Efficient Identity-Based Key Encapsulation to Multiple Parties

M. Barbosa[1⋆] and P. Farshim[2]

[1] Departamento de Informática, Universidade do Minho,
Campus de Gualtar, 4710-057 Braga, Portugal.
mbb@di.uminho.pt
[2] Department of Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road,
Bristol BS8 1UB, United Kingdom.
farshim@cs.bris.ac.uk

Abstract. We introduce the concept of identity based key encapsulation to multiple parties (mID-KEM), and define a security model for it. This concept is the identity based analogue of public key KEM to multiple parties. We also analyse possible mID-KEM constructions, and propose an efficient scheme based on bilinear pairings. We prove our scheme secure in the random oracle model under the Gap Bilinear Diffie-Hellman assumption.
Keywords. Key Encapsulation Mechanism (KEM), Hybrid Encryption, Identity Based Cryptography, Provable Security.

1 Introduction

A traditional public key encryption scenario involves communication between two users. One party, Alice, wishes to communicate securely with another party, Bob. Alice uses Bob's public key, which she (somehow) knows to be authentic, to create a message that only Bob can decrypt using his private key.

Due to the large computational cost associated with public key encryption algorithms, most practical applications use *hybrid encryption* to handle large plaintext messages. Rather than using the public key encryption algorithm directly over the plaintext, one generates a random *session key* and uses it to encrypt the message under a more efficient symmetric algorithm. The (small) session key is then encrypted using the public key encryption algorithm, and the message is transferred as the combination of both ciphertexts.

The hybrid public key encryption paradigm has gained strength in recent years with the definition and formal security analysis of the generic KEM-DEM construction [6, 7, 16]. In this approach to hybrid encryption one defines a symmetric *data encapsulation mechanism* (DEM) which takes a key k and a message M and computes $C \leftarrow \mathtt{DEM}_k(M)$. Given knowledge of k one can also recover M via $M \leftarrow \mathtt{DEM}_k^{-1}(C)$.

⋆ Funded by scholarship SFRH/BPD/20528/2004, awarded by the Fundação para a Ciência e Tecnologia, Ministério da Ciência e do Ensino Superior, Portugal.

The key k is transferred to the recipient of the ciphertext in the form of an encapsulation. To create this encapsulation, the sender uses a *key encapsulation mechanism* (KEM). This is an algorithm which takes as input a public key $\mathtt{pk}$ and outputs a session key k plus an encapsulation E of this session key under the public key. We write this as

$$(k, E) \leftarrow \mathtt{KEM}(\mathtt{pk}).$$

Notice that the session key is not used as input to the KEM. The recipient recovers the key k using his private key $\mathtt{sk}$ via the decapsulation mechanism. We denote this by

$$k \leftarrow \mathtt{KEM}^{-1}(E, \mathtt{sk}).$$

The full ciphertext of the message M is then given by $E||C$.

The use of the KEM-DEM philosophy allows the different components of a hybrid encryption scheme to be designed in isolation, leading to a simpler analysis and hopefully more efficient schemes. In fact, Cramer and Shoup [6] established that an hybrid encryption scheme is IND-CCA2 secure, if both the KEM and DEM modules are *semantically secure* [8] against *adaptive chosen ciphertext attacks* [13] (see Theorems 4 and 5 in [6]).

Smart [15] extends this notion to a setting in which Alice wants to send the same message to multiple parties. He considers the question: is it possible, in this situation, to avoid carrying out n instances of the KEM-DEM construction? Smart [15] proposes a multiple KEM (or mKEM) primitive whereby the sender can create an encapsulation of the session key k under n public keys. The purpose of this type of construction is to save $n - 1$ data encapsulations overall: a single DEM invocation using k suffices to obtain valid ciphertexts for all recipients. The entire message encryption process becomes

$$\begin{aligned}(k, E) &\leftarrow \mathtt{mKEM}(\mathtt{pk}_1, \ldots, \mathtt{pk}_n)\\ C &\leftarrow \mathtt{DEM}_k(M)\end{aligned}$$

and each recipient i will get the pair (E, C). For decryption, user i simply performs the following two operations:

$$\begin{aligned}k &\leftarrow \mathtt{mKEM}^{-1}(E, \mathtt{sk}_i)\\ M &\leftarrow \mathtt{DEM}_K^{-1}(C).\end{aligned}$$

Identity-based encryption is a form of public key encryption in which users' public keys can take arbitrary values such as e-mail addresses. A central trusted authority manages the public keys in the system, providing legitimate users with private keys that allow recovering messages encrypted under their identities.

