we will make use of the following key lemma [6].

Lemma 1. *Let U_1, U_2 and $\mathtt{F}$ be events defined on some probability space. Suppose that $\Pr[U_1 \wedge \neg\mathtt{F}] = \Pr[U_2 \wedge \neg\mathtt{F}]$, then*

$$|\Pr[U_1] - \Pr[U_2]| \leq \Pr[\mathtt{F}].$$

The following theorem is a natural generalisation of Theorem 5 of [6]. Note that, unlike the equivalent result in [6], we are implicitly assuming that for all keys, all encapsulations decapsulate properly. It would be straightforward to generalise 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 ID-based encryption scheme (resp. the hybrid certificateless scheme) in the sense of ID-IND-CCA2 (resp. Type-I$^-$ and Type-II) adversaries, then there exists PPT adversaries B_1 and B_2, whose running time is essentially that of A, such that*

$$\begin{aligned}
\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-IND-CCA}}(A) &\leq 2\mathrm{Adv}_{\mathtt{ID-KEM}}^{\mathtt{ID-IND-CCA}}(B_1) + \mathrm{Adv}_{\mathtt{DEM}}^{\mathtt{FG-CCA}}(B_2),\\
\mathrm{Adv}_{\mathtt{CL}}^{\mathtt{Type-I^-}}(A) &\leq 2\mathrm{Adv}_{\mathtt{CL-KEM}}^{\mathtt{Type-I^-}}(B_1) + \mathrm{Adv}_{\mathtt{DEM}}^{\mathtt{FG-CCA}}(B_2),\\
\mathrm{Adv}_{\mathtt{CL}}^{\mathtt{Type-II}}(A) &\leq 2\mathrm{Adv}_{\mathtt{CL-KEM}}^{\mathtt{Type-II}}(B_1) + \mathrm{Adv}_{\mathtt{DEM}}^{\mathtt{FG-CCA}}(B_2).
\end{aligned}$$

Before proceeding with the proof we note that 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 $\mathtt{Game}_0, \mathtt{Game}_1, \mathtt{Game}_2$ 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 $\mathtt{ID}^*$ – the identity chosen by the adversary on which it wishes to be challenged. For any $i = 0, 1, 2$, we let $\mathtt{S}_i$ be the event that $b' = b$ in game $\mathtt{Game}_i$, 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 $\mathtt{Game}_0$ be the genuine attack game played by A. So by definition we have

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

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*

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

where MOD is Type-I$^-$, Type-II or IND-CCA2 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, then B_1 simply relays these queries to its own equivalent oracle.
- To respond to A's decryption oracle query for an identity $\texttt{ID}$ 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 $\texttt{ID}^*$ and messages (m_0, m_1), B_1 first calls its own challenge encryption oracle with $\texttt{ID}^*$ to obtain $(k^\dagger, c_1^*)$. It then chooses a bit d at random and computes $c_2^* \leftarrow \mathbb{E}_{\texttt{DEM}}(k^\dagger, m_d)$. Finally, it responds to A with (c_1^*, c_2^*).
- To respond to A's decryption oracle query for an identity $\texttt{ID}$ and a ciphertext (c_1, c_2) after A has queried its challenge encryption oracle, B_1 proceeds as follows.
  - If $(\texttt{ID}, c_1) \neq (\texttt{ID}^*, c_1^*)$ 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 $(\texttt{ID}, c_1) = (\texttt{ID}^*, c_1^*)$ and the public key has been replaced, then B_1 responds by calling the decapsulation oracle provided to it by A with input $(\texttt{ID}^*, c_1^*)$ 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

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

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

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

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 decapsulation 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^*) 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.

6 Review of Pairings and the Boneh–Franklin Construction

Some of our constructions for CL-KEMs and ID-KEMs require groups equipped with a *bilinear map*. We briefly review the necessary facts about bilinear maps and bilinear groups. Further details may be found in [4, 5]. Having done this, we go on to discuss the Boneh and Franklin construction of a secure ID-based encryption scheme based on such pairings.

6.1 Bilinear Groups

Let G_1, G_2 and G_T be groups with the following properties.

- G_1 and G_2 are additive groups of prime order q.
- G_1 has generator P_1 and G_2 has generator P_2.
- There is an isomorphism ρ from G_2 to G_1, with $\rho(P_2) = P_1$.
- There is a bilinear map $\hat{t} : G_1 \times G_2 \rightarrow G_T$.

In many cases one can set $G_1 = G_2$ as is done in [4]. When this is so, we can take ρ to be the identity map; however, to take advantage of certain families of groups [11], we do not restrict ourselves to this case.

We have stipulated that our groups should have a bilinear map, $\hat{t} : G_1 \times G_2 \rightarrow G_T$. This should satisfy the following conditions:

1. Bilinear: Given any $Q \in G_1$, $W \in G_2$ and $a, b \in \mathbb{F}_q$ we have
$$\hat{t}(aQ, bW) = \hat{t}(Q, W)^{ab}$$
2. Non-degenerate: $\hat{t}(P_1, P_2) \neq 1$.
3. Efficiently computable.

The map $\hat{t}$ is usually derived from the Weil or Tate pairings on an elliptic curve.

Definition 1 (Bilinear groups). *We say that G_1 and G_2 are bilinear groups if there exists a group G_T with $|G_T| = |G_1| = |G_2| = q$, an isomorphism ρ and a bilinear map $\hat{t}$ satisfying the conditions above; moreover, the group operations in G_1, G_2 and G_T, ρ and $\hat{t}$ must be efficiently computable.*

A *group description* $\Gamma[G_1, G_2, G_T, \hat{t}, \rho, q, P_1, P_2]$ describes a given set of bilinear groups as above. We abbreviate such a group description to Γ henceforth. There are several hard problems associated with a group description Γ which we are interested in for building cryptosystems. These have their origins in the work of Boneh and Franklin [4].

Bilinear Diffie–Hellman Problem (BDH). Consider the following game, for a group description Γ and an adversary A,

1. $a, b, c \leftarrow \mathbb{F}_q^*$.
2. $\alpha \leftarrow A(aP_2, bP_2, cP_2, \Gamma)$.

The advantage of the adversary is defined to be

$$\mathrm{Adv}_{\mathtt{BDH}}(A) = \Pr[\alpha = \hat{t}(P_1, P_2)^{abc}]. \tag{4}$$

Note, there are various equivalent formulations of this, since one could assume the input is in $G_1 \times G_2 \times G_1$, as if we have xP_2 then we can compute xP_1 via the isomorphism ρ.

Decisional Bilinear Diffie-Hellman Problem (DBDH). Consider Γ and the following two sets

$$\mathcal{D}_\Gamma = \{(aP_1, bP_2, cP_1, \hat{t}(P_1, P_2)^{abc}) : \ a, b, c \in [1, \ldots, q]\},$$
$$\mathcal{R}_\Gamma = G_1 \times G_2 \times G_1 \times G_T.$$

The goal of an adversary is to be able to distinguish between the two sets. This idea is defined by the following game.