Boneh and Franklin [5] introduced a secure and efficient identity-based encryption scheme based on pairings on elliptic curves. Lynn [10] mentions that the encryption algorithm proposed by Boneh and Franklin is unlikely to be used, since in practice one will use a form of identity based key encapsulation mechanism. Bentahar et al. [4] call such an encapsulation mechanism ID-KEM.

Lynn [10] describes a possible ID-KEM construction, but he gives no security model or proof. Bentahar et al. [4] formalise the notion of key encapsulation for the identity-based setting, define a security model and propose several provably secure ID-KEM constructions.

In this work we bring together the concepts of ID-KEM and mKEM and propose mID-KEM: identity-based key encapsulation to multiple users.

This paper is organised as follows. In Sections 2 and 3 we briefly discuss background on pairings, ID-KEMs and mKEMs. The concept and security of an mID-KEM primitive is defined in Section 4. In Section 5 we present an efficient mID-KEM construction, and prove it is secure in Section 6. We conclude the paper by analysing the efficiency of our scheme in Section 7.

2 Background on Pairings

Our constructions will use bilinear maps and bilinear groups. We briefly review the necessary facts about these here. Further details may be found in [5, **?**].

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 $\hat{t} : G_1 \times G_2 \to G_T$.

In many cases one can set $G_1 = G_2$ as is done in [5]. 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.

The map $\hat{t}$, which is usually derived from the Weil or Tate pairings on an elliptic curve, is required to satisfy the following conditions:

1. Bilinear: $\hat{t}(aQ, bR) = \hat{t}(Q, R)^{ab}$, for all $Q \in G_1$, $R \in G_2$ and $a, b \in \mathbb{Z}_q$.
2. Non-degenerate: $\hat{t}(P_1, P_2) \neq 1$.
3. Efficiently computable.

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 and the isomorphism ρ 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. There are several hard problems associated with a group description Γ which can be used for building cryptosystems. These have their origins in the work of Boneh and Franklin [5].

Notation. 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 A is a probabilistic polynomial time (PPT) 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. We also extend the set membership notation to tuples, writing $x \in (x_1, \ldots, x_n)$ to mean $x \in \{x_1, \ldots, x_n\}$.

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

1. $a, b, c \leftarrow \mathbb{Z}_q^*$
2. $t \leftarrow A(\Gamma, aP_1, bP_1, cP_2)$

The advantage of the adversary is defined to be

$$\mathrm{Adv}_{\mathtt{BDH}}(A) = \Pr[t = \hat{t}(P_1, P_2)^{abc}]. \tag{1}$$

Notice there are various equivalent formulations of this: given xP_2 one can compute xP_1 via the isomorphism ρ. This particular formulation allows for a clearer exposition of the security proof in Section 6.

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

$$\mathcal{D}_\Gamma = \{(aP_1, bP_1, cP_2, \hat{t}(P_1, P_2)^{abc}) : \ a, b, c \in [1, \ldots, q]\},$$
$$\mathcal{R}_\Gamma = G_1 \times G_1 \times G_2 \times G_T.$$

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

1. $a, b, c \leftarrow \mathbb{Z}_q^*$
2. $d \leftarrow \{0, 1\}$
3. If $d = 0$ then $\alpha \leftarrow G_T$
4. Else $t \leftarrow \hat{t}(P_1, P_2)^{abc}$
5. $d' \leftarrow A(\Gamma, aP_1, bP_1, cP_2, t)$

We define the advantage of such an adversary by

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

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

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

3 Review of ID-KEMs and mKEMs

An identity based KEM (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 $M_{\mathtt{pk}}$ and the master secret key $M_{\mathtt{sk}}$. We assume that $M_{\mathtt{pk}}$ contains all system parameters, including the security parameter t.