1. $a, b, c \leftarrow \mathbb{F}_q^*$.
2. $d \leftarrow \{0, 1\}$.
3. If $d = 0$ then $\alpha \leftarrow G_T$.
4. Else $\alpha \leftarrow \hat{t}(P_1, P_2)^{abc}$.
5. $d' \leftarrow A(aP_1, bP_2, cP_1, \alpha, \Gamma)$.

We define the advantage of such an adversary by

$$\mathrm{Adv}_{\mathtt{DBDH}}(A) = |2 \Pr[d = d'] - 1|.$$

The Gap Bilinear Diffie–Hellman (GBDH) problem. Informally, the *gap bilinear Diffie-Hellman problem* is the problem of solving the BDH problem with the help of an oracle to solve the DBDH problem. The use of such relative or "gap" problems was first proposed by Okamoto and Pointcheval [12].

Let $\mathcal{O}$ be an oracle that, given $\beta \in \mathcal{R}_\Gamma$, returns 1 if $\beta \in \mathcal{D}_\Gamma$, and 0 otherwise. For an algorithm A, the advantage in solving the GBDH problem, which we denote by $\mathrm{Adv}_{\mathtt{GBDH}}(A, q_G)$, is defined as in (4) except that A is granted oracle access to $\mathcal{O}$ and can makes at most q_G queries.

6.2 The Boneh–Franklin Construction

Boneh and Franklin give two ID-based encryption schemes, one called BasicIdent satisfying the ID-OW-CPA security definition, the other called FullIdent satisfying the ID-IND-CCA2 definition. In our constructions we will use the BasicIdent system, but will then compare the construction with the FullIdent system.

For a group description Γ we will need to define cryptographic hash functions:

$$H_1 : \{0, 1\}^* \longrightarrow G_2,$$
$$H_2 : G_T \longrightarrow \{0, 1\}^n,$$
$$H_3 : \{0, 1\}^* \longrightarrow \mathbb{F}_q^*$$

for an integer n corresponding to the length of messages to be encrypted.

In both schemes the trust authorities keys are given by $M_{\mathfrak{pk}} = M_{\mathfrak{sk}} \cdot P_1$ for $M_{\mathfrak{sk}} \in \mathbb{F}_q^*$ chosen uniformly at random, and the user private key $D_{\mathtt{ID}}$ for identity $\mathtt{ID}$ is $D_{\mathtt{ID}} = M_{\mathfrak{sk}} \cdot H_1(\mathtt{ID})$.

BasicIdent: We define $\mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}}) = \{0,1\}^n$, $\mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pk}}) = G_1 \times \{0,1\}^n$ and $\mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}}) = \mathbb{F}_q^*$. We then define

$\mathbb{E}_{\mathtt{ID}}^{\mathtt{BasicIdent}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; r)$
- $U \leftarrow rP_1$.
- $Q_{\mathtt{ID}} \leftarrow H_1(\mathtt{ID})$.
- $\alpha \leftarrow \hat{t}(M_{\mathfrak{pk}}, Q_{\mathtt{ID}})^r$.
- $V \leftarrow m \oplus H_2(\alpha)$.
- Return (U, V).

$\mathbb{D}_{\mathtt{ID}}^{\mathtt{BasicIdent}}(D_{\mathtt{ID}}, c)$
- $(U, V) \leftarrow c$.
- $\alpha \leftarrow \hat{t}(U, D_{\mathtt{ID}})$.
- $m \leftarrow V \oplus H_2(\alpha)$.
- Return m.

Note that $\gamma(M_{\mathfrak{pk}})/|\mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})|$ for BasicIdent is equal to $1/q \approx 2^{-t}$.

Boneh and Franklin prove the following security result about BasicIdent.

Theorem 2. *Let H_1 and H_2 be modelled as random oracles and suppose A is an adversary against BasicIdent making at most q_X private key extraction queries and q_2 queries of H_2 then there is an algorithm B with essentially the same running time of A such that*

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-OW-CPA}}(A) \leq e \cdot (1 + q_X) \cdot q_2 \cdot \left(\mathrm{Adv}_{\mathtt{BDH}}(B) + \frac{1}{q_2 \cdot 2^n} \right).$$

Here, and in theorems 4 and 3 below, $e \approx 2.71$ is the base of the natural logarithm.

FullIdent: Although not explicitly defined in [4], there are in fact two variants of FullIdent mentioned in [4]. We shall present both here:

FullIdent-1: We define $\mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}}) = \{0,1\}^n$, $\mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pk}}) = G_1 \times \{0,1\}^n \times \{0,1\}^{|m|}$, $\mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}}) = \{0,1\}^n$. We require all the hash functions used by BasicIdent and in addition we require a cryptographic hash function $H_4 : \{0,1\}^n \longrightarrow \{0,1\}^{|m|}$, to encrypt messages of length $|m|$. We define the scheme as follow.

$\mathbb{E}_{\mathtt{ID}}^{\mathtt{FullIdent-1}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; \sigma)$
- $r \leftarrow H_3(\sigma || m)$.
- $(U, V) \leftarrow \mathbb{E}_{\mathtt{ID}}^{\mathtt{BasicIdent}}(\mathtt{ID}, M_{\mathfrak{pk}}, \sigma; r)$.
- $W \leftarrow m \oplus H_4(\sigma)$.
- Return (U, V, W).

$\mathbb{D}_{\mathtt{ID}}^{\mathtt{FullIdent-1}}(D_{\mathtt{ID}}, c)$
- $(U, V, W) \leftarrow c$.
- $\sigma \leftarrow \mathbb{D}_{\mathtt{ID}}^{\mathtt{BasicIdent}}(D_{\mathtt{ID}}, (U, V))$.
- $m \leftarrow W \oplus H_4(\sigma)$.
- $r \leftarrow H_3(\sigma || m)$.
- If $rP_1 \neq U$ return $\perp$.
- Return m.

Using the Fujisaki–Okamoto transform the following result is proved for the scheme FullIdent-1 in [4].

Theorem 3. *Let H_1, H_2, H_3 and H_4 be modelled as random oracles and suppose A is an adversary against FullIdent-1 making at most q_X private key extraction queries, q_D decryption queries and q_2, q_3 and q_4 queries of H_2, H_3 and H_4 respectively. Then there is an algorithm B, running in time essentially that of A, such that*

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-IND-CCA2}}(A) \leq c_1 \cdot \left(\frac{1}{c_2} \left(c_3 \cdot \left(q_2 \cdot \mathrm{Adv}_{\mathtt{BDH}}(B) + \frac{1}{2^n} \right) + 1 \right) - 1 \right)$$

where $c_1 = e \cdot (1 + q_X + q_D)$, $c_2 = (1 - 2/q)^{q_D}$ and $c_3 = 2 \cdot (q_3 + q_4)$.