– $\mathbb{X}_{\mathtt{ID-KEM}}(\mathtt{ID}, M_{\mathtt{sk}})$. A deterministic algorithm which takes as input $M_{\mathtt{sk}}$ and an identifier string $\mathtt{ID} \in \{0,1\}^*$, and returns the associated private key $S_{\mathtt{ID}}$.
– $\mathbb{E}_{\mathtt{ID-KEM}}(\mathtt{ID}, M_{\mathtt{pk}})$. This is the PPT encapsulation algorithm. On input of $\mathtt{ID}$ and $M_{\mathtt{pk}}$ this outputs a pair (k, C) where $k \in \mathbb{K}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}})$ is a key in the space of possible session keys at a given security level, and $C \in \mathbb{C}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}})$ is the encapsulation of that key.
– $\mathbb{D}_{\mathtt{ID-KEM}}(S_{\mathtt{ID}}, C)$. This is the deterministic decapsulation algorithm. On input of C and $S_{\mathtt{ID}}$ this outputs k or a failure symbol $\perp$.

The soundness of an ID-KEM scheme is defined as:

$$\Pr\left(k = \mathbb{D}_{\mathtt{ID-KEM}}(S_{\mathtt{ID}}, C) \middle| \begin{array}{l}(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow \mathbb{G}_{\mathtt{mID-KEM}}(1^t) \\ \mathtt{ID} \leftarrow \{0,1\}^* \\ (k, C) \leftarrow \mathbb{E}_{\mathtt{ID-KEM}}(\mathtt{ID}, M_{\mathtt{pk}}) \\ S_{\mathtt{ID}} \leftarrow \mathbb{X}_{\mathtt{ID-KEM}}(\mathtt{ID}, M_{\mathtt{sk}})\end{array}\right) = 1$$

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

IND-CCA2 Adversarial Game
1. $(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID-KEM}}(1^t)$
2. $(s, \mathtt{ID}^*) \leftarrow A_1^{\mathcal{O}}(M_{\mathtt{pk}})$
3. $(k_0, C^*) \leftarrow \mathbb{E}_{\mathtt{ID-KEM}}(\mathtt{ID}^*, M_{\mathtt{pk}})$
4. $k_1 \leftarrow \mathbb{K}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}})$
5. $b \leftarrow \{0,1\}$
6. $b' \leftarrow A_2^{\mathcal{O}}(M_{\mathtt{pk}}, C^*, s, \mathtt{ID}^*, k_b)$

In the above s is some state information and $\mathcal{O}$ denotes oracles to which the adversary has access. In this model the adversary has access to two oracles: a private key extraction oracle which, on input of $\mathtt{ID} \neq \mathtt{ID}^*$, will output the corresponding value of $S_{\mathtt{ID}}$; and 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 it with the pair $(C^*, \mathtt{ID}^*)$.

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

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

An ID-KEM is considered to be IND-CCA2 secure if for all PPT adversaries A, the advantage in this game is a negligible function of the security parameter t.

The notion of KEM is extended in [15] to multiple parties. An mKEM is defined by the following three polynomial time algorithms:

– $\mathbb{G}_{\mathtt{mKEM}}(1^t)$ which is a probabilistic key generation algorithm. On input of 1^t and the domain parameters, this algorithm outputs a public/private key pair $(\mathtt{pk}, \mathtt{sk})$.

– $\mathbb{E}_{\mathtt{mKEM}}(\mathcal{P})$ which is a probabilistic encapsulation algorithm. On input of a set of public keys $\mathcal{P} = \{\mathtt{pk}_1, \ldots, \mathtt{pk}_n\}$ this algorithm outputs an encapsulated key-pair (k, C), where $k \in \mathbb{K}_{\mathtt{mKEM}}(t)$ is the session key and C is an encapsulation of the key k under the public keys $\{\mathtt{pk}_1, \ldots, \mathtt{pk}_n\}$.
– $\mathbb{D}_{\mathtt{mKEM}}(\mathtt{sk}, C)$ which is a decapsulation algorithm. This takes as input an encapsulation C and a private key $\mathtt{sk}$, and outputs a key k or a special symbol $\perp$ representing the case where C is an invalid encapsulation with respect to the private key $\mathtt{sk}$.