FullIdent-2: Let $\mathbb{E}_{\mathtt{SK}}, \mathbb{D}_{\mathtt{SK}}$ denote a symmetric encryption algorithm, with message space $\mathbb{M}_{\mathtt{SK}}(t)$, ciphertext space $\mathbb{C}_{\mathtt{SK}}(t)$ and key space $\mathbb{K}_{\mathtt{SK}}(t)$ for security parameter t. We assume that this symmetric algorithm is secure in the sense of FG-PA. The following scheme is mentioned but not analysed in [4]. We define

$$\mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}}) = \mathbb{M}_{\mathtt{SK}}(t), \mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pk}}) = G_1 \times \{0,1\}^n \times \mathbb{C}_{\mathtt{SK}}(t) \text{ and } \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}}) = \{0,1\}^n$$

and we let

$$H_5 : \{0,1\}^* \longrightarrow \mathbb{K}_{\mathtt{SK}}(t)$$

denote a cryptographic hash function.

We then define

$\mathbb{E}_{\mathtt{ID}}^{\mathtt{FullIdent-2}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; \sigma)$
- $r \leftarrow H_3(\sigma || m)$.
- $(U, V) \leftarrow \mathbb{E}_{\mathtt{ID}}^{\mathtt{BasicIdent}}(\mathtt{ID}, M_{\mathfrak{pk}}, \sigma; r)$.
- $W \leftarrow \mathbb{E}_{\mathtt{SK}}(H_5(\sigma), m)$.
- Return (U, V, W).

$\mathbb{D}_{\mathtt{ID}}^{\mathtt{FullIdent-2}}(D_{\mathtt{ID}}, c)$
- $(U, V, W) \leftarrow c$.
- $\sigma \leftarrow \mathbb{D}_{\mathtt{ID}}^{\mathtt{BasicIdent}}(D_{\mathtt{ID}}, (U, V))$.
- $m \leftarrow \mathbb{D}_{\mathtt{SK}}(H_5(\sigma), W)$.
- $r \leftarrow H_3(\sigma || m)$.
- If $rP_1 \neq U$ return $\perp$.
- Return m.

Using the Fujisaki–Okamoto transform the following result can be proved for the scheme FullIdent-2 using the same argument as is used in [4] for FullIdent-1.

Theorem 4. *If H_1, H_2, H_3 and H_5 are modelled as random oracles and suppose A is an adversary against FullIdent-2 making at most q_X private key extraction queries, q_D decryption queries and q_2, q_3 and q_5 queries of H_2, H_3 and H_5 respectively. Then there are algorithms B_1 and B_2, running in time essentially that of A, such that*

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-IND-CCA2}}(A) \leq c_1 \cdot \left(\frac{1}{c_2} \left(\begin{array}{r} c_3 \cdot \left(q_2 \cdot \mathrm{Adv}_{\mathtt{BDH}}(B_1) + \frac{1}{2^n} \right) \\ + \mathrm{Adv}_{\mathtt{SK}}^{\mathtt{FG-PA}}(B_2) + 1 \end{array} \right) - 1 \right)$$

where $c_1 = e \cdot (1 + q_X + q_D)$, $c_3 = 2 \cdot (q_3 + q_5)$ and

$$c_2 = \left(1 - 2 \left(q_2 \cdot \mathrm{Adv}_{\mathtt{BDH}}(B_1) + \frac{1}{2^n} \right) - 2 \cdot \mathrm{Adv}_{\mathtt{SK}}^{\mathtt{FG-PA}}(B_2) - \frac{1}{q} - \frac{1}{|\mathbb{M}_{\mathtt{SK}}(M_{\mathfrak{pk}})|} \right)^{q_D}.$$

7 ID-KEM Constructions

We present three constructions, the first two are based on specific computational problems, namely variants of the bilinear Diffie–Hellman problem on elliptic curves, the third construction is generic. Our first construction is due to Lynn [10]. We note that Lynn only mentions this ID-KEM construction in passing and did not give a security definition or proof.

In this section we present a security proof of Lynn's construction by proving security under the GBDH problem. The second construction is a variant on Lynn's construction and is based on the fifth of Dent's generic constructions [7], this is secure under the standard BDH problem. Our third construction takes any probabilistic OW-ID-CPA secure ID-based encryption scheme and produces an IND-ID-CCA2 secure ID-KEM.

The second construction can be considered as either a strengthening of Lynn's construction or as an optimisation of the third generic construction, when in the third construction we use the BasicIdent algorithm from [4]. The second construction is less efficient than Lynn's method, but its security rests on the BDH problem rather than the GBDH problem. Since the BDH problem is a more natural and well studied problem than the GBDH problem we see this extra confidence in the security as more than offsetting the slight performance reduction compared to Lynn's construction.

Our first two constructions follow the standard setup for ID-based systems built on pairings, as first explained in [4]. For completeness, we recall the setup here. We let H_1, H_2, H_3, H_4 and H_5 denote cryptographic hash functions as in Section 6.2. However, now we let $(\mathbb{E}_{\mathtt{DEM}}, \mathbb{D}_{\mathtt{DEM}})$ denote a DEM, and so the codomain of H_5 is the key space of the DEM.

For our first two constructions we adopt the initial set up as in the Boneh–Franklin scheme by defining the trust authorities keys as $M_{\mathfrak{pk}} = M_{\mathfrak{sk}} \cdot P_1$ for $M_{\mathfrak{sk}} \in \mathbb{F}_q^*$ chosen uniformly at random, and the user private key $D_{\mathtt{ID}}$ corresponding to identity $\mathtt{ID}$ is $D_{\mathtt{ID}} = M_{\mathfrak{sk}} \cdot H_1(\mathtt{ID})$.

7.1 Construction 1 :

This is the construction of Lynn [10].

$\mathbb{E}_{\mathtt{ID-KEM}}(\mathtt{ID}, M_{\mathfrak{pk}})$	$\mathbb{D}_{\mathtt{ID-KEM}}(D_{\mathtt{ID}}, C)$
– $r \leftarrow \mathbb{F}_q^*$.	– $T \leftarrow \hat{t}(C, D_{\mathtt{ID}})$.
– $C \leftarrow rP_1$.	– $k \leftarrow H_5(T)$.
– $Q_{\mathtt{ID}} \leftarrow H_1(\mathtt{ID})$.	– Return k.
– $T \leftarrow \hat{t}(M_{\mathfrak{pk}}, Q_{\mathtt{ID}})^r$.	
– $k \leftarrow H_5(T)$.	
– Return (k, C).	

7.2 Construction 2 :

$\mathbb{E}_{\mathtt{ID-KEM}}(\mathtt{ID}, M_{\mathfrak{pk}})$
- $m \leftarrow \{0,1\}^n$.
- $r \leftarrow H_3(m)$.
- $U \leftarrow rP_1$.
- $Q_{\mathtt{ID}} \leftarrow H_1(\mathtt{ID})$.
- $V \leftarrow m \oplus H_2(\hat{t}(M_{\mathfrak{pk}}, Q_{\mathtt{ID}})^r)$.
- $c \leftarrow (U, V)$.
- $k \leftarrow H_5(m)$.
- Return (k, c).