The soundness of an mKEM scheme is defined as:

$$\Pr\left(k = \mathbb{D}_{\mathtt{mKEM}}(\mathtt{sk}_j, C) \middle| \begin{array}{l} (\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathbb{G}_{\mathtt{mKEM}}(1^t), 1 \leq i \leq n \\ (k, C) \leftarrow \mathbb{E}_{\mathtt{mKEM}}(\mathtt{pk}_1, \ldots, \mathtt{pk}_n) \\ j \leftarrow \{1, \ldots, n\} \end{array} \right) = 1$$

Security of an mKEM is defined in a similar manner to a KEM via the following game.

(m, n)-IND-CCA2 Adversarial Game
1. $(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathbb{G}_{\mathtt{mKEM}}(1^t)$ for $1 \leq i \leq n$
2. $\mathcal{P} \leftarrow \{\{\mathtt{pk}_1, \ldots, \mathtt{pk}_n\}\}$
3. $(s, \mathcal{P}^*) \leftarrow A_1^{\mathcal{O}}(\mathcal{P})$ where $\mathcal{P}^* \subseteq \mathcal{P}$ and $m = |\mathcal{P}^*| \leq n$
4. $(k_0, C^*) \leftarrow \mathbb{E}_{\mathtt{mKEM}}(\mathcal{P}^*)$
5. $k_1 \leftarrow \mathbb{K}_{\mathtt{mKEM}}(t)$
6. $b \leftarrow \{0, 1\}$
7. $b' \leftarrow A_2^{\mathcal{O}}(\mathtt{pk}, C^*, s, \mathcal{P}^*, k_b)$

In the above s is some state information and $\mathcal{O}$ denotes the decapsulation oracle to which the adversary has access in rounds 1 and 2. As it is noted in [15], restricting access to this oracle by simply disallowing the query $(C^*, \mathtt{pk}^*)$ for some $\mathtt{pk}^* \in \mathcal{P}^*$ would be too lenient. Consider an mKEM in which an encapsulation is of the form $C = c_1 || \ldots || c_n$ where each c_i is intended for the recipient with public key $\mathtt{pk}_i$. On reception of a challenge encapsulation of this form, the adversary could query the decapsulation oracle with a pair $(c_i^*, \mathtt{pk}_i^*)$, which is *not* disallowed, and win the game.

To avoid this type of problem, Smart [15] defines more restrictive, but still reasonable, oracle access. The restriction imposed in the model is that in the second stage the adversary is only allowed to query the decapsulation oracle with those C which decapsulate to a different key from that encapsulated by C^*. We will come back to this in the next section.

The advantage of the adversary A is defined as

$$\mathrm{Adv}_{\mathtt{mKEM}}^{(m,n)-\mathtt{IND-CCA2}}(A) = |2 \Pr[d = d'] - 1|.$$

An mKEM is considered to be (m, n)-IND-CCA2 secure if for all PPT adversaries A, the above advantage is a negligible function of the security parameter t.

4 ID-KEM to Multiple Parties

In this section we propose a new cryptographic primitive: identity based key encapsulation to multiple parties (mID-KEM). This primitive is a direct adaptation of the mKEM primitive in [15] to the identity-based setting.

An mID-KEM scheme is a four-tuple of polynomial time algorithms:

- $\mathbb{G}_{\mathtt{mID-KEM}}(1^t)$. A PPT algorithm which takes as input a security parameter 1^t and returns the master public key $M_{\mathtt{pk}}$ and the master secret key $M_{\mathtt{sk}}$. We assume that $M_{\mathtt{pk}}$ contains all system parameters, including the security parameter t.
- $\mathbb{X}_{\mathtt{mID-KEM}}(\mathtt{ID}, M_{\mathtt{sk}})$. A deterministic algorithm which takes as input $M_{\mathtt{sk}}$ and an identifier string $\mathtt{ID} \in \{0,1\}^*$, and returns the associated private key $S_{\mathtt{ID}}$.
- $\mathbb{E}_{\mathtt{mID-KEM}}(\mathcal{I}, M_{\mathtt{pk}})$. This is the PPT encapsulation algorithm. On input of a tuple $\mathcal{I} = (\mathtt{ID}_1, \ldots, \mathtt{ID}_n)$ of n identities and $M_{\mathtt{pk}}$, this outputs a pair (k, C), where $k \in \mathbb{K}_{\mathtt{mID-KEM}}(M_{\mathtt{pk}})$ is a session key and $C = (c_1, \ldots, c_n)$, with each $c_i \in \mathbb{C}_{\mathtt{mID-KEM}}(M_{\mathtt{pk}})$ an encapsulation of k for $\mathtt{ID}_i \in \mathcal{I}$.
- $\mathbb{D}_{\mathtt{mID-KEM}}(S_{\mathtt{ID}_i}, c_i)$. This is the deterministic decapsulation algorithm. On input of the private key $S_{\mathtt{ID}_i}$ for identity $\mathtt{ID}_i$ and c_i, this outputs k or a failure symbol $\perp$.

The soundness of an mID-KEM scheme is defined as:

$$\Pr\left(k = \mathbb{D}_{\mathtt{mID-KEM}}(S_{\mathtt{ID}_j}, c_j, M_{\mathtt{pk}}) \;\middle|\; \begin{array}{l} (M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow \mathbb{G}_{\mathtt{mID-KEM}}(1^t) \\ \mathtt{ID}_i \leftarrow \{0,1\}^*, 1 \le i \le n \\ (k, c_1, \ldots, c_n) \leftarrow \mathbb{E}_{\mathtt{mID-KEM}}(\mathtt{ID}_1, \ldots, \mathtt{ID}_n, M_{\mathtt{pk}}) \\ S_{\mathtt{ID}_i} \leftarrow \mathbb{X}_{\mathtt{mID-KEM}}(\mathtt{ID}_i, M_{\mathtt{sk}}), 1 \le i \le n \\ j \leftarrow \{1, \ldots, n\} \end{array}\right) = 1$$

There is a subtle difference between this and the mKEM definition presented in the previous section. Recall that Smart [15] views encapsulation for multiple users as an algorithm taking a set $\mathcal{P}$ of public keys and producing a *single* token C. Decapsulation must then recover the encapsulated key whenever C is provided together with the secret key for one of the users in the set. This is the most natural extension of KEM to the multi-user setting.

In our approach, however, C is seen as an n-tuple, where the i-th component is an encapsulation specific to $\mathtt{ID}_i$, and the decapsulation algorithm does not accept the whole of C as input, but only c_i. We take this approach because it simplifies the security model (note the definition and access restrictions of the decapsulation oracle below) and allows for a clearer explanation of the security proof in Section 6.

Conceptually, these two approaches are equivalent. Any C in the mKEM model can be recast as an n-tuple by setting $c_i = C$ for all i. Conversely an n-tuple in the mID-KEM model is just a special case of the first approach that provides more information, as it implies a semantic interpretation of the encapsulation.

We define the semantic security of an mID-KEM scheme according to the following two-stage game between an adversary A and a challenger.

n-IND-CCA2 Adversarial Game

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

In the above, $\mathcal{I}^* = (\mathtt{ID}_1^*, \ldots, \mathtt{ID}_n^*)$ denotes the identities chosen by the adversary to be challenged on, $C^* = (c_1^*, \ldots, c_n^*)$ denotes the challenge encapsulations, s is some state information and $\mathcal{O}$ denotes oracles to which the adversary has access.

In this model the adversary has access to a private key extraction oracle which, on input of $\mathtt{ID} \notin \mathcal{I}^*$, will output the corresponding value of $S_{\mathtt{ID}}$. The adversary can also query a decapsulation oracle with respect to any identity $\mathtt{ID}$ of its choice. It has access to this decapsulation oracle, subject to the restriction that in the second phase A is not allowed to call $\mathcal{O}$ with any pair $(c_i^*, \mathtt{ID}_i^*)$. [1]

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

$$\mathrm{Adv}_{\mathtt{mID-KEM}}^{n-\mathtt{IND-CCA2}}(A) = |2\Pr[b' = b] - 1|.$$

An mID-KEM is considered to be n-IND-CCA2 secure, if for all PPT adversaries A, the advantage in the game above is a negligible function of the security parameter t.