$\mathbb{D}_{\mathtt{ID-KEM}}(D_{\mathtt{ID}}, c)$
- $(U, V) \leftarrow c$.
- $m \leftarrow V \oplus H_2(\hat{t}(U, D_{\mathtt{ID}}))$.
- $r \leftarrow H_3(m)$.
- If $U \neq rP_1$ then output $\perp$ and halt.
- $k \leftarrow H_5(m)$.
- Return k.

7.3 Construction 3 :

Here we take a generic probabilistic ID-based encryption scheme, with encryption algorithm $\mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; r)$ and associated decryption algorithm $\mathbb{D}_{\mathtt{ID}}(D_{\mathtt{ID}}, c)$, where $D_{\mathtt{ID}}$ is the output from the extraction algorithm $\mathbb{X}_{\mathtt{ID-KEM}}(M_{\mathfrak{sk}}, \mathtt{ID})$. We assume the message space of $\mathbb{E}_{\mathtt{ID}}$ is given by $\mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$ and the space of randomness is given by $\mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})$. We assume a cryptographic hash function $H_3' : \{0,1\}^* \longrightarrow \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.

In the following construction we make no assumption on how such an ID-based scheme is constructed only that it exists. In practice one can take the method BasicIdent from [4].

$\mathbb{E}_{\mathtt{ID-KEM}}(\mathtt{ID}, M_{\mathfrak{pk}})$
- $m \leftarrow \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.
- $r \leftarrow H_3'(m)$.
- $c \leftarrow \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; r)$.
- $k \leftarrow H_5(m)$.
- Return (k, c).

$\mathbb{D}_{\mathtt{ID-KEM}}(D_{\mathtt{ID}}, c)$
- $m \leftarrow \mathbb{D}_{\mathtt{ID}}(D_{\mathtt{ID}}, c)$.
- If $m = \perp$ then output $\perp$ and halt.
- $r \leftarrow H_3'(m)$.
- If $c \neq \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; r)$ then output $\perp$ and halt.
- $k \leftarrow H_5(m)$.
- Return k.

7.4 Construction 1 : Security Proof

Theorem 5. *If the GBDH problem is hard and H_5 and H_1 are modelled as random oracles then Construction 1 is secure against adaptive chosen ciphertext attack.*

Specifically, if A is a PPT algorithm that breaks the ID-KEM of Construction 1 using a chosen ciphertext attack, then there exists a PPT algorithm B, with

$$\mathrm{Adv}_{\mathtt{ID-KEM}}^{\mathtt{ID-IND-CCA2}}(A) \leq 2(q_1 + q_X + q_D) \cdot \mathrm{Adv}_{\mathtt{GBDH}}(B, q_5(2q_D + 1)) + \frac{2q_5}{q}.$$

where q_1, q_5, q_D and q_X are the number of queries made by A to H_1, H_5, the decryption oracle and the private key extraction oracle respectively.

Proof. Let $\texttt{S}$ be the event that A correctly determines the bit chosen by the challenge encryption oracle during its attack. Let $\texttt{ID}^*$ be the identity chosen by A during its attack and let T^* be the output of the bilinear map generated by A's challenge encapsulation oracle at the third stage of the encapsulation process. We define the following events.

$\texttt{AskK}$: The event that A makes the query T^* to H_5 during its attack.
$\texttt{AskH}$: The event that A makes the query $\texttt{ID}^*$ to H_1 during its attack or that it makes a decryption query involving $\texttt{ID}^*$.

We have

$$\begin{aligned}\Pr[\texttt{S}] &= \Pr[\texttt{S} \wedge \texttt{AskK}] + \Pr[\texttt{S} \wedge \neg\texttt{AskK}] \\ &\leq \Pr[\texttt{S} \wedge \texttt{AskK}] + \frac{1}{2}. \end{aligned} \tag{5}$$

This follows from the fact that if A does not query T^* to H_5 then it can have no advantage. We can then express

$$\begin{aligned}\Pr[\texttt{S} \wedge \texttt{AskK}] &= \Pr[\texttt{S} \wedge \texttt{AskK} \wedge \texttt{AskH}] + \Pr[\texttt{S} \wedge \texttt{AskK} \wedge \neg\texttt{AskH}] \\ &\leq \Pr[\texttt{S} \wedge \texttt{AskK} \wedge \texttt{AskH}] + \frac{q_5}{q}. \end{aligned} \tag{6}$$

Here the inequality comes from the fact that, if A does not make the query $\texttt{ID}^*$ to H_1, then $Q_{\texttt{ID}^*}$ is completely unknown to A and hence, from A's view, there are q equally likely possibilities for T^*.

Using the definition of A's advantage with (5) and (6) we have

$$\mathrm{Adv}_{\texttt{ID-KEM}}^{\texttt{ID-IND-CCA2}}(A) \leq 2\Pr[\texttt{S} \wedge \texttt{AskK} \wedge \texttt{AskH}] + \frac{2q_5}{q}. \tag{7}$$

To complete the proof we argue that, in the event $\texttt{S} \wedge \texttt{AskK} \wedge \texttt{AskH}$, there is an adversary B that solves the GBDH problem with probability at least $1/(q_1 + q_X + q_D)$ while making at most $q_5(2q_D + 1)$ queries to its DBDH oracle.

Suppose that we have a BDH problem instance (aP_2, bP_2, cP_2) that we wish to solve. We construct an algorithm B to do this by using A. We set the value of $M_{\mathfrak{pk}}$ for algorithm A to be $cP_1 = \rho(cP_2)$.

We maintain three lists $L_1 \subset \{0,1\}^* \times G_2 \times \mathbb{F}_q$, $L_5 \subset G_T \times \mathbb{K}_{\texttt{DEM}}(t)$ and $L_D \subset G_1 \times G_2 \times G_1 \times \mathbb{K}_{\texttt{DEM}}(t)$ that allow us to provide consistent answers to A's oracle calls. We simulate these oracles as follows.

- H_1 **Queries:** We choose an index i from $[1, \ldots, q_1 + q_X + q_D]$. We respond to queries to H_1 as follows (assuming that the query is not already present in L_1).
  - If we are responding to the i-th query, which we denote $\texttt{ID}'$, we respond with bP_2 and add $(\texttt{ID}', bP_2, \perp)$ to L_1.
  - If we are responding to any other query $\texttt{ID}$ we choose $x \leftarrow [1, \ldots, q]$ and respond with xP_2. We also add $(\texttt{ID}, xP_2, x)$ to L_1.

– H_5 **Queries:** We respond to a (non-repeat) query T to H_5 as follows. We first search L_D for an entry (C, bP_2, cP_1, k) such that the DBDH oracle returns 1 when queried with (C, bP_2, cP_1, T). If such an entry is found we respond with k. Otherwise, we call the DBDH oracle with (aP_1, bP_2, cP_1, T). If the oracle returns 1 we output T and terminate the simulation – the solution to the BDH problem has been found. If the oracle returns 0 we simply choose a random response k_T to respond with and update L_5 by adding (T, k_T).
– $\mathbb{X}_{\mathtt{ID-KEM}}$ **Queries:** We respond to an extraction oracle query $\mathtt{ID}$ made by A as follows. We first call the H_1-oracle with $\mathtt{ID}$. We then search the list L_1 for the entry corresponding to $\mathtt{ID}$. If this entry has third component equal to $\perp$ then we abort. Otherwise we obtain the triple $(\mathtt{ID}, xP_2, x)$. We then respond with $D_{\mathtt{ID}} \leftarrow x(cP_2)$.
– $\mathbb{D}_{\mathtt{ID-KEM}}$ **Queries:** We respond to a decapsulation oracle query $(\mathtt{ID}, C)$ as follows. We first call the oracle H_1. We then look for the entry corresponding to $\mathtt{ID}$ in L_1. There are two possibilities.
  - If $\mathtt{ID} = \mathtt{ID}'$, so the response we get from H_1 is bP_2. We then search L_5 for an element (T, k_T) such that the DBDH oracle returns 1 when queried with (C, bP_2, cP_1, T). If such an element is found we respond with k_T. If no such element is found we choose a response k at random and add (C, bP_2, cP_1, k) to L_D.
  - If $\mathtt{ID} \neq \mathtt{ID}'$ we obtain the entry corresponding entry $(\mathtt{ID}, xP_2, x)$ from L_1 and we compute $D_{\mathtt{ID}} \leftarrow x(cP_2)$. We then compute $T \leftarrow \hat{t}(C, D_{\mathtt{ID}})$ and call H_5 with T. We relay the response to A.

– **Challenge Query:** When A makes its challenge encryption oracle query we choose k' at random and we set $C^* \leftarrow aP_1$. We respond with (k', C^*).

In the above simulation what we want is that $H_1(\mathtt{ID}^*) = bP_1$. The oracle H_1 is called maximum of $(q_1 + q_X + q_D)$ times in the simulation above; therefore, we achieve this with probability at least $1/(q_1 + q_X + q_D)$ in the event $\mathtt{AskH}$. Since by construction we also have $C^* = aP_1$ and $M_{\mathtt{pk}} = cP_1$ we conclude that, in the event $\mathtt{AskH}$, with probability at least $1/(q_1 + q_X + q_D)$, the value of T^* implicit in the challenge ciphertext is $\hat{t}(P_1, P_2)^{abc}$; this is the value which we wish to compute. We conclude that

$$\Pr[\mathtt{S} \wedge \mathtt{AskK} \wedge \mathtt{AskH}] \leq (q_1 + q_X + q_D) \cdot \mathrm{Adv}_{\mathtt{GBDH}}(B, q_5(2q_D + 1)). \qquad (8)$$

The result follows from (7) and (8).

7.5 Construction 3 : Security Proof

We first give the security proof for construction 3, as the security proof for construction 2 will then follow immediately from Theorem 2.

Theorem 6. *If $\mathbb{E}_{\mathtt{ID}}$ is an ID-OW-CPA secure ID-based encryption scheme and H_3' and H_5 are modelled as random oracles then Construction 3 is secure against adaptive chosen ciphertext attack.*

Specifically, if A is a PPT algorithm that breaks the ID-KEM of Construction 3 using a chosen ciphertext attack, then there exists a PPT algorithm B, with

$$\mathrm{Adv}_{\mathtt{ID-KEM}}^{\mathtt{ID-IND-CCA2}}(A) \leq 2(q_3 + q_5 + q_D) \cdot \mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-OW-CPA}}(B) + \frac{2q_D\gamma(M_{\mathfrak{pt}})}{|\mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pt}})|},$$

where q_3, q_5 and q_D are the number of queries made by A to H_3', H_5 and the decryption oracle respectively, and $\gamma(M_{\mathfrak{pt}})$ is as in (1).

Proof. Let A denote an ID-IND-CCA2 adversary against Construction 3, as specified in the statement of the theorem. Security is proved via a sequence of games $\mathtt{Game}_0, \mathtt{Game}_1$. In each game we let $\mathtt{S}_i$ denote the event that $b = b'$. We let $\mathtt{Game}_0$ denote the original attack game so that

$$\mathrm{Adv}_{\mathtt{ID-KEM}}^{\mathtt{ID-IND-CCA2}}(A) = |2\Pr[\mathtt{S}_0] - 1|.$$

Let $\mathtt{Game}_1$ be the same as $\mathtt{Game}_0$ except we simulate the H_3', H_5, extraction and decapsulation queries as described below.

- H_3' **Queries:** We maintain a list L_3 which contains at most q_3 pairs (x, h_3). On input of x, if $(x, h_3) \in L_3$ then we return h_3, otherwise we select h_5 at random append (x, h_3) to the list and return h_3.
- H_5 **Queries:** We maintain a list L_5 of length at most $q_5 + q_D$ pairs (x, h_5). On input of x, if $(x, h_5) \in L_3$ then we return h_5, otherwise we select h_3 at random append (x, h_5) to the list and return h_5.
- $\mathbb{X}_{\mathtt{ID-KEM}}$ **Queries:** These are answered as in the genuine attack game $\mathtt{Game}_0$.
- $\mathbb{D}_{\mathtt{ID-KEM}}$ **Queries:** We respond to such a query $(c, \mathtt{ID})$ as follows:
  - If $\mathtt{ID} \neq \mathtt{ID}^*$ then we compute the private key $D_{\mathtt{ID}}$ via the $\mathbb{X}_{\mathtt{ID-KEM}}$ oracle and respond the real decapsulation oracle would.
  - If $\mathtt{ID} = \mathtt{ID}^*$ but $c \neq c^*$ we check for each pair $(x, h_3) \in L_3$, if

    $$\mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pt}}, x; h_3) = c.$$

    If such a pair exists then run the simulator of H_5 on input x so as to obtain h_5, we then return h_5. If no such pair exists we return $\perp$.

Note that $\mathtt{Game}_0$ and $\mathtt{Game}_1$ are identical, as A operates in the random oracle model, except if a decapsulation query with input c where $x = \mathbb{D}_{\mathtt{ID-KEM}}(D_{\mathtt{ID}}, c)$ and $c = \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pt}}, x; H_3'(x))$ is made but no H_3' queries on input x have been made. Call this event $\mathtt{E}$, hence

$$\Pr[\mathtt{S}_0 \wedge \neg\mathtt{E}] = \Pr[\mathtt{S}_1 \wedge \neg\mathtt{E}].$$

From Lemma 1 we have

$$|\Pr[\mathtt{S}_0] - \Pr[\mathtt{S}_1]| \leq \Pr[\mathtt{E}] \leq \frac{q_D\gamma(M_{\mathfrak{pt}})}{\mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pt}})}.$$

This follows from the fact that, for each decryption query for which $\mathtt{ID} = \mathtt{ID}^*$ and $c \neq c^*$ but for no pair $(x, h_3) \in L_3$ do we have $\mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pt}}, x; h_3) = c$, there is probability at most $\gamma(M_{\mathtt{p}\mathfrak{k}})$ that for r chosen at random there is an m such that $\mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathtt{p}\mathfrak{k}}, m; r) = c$; moreover, there are at most q_D such queries.