5 mID-KEM Schemes

In this section we present two mID-KEM schemes, one of which is a simple construction that uses identity based encryption. The second scheme is non-trivial in the sense that it provides a significant improvement in efficiency.

5.1 A Simple mID-KEM Scheme

One might think that an obvious way to construct an mID-KEM scheme would be to use n instances of an ID-KEM scheme. However this is not valid since it would not guarantee that the same key k is encapsulated to all users. In fact, this would imply using n parallel instances of the associated DEM, thereby subverting the principle of efficient hybrid encryption to multiple users.

An mID-KEM scheme can be easily built using a generic construction similar to that used in [15], whereby an IND-CCA2 secure public key encryption algorithm is used to build a secure mKEM.

In the identity based setting, one would take a secure identity based encryption algorithm, for instance Boneh and Franklin's Full-Ident scheme [5], as the starting point. The resulting mID-KEM construction would operate as follows.

[1] Note that this accounts for cases where the adversary includes repetitions in $\mathcal{I}^*$.

$\mathbb{E}_{\mathtt{mID-KEM}}(\mathtt{ID}_1, \ldots, \mathtt{ID}_n, M_{\mathtt{pk}})$
- $m \leftarrow \mathbb{M}_{\mathtt{IBE}}(M_{\mathtt{pk}})$
- $k \leftarrow \mathtt{KDF}(m)$
- $c_i \leftarrow \mathbb{E}_{\mathtt{IBE}}(m, \mathtt{ID}_i, M_{\mathtt{pk}})$, for $1 \leq i \leq n$
- Return $(k, c_1, \ldots, c_n)$

$\mathbb{D}_{\mathtt{mID-KEM}}(S_{\mathtt{ID}_i}, c_i, M_{\mathtt{pk}})$
- $m \leftarrow \mathbb{D}_{\mathtt{IBE}}(S_{\mathtt{ID}_i}, c_i, M_{\mathtt{pk}})$
- $k \leftarrow \mathtt{KDF}(m)$
- Return k

where KDF is a key derivation function that maps the message space of the identity based encryption algorithm to the key space of the DEM.

The security proof in [15] for the generic mKEM construction builds on the work by Bellare et al. [3], who established that a public key encryption algorithm which is secure in the single user setting, is also secure when multiple users are involved. Given that the authors have no knowledge of such a result for identity based encryption algorithms, proving the security of the previous construction will require extending Bellare et al.'s results. Since the main result of this paper is a more efficient scheme proposed in the next section, we do not further debate the security of this simpler construction.

5.2 An Efficient mID-KEM from Bilinear Pairings

The scheme we propose in this section is inspired by Smart's OR constructions in [14]. Our construction uses a parameter generator $\mathbb{G}_{\mathtt{mID-KEM}}(1^t)$ similar to that of most identity based schemes. On input of a security parameter 1^t, it outputs the master public key $M_{\mathtt{pk}}$ and the master secret key $M_{\mathtt{sk}}$, where $M_{\mathtt{sk}} \leftarrow \mathbb{Z}_q^*$ and $M_{\mathtt{pk}} \leftarrow M_{\mathtt{sk}} \cdot P_1$.

Private key extraction is also performed in the usual way. On input of an identity ID and the master secret key $M_{\mathtt{sk}}$, the extraction algorithm $\mathbb{X}_{\mathtt{mID-KEM}}$ returns $S_{\mathtt{ID}} \leftarrow M_{\mathtt{sk}} \cdot H_1(\mathtt{ID})$, where $H_1 : \{0,1\}^* \rightarrow G_2$ is a cryptographic hash function.

The key encapsulation and decapsulation algorithms work as follows.

$\mathbb{E}_{\mathtt{mID-KEM}}(\mathtt{ID}_1, \ldots, \mathtt{ID}_n, M_{\mathtt{pk}})$
- $r \leftarrow \mathbb{Z}_q^*$
- $s \leftarrow \mathbb{Z}_q^*$
- $U_r \leftarrow rP_1$
- $U_s \leftarrow sP_1$
- $Q_{\mathtt{ID}_i} \leftarrow H_1(\mathtt{ID}_i)$
- For $i = 1$ to n
  - $U_i \leftarrow r(Q_{\mathtt{ID}_i} - sP_2)$
  - $c_i \leftarrow (U_r, U_s, U_i)$
- $T \leftarrow \hat{t}(M_{\mathtt{pk}}, rsP_2)$
- $k \leftarrow H_2(T||U_r||U_s)$
- Return $(k, c_1, \ldots, c_n)$

$\mathbb{D}_{\mathtt{mID-KEM}}(S_{\mathtt{ID}_i}, c_i, M_{\mathtt{pk}})$
- $t \leftarrow \hat{t}(U_r, S_{\mathtt{ID}_i})$
- $t' \leftarrow \hat{t}(M_{\mathtt{pk}}, -U_i)$
- $T \leftarrow t \cdot t'$
- $k \leftarrow H_2(T||U_r||U_s)$
- Return k

In the above, H_2 is a cryptographic hash function that takes elements in $G_T||G_1||G_1$ to the DEM key space.

6 Proof of Security

Theorem 1 describes our result on the security of the mID-KEM scheme in Section 5.2 with respect to the security model defined in Section 4. The technique we use in the proof might allow one to establish the security of a suitable modification of the scheme in [14], for which no proof is provided.

Theorem 1. *The mID-KEM construction described in Section 5.2 is n-IND-CCA2 secure under the hardness of GBDH in the random oracle model. Specifically, for any PPT algorithm A that breaks this mID-KEM scheme, there exists a PPT algorithm B with*

$$\mathrm{Adv}^{n-\mathtt{IND-CCA2}}_{\mathtt{mID-KEM}}(A) \leq 2q_T^n \cdot \mathrm{Adv}_{\mathtt{GBDH}}(B, q_D^2 + 2q_D q_2 + q_2)$$

where q_1, q_2, q_X and q_D denote the maximum number of queries the adversary places to the H_1, H_2, private key extraction and decapsulation oracles respectively, and $q_T = q_1 + q_X + q_D$ is the total number of queries placed to H_1.

Proof. Let $\mathtt{S}$ be the event that A wins the n-IND-CCA2 adversarial game in Section 4. Let also $T^*||U_r^*||U_s^*$ denote the value that must be passed to H_2 to obtain the correct key encapsulated in the challenge ciphertext. Then, we have

$$\Pr[\mathtt{S}] = \Pr[\mathtt{S} \wedge \mathtt{Ask}] + \Pr[\mathtt{S} \wedge \neg\mathtt{Ask}] \leq \Pr[\mathtt{S} \wedge \mathtt{Ask}] + \frac{1}{2} \tag{2}$$

where $\mathtt{Ask}$ denotes the event that A queries H_2 with $T^*||U_r^*||U_s^*$ during the simulation. This follows from the fact that if A does not place this query, then it can have no advantage.

We argue that in the event $\mathtt{S} \wedge \mathtt{Ask}$, there is an adversary B that uses A to solve the GBDH problem with probability q_T^{-n} while making at most $q_D^2 + 2q_D q_2 + q_2$ DBDH oracle queries. We implement algorithm B as follows.

For a given GBDH problem instance $(\Gamma, aP_2, bP_2, cP_2)$ the strategy is to embed the problem into the mID-KEM challenge presented to the adversary. Namely we construct the simulation so that the value of the pairing associated with the challenge encapsulation is $T^* = \hat{t}(P_1, P_2)^{abc}$. Hence, when A places a query with the correct value of T^* to H_2, it provides us with the solution to the GBDH problem instance.

We pass Γ and $M_{\mathtt{pk}} = cP_1 = \rho(cP_2)$ on to A as the global public parameters. We also maintain three lists that allow us to provide consistent answers to A's oracle calls: $L_1 \subset \{0,1\}^* \times G_2 \times \mathbb{F}_q$; $L_2 \subset G_T \times G_1 \times G_1 \times \mathbb{K}_{\mathtt{mID-KEM}}(M_{\mathtt{pk}})$; and $L_D \subset \{0,1\}^* \times G_1 \times G_1 \times \mathbb{K}_{\mathtt{mID-KEM}}(M_{\mathtt{pk}})$.