Let m^* denote the hidden value encrypted by c^*. We let $\mathtt{E}'$ denote the event that the attacker queries either H_3' or H_5 with m^*. Then, since we are in the random oracle model,

$$\begin{aligned}\Pr[\mathtt{S}_1] &= \Pr[\mathtt{S}_1 \wedge \mathtt{E}'] + \Pr[\mathtt{S}_2 \wedge \neg\mathtt{E}'] \\ &\leq \Pr[\mathtt{S}_1 \wedge \mathtt{E}'] + \frac{1}{2} \\ &\leq \Pr[\mathtt{S}_1 | \mathtt{E}'] + \frac{1}{2}.\end{aligned}$$

The theorem follows if we can describe an algorithm B with

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-OW-CPA}}(B) = \Pr[\mathtt{S}_1 | \mathtt{E}']/(q_3 + q_5 + q_D).$$

Algorithm B operates as follows. On input of $M_{\mathtt{p}\mathfrak{k}}$ it simply relays this onto algorithm A's first stage. Algorithm A then responds with an identity $\mathtt{ID}^*$. This value of $\mathtt{ID}^*$ is passed onto B's challenger who responds with an encryption c^* of a random message m^*. The value of c^* is then passed onto A's second stage, along with a random key k^* from the codomain of H_5. All oracle queries are answered as in $\mathtt{Game}_2$, except the private key extraction queries which are now answered using the oracle given to algorithm B. At the end of the game algorithm B picks a random value which has been input into H_3' or H_5 from the L_3 or L_5 list and returns this as its guess for m^*. It is clear that $\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-OW-CPA}}(B) = \Pr[\mathtt{S}_1 | \mathtt{E}']/(q_3 + q_5 + q_D)$ and so the result follows.

7.6 Construction 2 : Security Proof

Combining Theorem 2 and Theorem 6 we obtain.

Theorem 7. *If H_2, H_3 and H_5 are modelled as random oracles then Construction 2 is secure against adaptive chosen ciphertext attack.*

Specifically, if A is a PPT algorithm that breaks the ID-KEM of Construction 2 using a chosen ciphertext attack, then there exists a PPT algorithm B, with

$$\mathrm{Adv}_{\mathtt{ID-KEM}}^{\mathtt{ID-IND-CCA2}}(A) \leq 2e \cdot q_2 \cdot (q_3 + q_5 + q_D)(1 + q_X) \cdot \left(\mathrm{Adv}_{\mathtt{BDH}}(B) + \frac{1}{q_2 \cdot 2^n}\right) + \frac{2q_D}{q},$$

where q_2, q_3, q_5, q_D and q_X are the number of queries made by A to H_2, H_3, H_5, the decryption oracle and the private key extraction oracle respectively.

8 Comparison with FullIdent

In the following table we count the various operation counts for encryption and decryption of our ID-based encryption schemes. Note, we make no-distinction as to in which group exponentiations occur since one can select the group so as to make the operation more efficient in any given implementation. We let C-1, denote our Lynn's KEM construction combined with a DEM, and C-2 denote our second construction combined with a DEM, which is an optimised version of construction three.

	Pairings		Exp's		Hash Fncs		Message
Scheme	$\mathbb{E}_{\mathtt{ID}}$	$\mathbb{D}_{\mathtt{ID}}$	$\mathbb{E}_{\mathtt{ID}}$	$\mathbb{D}_{\mathtt{ID}}$	$\mathbb{E}_{\mathtt{ID}}$	$\mathbb{D}_{\mathtt{ID}}$	Size
FullIdent-1	1	1	2	1	4	3	$\lvert G_1\rvert + n + \lvert m\rvert$
FullIdent-2	1	1	2	1	4	3	$\lvert G_1\rvert + n + \lvert\mathbb{E}_{\mathtt{SK}}(m)\rvert$
C-1	1	1	2	0	2	1	$\lvert G_1\rvert + \lvert\mathbb{E}_{\mathtt{DEM}}(m)\rvert$
C-2	1	1	2	1	4	3	$\lvert G_1\rvert + n + \lvert\mathbb{E}_{\mathtt{DEM}}(m)\rvert$

From the table we see that the KEM/DEM approach with Lynn's method is marginally more efficient than the technique of Boneh and Franklin, or the ID-KEM approach of our construction two. We also note that FullIdent-2 requires requires a less stringent definition of security of the symmetric encryption scheme, this is because chosen-ciphertext security is provided by the Fujisaki–Okamoto transform rather than the CCA security of the symmetric cipher.

We also need to compare the tightness of the security guarantees offered by the various constructions. To do this we make an explicit numerical comparison. We take a security parameter where for our bilinear groups which results in a value of $q \approx 2^{\lambda}$. We assume adversaries against our schemes exists which make at most $q_X \approx 2^{32}$ calls to their key extraction oracles and at most $q_D \approx 2^{32}$ calls to their decryption oracles. For our GBDH adversary we also limit the number of DBDH queries to 2^{32}. We also assume that the number of hash function queries is bounded, for each hash function, by approximately 2^{32}.

We then obtain the following tightness results:

FullIdent-1 :

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-IND-CCA2}}(A) \leq 2^{99} \cdot \mathrm{Adv}_{\mathtt{BDH}}(B) + 2^{67-n}.$$

FullIdent-2 :

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-IND-CCA2}}(A) \leq 2^{99} \cdot \mathrm{Adv}_{\mathtt{BDH}}(B) + 2^{67-n} + 2^{34}\mathrm{Adv}_{\mathtt{SK}}^{\mathtt{FG-PA}}(C)$$

C-1 :

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-IND-CCA2}}(A) \leq 2^{34} \cdot \mathrm{Adv}_{\mathtt{GBDH}}(B, 2^{32}) + 2^{33-\lambda} + \mathrm{Adv}_{\mathtt{FG-DEM}}^{\mathtt{CCA}}(C).$$

C-2 :

$$\mathrm{Adv}_{\mathtt{ID-KEM}}^{\mathtt{ID-IND-CCA2}}(A) \leq 2^{100}\mathrm{Adv}_{\mathtt{BDH}}(B) + 2^{67-n} + 2^{33-\lambda} + \mathrm{Adv}_{\mathtt{FG-DEM}}^{\mathtt{CCA}}(C).$$

Suppose we use an elliptic curve system to implement our bilinear groups, with MOV embedding degree d. We then know that there exists a sub-exponential algorithm to solve the BDH and GBDH algorithms with running time essentially [14]

$$L_{2^{d\lambda}}(1/3, (64/9)^{1/3}) = \exp\left((64/9)^{1/3}(\log 2^{d\lambda})^{1/3}(\log\log 2^{d\lambda})^{2/3}\right).$$

In addition there is an exponential algorithm [13] with running time essentially

$$2^{\lambda/2}.$$

Hence, we can estimate a lower bound for $\mathrm{Adv}_{\mathtt{BDH}}(B)$ and $\mathrm{Adv}_{\mathtt{GBDH}}(B,c)$ via

$$1/\min(L_{2^{d\lambda}}(1/3,(64/9)^{1/3}),2^{\lambda/2}).$$

Suppose we wished to obtain a security guarantee of an advantage of the adversary A against our ID-based scheme of 2^{-80}, in addition we assume that our symmetric cipher is chosen so that it can guarantee an advantage of $2^{-\mu}$. We now look at what this implies for the parameters of the pairing based parts of the scheme.

FullIdent-1 :

$$n \geq 147 \text{ and } \lambda \geq \max(358, 5700/d) \approx 950 \text{ (if } d=6 \text{).}$$

FullIdent-2 :

$$n \geq 147, \quad \mu \geq 114 \text{ and } \lambda \geq \max(358, 5700/d) \approx 950 \text{ (if } d=6 \text{).}$$

C-1 :

$$\mu \geq 80 \text{ and } \lambda \geq \max(112, 226, 1900/d) \approx 316 \text{ (if } d=6 \text{).}$$

C-2 :

$$n \geq 147, \quad \mu \geq 80 \text{ and } \lambda \geq \max(113, 360, 5740/d) \approx 957 \text{ (if } d=6 \text{).}$$

Notice, the weaker security requirement on the symmetric cipher in FullIdent-2, is not such an advantage when one considers the tightness result. Also note, the tightness of the reduction of our Construction 1, needs to be balanced against the fact that one is reducing to a possibly easier problem.

9 CL-KEM Construction

Our generic construction of a CL-KEM can now be given as follows.

- Let $(\mathbb{G}_{\mathtt{PK}}, \mathbb{E}_{\mathtt{PK}}, \mathbb{D}_{\mathtt{PK}})$ be a OW-CPA^{++} secure public key encryption algorithm which 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 ID-based encryption algorithm.

We define our seven algorithms as follows. The algorithm $\mathbb{G}_{\mathtt{CL-KEM}}$ is defined to be equal to $\mathbb{G}_{\mathtt{ID}}$. The algorithm Partial-Private-Key-Extraction returns $D_{\mathtt{ID}}$: 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 $\mathbb{G}_{\mathtt{PK}}$. The algorithm Set-Private-Key returns the pair $S_A = (D_{\mathtt{ID}}, \mathfrak{sk}_A)$. Finally, using a hash function H, encapsulation and decapsulation work as follows.

$\mathbb{E}_{\mathtt{CL-KEM}}(\mathfrak{pk}_A, \mathtt{ID}_A, M_{\mathfrak{pk}})$:
- $m_1 \leftarrow \mathbb{M}_{\mathtt{PK}}(\mathfrak{pk})$, $r_1 \leftarrow \mathbb{R}_{\mathtt{PK}}(\mathfrak{pk})$.
- $m_2 \leftarrow \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$,
  $r_2 \leftarrow \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.
- $c_1 \leftarrow \mathbb{E}_{\mathtt{PK}}(\mathfrak{pk}_A, m_1; r_1)$.
- $c_2 \leftarrow \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}_A, M_{\mathfrak{pk}}, m_2; r_2)$.
- $k \leftarrow H(c_1, \mathfrak{pk}_A, m_1, m_2)$.
- $c \leftarrow (c_1, c_2)$.

$\mathbb{D}_{\mathtt{CL-KEM}}(S_A, c)$:
- $(c_1, c_2) \leftarrow c$.
- $(D_{\mathtt{ID}}, \mathfrak{sk}_A) \leftarrow S_A$.
- $m_1 \leftarrow \mathbb{D}_{\mathtt{PK}}(\mathfrak{sk}_A, c_1)$.
- If $m_1 = \perp$ then return $\perp$.
- $m_2 \leftarrow \mathbb{D}_{\mathtt{ID}}(D_{\mathtt{ID}}, c_2)$.
- If $m_2 = \perp$ then return $\perp$.
- $k \leftarrow H(c_1, \mathfrak{pk}_A, m_1, m_2)$.

9.1 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 ID-based component of the scheme and on the security of the public key component.

Theorem 8. *Our generic CL-KEM construction is secure against Type-I adversaries in the random oracle, assuming the identity-based encryption 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-CCA2 security of the identity-based encryption 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 the the random oracle H. Both B_1 and B_2 run for essentially the same time as A and they are such that

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

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

We let $\mathtt{ID}^*$ 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 $\mathtt{ID}^*$ at the time the challenge ciphertext is created, and let m_2^* denote the message encrypted in c_2^* under $\mathtt{ID}^*$. Let k_0^* denote a random key selected from the codomain 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 $\mathtt{Game}_0$ and $\mathtt{Game}_1$. In each game $\mathtt{Game}_i$ we let $\mathtt{S}_i$ denote the event that $b' = b$. We let $\mathtt{Game}_0$ denote the original attack game and so by definition

$$\mathrm{Adv}_{\mathtt{CL-KEM}}^{\mathtt{Type-I}}(A) = |2\Pr[\mathtt{S}_0] - 1|. \tag{9}$$

$\mathtt{Game_1}$**:** 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 $q_{PK}+q_{SK}+q_{PX}+q_R+q_D$. 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 $(\mathtt{ID}, \mathfrak{pk}, \mathfrak{sk})$ onto 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 $\mathtt{ID}$ with $\mathfrak{pk}'$ then we search the L_X list for an entry corresponding to $\mathtt{ID}$ and replace this entry with $(\mathtt{ID}, \mathfrak{pk}', \perp)$. If no such entry exists then we add $(\mathtt{ID}, \mathfrak{pk}', \perp)$ 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 algorithm is verifiable.
  - Otherwise we generate k at random from the codomain 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 the component 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 algorithm is verifiable. If such an entry exists then we return k. Otherwise we generate k at random from the codomain 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

$$\Pr[\mathtt{S}_0] = \Pr[\mathtt{S}_1]. \tag{10}$$

Before proceeding we define three events.

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

We immediately have the following.

$$\begin{aligned}\Pr[\mathtt{S}_1] &= \Pr[\mathtt{S}_1 \wedge \mathtt{Replace}] + \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Replace}] \\ &= \Pr[\mathtt{S}_1|\mathtt{Replace}]\Pr[\mathtt{Replace}] \\ &\quad + \Pr[\mathtt{S}_1|\neg\mathtt{Replace}](1 - \Pr[\mathtt{Replace}]).\end{aligned} \tag{11}$$

Also,

$$\Pr[\mathtt{S}_1|\mathtt{Replace}] = \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Extract}|\mathtt{Replace}]. \tag{12}$$

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

Now,

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

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

$$\begin{aligned}\Pr[\mathtt{S}_1|\neg\mathtt{Replace}] &= \Pr[\mathtt{S}_1 \wedge \mathtt{Ask}|\neg\mathtt{Replace}] + \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Ask}|\neg\mathtt{Replace}] \\ &\leq \Pr[\mathtt{S}_1 \wedge \mathtt{Ask}|\neg\mathtt{Replace}] + \frac{1}{2}.\end{aligned} \tag{14}$$

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 identity based encryption scheme used in the construction. This algorithm runs A in a similar manner to how A is run in $\mathtt{Game}_1$. The first differences 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 codomain 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 codomain 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

$$\Pr[\mathtt{S}_1 \wedge \neg\mathtt{Extract} \wedge \mathtt{Ask} | \mathtt{Replace}] \leq (q_H + q_D)\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-OW-CCA2}}(B_1). \tag{15}$$

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 $(\mathtt{ID}, \mathfrak{pk}, \mathfrak{sk})$ onto 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 $\mathtt{Game}_1$ 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 codomain 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 $\mathtt{Game}_1$ up until the event $\mathtt{Ask}$ occurs and, moreover, if the event $\mathtt{Ask}$ occurs, B_2 succeeds. We conclude that

$$\Pr[\mathtt{S}_1 \wedge \mathtt{Ask}|\neg\mathtt{Replace}] \leq (q_{PK} + q_D + 1)\mathrm{Adv}_{\mathtt{PK}}^{\mathtt{OW-CPA}^{++}}(B_2). \tag{16}$$

The result now follows from (9), (10), (11), (12), (13), (15), (14) and (16).

9.2 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 9. *Our generic CL-KEM construction is secure against Type-II adversaries in the random oracle, 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

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

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

Security is proved via two games $\mathtt{Game}_1$ and $\mathtt{Game}_2$. We define $\mathtt{ID}^*$, $\mathfrak{pk}^*$, $c^* = (c_1^*, c_2^*)$, m_1^*, m_2^*, k_0^*, k_1^*, b, b', $\mathtt{S}_i$ and $\mathtt{Ask}$ exactly as in Theorem 8.

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

$$\mathrm{Adv}_{\mathtt{CL-KEM}}^{\mathtt{Type-II}}(A) = |2\Pr[\mathtt{S}_0] - 1|. \tag{17}$$

$\mathtt{Game}_1$**:** 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 $(\mathtt{ID}, \mathfrak{pk}, \mathfrak{sk})$ onto 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 codomain of H, we place $(k, c, \mathfrak{pk}, m_1, m_2)$ onto 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 the component c_2 to obtain m_2, since it knows the master key for the ID-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

$$\begin{aligned}
\Pr[\mathtt{S}_0] &= \Pr[\mathtt{S}_1] \\
&= \Pr[\mathtt{S}_1 \wedge \mathtt{Ask}] + \Pr[\mathtt{S}_1 \wedge \neg\mathtt{Ask}] \\
&\leq \Pr[\mathtt{S}_1 | \mathtt{Ask}] + \Pr[\mathtt{S}_1 | \neg\mathtt{Ask}] \\
&\leq \Pr[\mathtt{S}_1 | \mathtt{Ask}] + \frac{1}{2}. \qquad (18)
\end{aligned}$$

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 $\Pr[\mathtt{F}_2] \geq 1/(q_{PK} + q_{SK} + q_D)$. In addition we have $\Pr[\mathtt{S}_2 | \mathtt{Ask} \wedge \mathtt{F}_2] = \Pr[\mathtt{S}_1 | \mathtt{Ask}]$. Which gives us

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

We claim that $\Pr[\mathtt{S}_2 | \mathtt{Ask}] = (q_H + q_D)\mathrm{Adv}_{\mathtt{PK}}^{\mathtt{OW-CPA}^{++}}(B_2)$ for an algorithm B_2.

Algorithm B_2 takes as input a public key $\mathfrak{pk}^*$, it has access to a challenge oracle $\mathcal{O}_{\mathbb{E}_{\mathtt{PK}}}()$, 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. $(\mathtt{ID}^*, s) \leftarrow A_1(M_{\mathfrak{pk}}, M_{\mathfrak{sk}})$.
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. $b'' \leftarrow A_2(c^*, k^*, s, \mathtt{ID}^*)$.
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 on the 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 on the 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, place $(k, c_1, \mathfrak{pk}, \perp, m_2)$ onto the 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 on the L_H then we we output k.
  - If $(k, c_1, \mathfrak{pk}, \perp, m_2)$ is on the 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 $(k, c_1, \mathfrak{pk}, m_1, m_2)$ onto the 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 (17) and (18) we have

$$\begin{aligned}\mathrm{Adv}_{\mathtt{PK}}^{\mathtt{OW-CPA}}(B_2) &= \frac{\Pr[\mathtt{S}_2|\mathtt{Ask}]}{q_H + q_D} \\ &\geq \frac{\Pr[\mathtt{S}_1|\mathtt{Ask}]}{(q_H + q_D)(q_{PK} + q_{SK} + q_D)} \\ &\geq \frac{\Pr[\mathtt{S}_0] - \frac{1}{2}}{(q_H + q_D)(q_{PK} + q_{SK} + q_D)} \\ &= \frac{\mathrm{Adv}_{\mathtt{CL-KEM}}^{\mathtt{Type-II}}(A)}{2(q_H + q_D)(q_{PK} + q_{SK} + q_D)}.\end{aligned}$$

The result follows.

Acknowledgements

The authors would like to offer their thanks to Alex Dent and Kenny Paterson for providing detailed and insightful comments on an earlier version 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. Certificateless public key cryptography. In *Advances in Cryptology – ASIACRYPT 2003*, Springer-Verlag LNCS 2894, 452–473, 2003.
3. S.S. Al-Riyami and K.G. Paterson. CBE from CL-PKE: A generic construction and efficient schemes. To appear *Public Key Cryptography – PKC 2005.*
4. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, **32**, 586–615, 2003.
5. D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology – ASIACRYPT 2001*, Springer-Verlag LNCS 2248, 514–532, 2001.
6. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, **33**, 167–226, 2003.
7. A. Dent. A designer's guide to KEMs. In *Cryptography and Coding, 2003*, Springer-Verlag LNCS 2898, 133–151, 2003.
8. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology – CRYPTO '99*, Springer-Verlag LNCS 1666, 537–554, 1999.
9. M. Joye, J. Quisquater and M. Yung. On the power of misbehaving adversaries and security analysis of the original EPOC. In *Topics in Cryptography – CT-RSA 2001*, Springer-Verlag LNCS 2020, 208–222, 2001.
10. B. Lynn. Authenticated Identity-Based Encryption. Cryptology ePrint Archive, Report 2002/072, 2002. `http://eprint.iacr.org/`.

11. A. Miyaji, M. Nakabayashi and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. In *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E84-A**, 1234–1243, 2001.
12. 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*, Springer-Verlag LNCS 1992, 104–118, 2001.
13. J. Pollard. Monte Carlo methods for index computation (mod p). *Math. Comp.*, **32**, 918–924, 1978.
14. O. Schirokauer. Using number fields to compute logarithms in finite fields. *Math. Comp.*, **69**, 1267–1283, 2000.
15. V. Shoup. Using Hash Functions as a Hedge against Chosen Ciphertext Attack. In *Advances in Cryptology - EUROCRYPT 2000*, Springer-Verlag LNCS 1807, 275-288, 2000.