Initially, we generate a list of n random indices $i_1^*, \ldots, i_n^*$, with $1 \leq i_k^* \leq q_T$. These indices will determine the set of n identities which B is guessing the adversary will include in its challenge query $\mathcal{I}^*$. Namely, these indices determine the way in which the hash function H_1 is simulated.

H_1 queries: $H_1(\mathtt{ID}_i)$. On the i-th query to H_1, we check whether i matches one of the indices i_k^*. If this is the case, we generate random $r_{i_k^*} \in \mathbb{Z}_q^*$, return $Q_{\mathtt{ID}_{i_k^*}} = r_{i_k^*} P_2 + bP_2$ and add $(\mathtt{ID}_{i_k^*}, Q_{\mathtt{ID}_{i_k^*}}, r_{i_k^*})$ to L_1. Otherwise, we generate random $r_i \in \mathbb{Z}_q^*$, return $Q_{\mathtt{ID}_i} = r_i P_2$ and add $(\mathtt{ID}_i, Q_{\mathtt{ID}_i}, r_i)$ to L_1.

Algorithm B requires that at the end of phase one, the adversary outputs a list of identities $\mathcal{I}^* = (\mathtt{ID}_1^*, \ldots, \mathtt{ID}_n^*)$ such that $\mathtt{ID}_k^* = \mathtt{ID}_{i_k^*}$. This happens with probability at least q_T^{-n}. If this is not the case, the simulation fails. Otherwise, the mID-KEM challenge encapsulation is created as

$$\begin{aligned} U_r &\leftarrow \rho(aP_2) \\ U_s &\leftarrow \rho(bP_2) \\ U_k &\leftarrow r_{i_k^*} aP_2 \\ c_k^* &\leftarrow (U_r, U_s, U_k) \end{aligned}$$

for each $\mathtt{ID}_k^* \in \mathcal{I}^*$. Notice that this is a well-formed encapsulation:

$$U_k = a(Q_{\mathtt{ID}_k^*} - bP_2) = a(r_{i_k^*} P_2 + bP_2 - bP_2) = r_{i_k^*} aP_2.$$

When A outputs $\mathcal{I}^*$, algorithm B responds with $C^* = (c_1^*, \ldots, c_n^*)$ constructed as above plus a random challenge key k^*.

Given that A is not allowed to obtain the private keys associated with the identities in the challenge, the extraction oracle is easily simulated as follows.

Extraction queries: $\mathcal{O}_X(\mathtt{ID}_j)$. On input $\mathtt{ID}_j$, we obtain $Q_{\mathtt{ID}_j}$ by calling H_1 first. If $Q_{\mathtt{ID}_j}$ is of the form $r_i P_2$, we look in L_1 for the corresponding r_i and return $S_{\mathtt{ID}_i} = r_i(cP_2)$. If during the first stage of the game, A should query for the extraction of a private key corresponding to a $Q_{\mathtt{ID}}$ of the form $r_{i_k^*} P_2 + bP_2$, the simulation would fail.

Finally, the decapsulation and H_2 oracles are simulated as follows.

Decapsulation queries: $\mathcal{O}_D(\mathtt{ID}_m, U_{r_m}, U_{s_m}, U_m)$. On queries in which we can calculate a value T_m, we do so, and call H_2 with parameters (T_m, U_{r_m}, U_{s_m}) to obtain the return value. This will happen when $\mathtt{ID}_m \notin \mathcal{I}^*$, in which case we query the private key extraction oracle, and compute T_m as in a standard decapsulation:

$$T_m = \hat{t}(U_{r_m}, S_{\mathtt{ID}_m})\hat{t}(\rho(cP_2), -U_m).$$

When $\mathtt{ID}_m \in \mathcal{I}^*$ we first search L_2 for an entry such that $U_{r_m} = U_{r_n}$, $U_{s_m} = U_{s_n}$ and the DBDH oracle returns 1 when queried with $(U_{r_m}, U_{s_m}, cP_2, T_n)$. If this entry exists, we return the corresponding k_n. If not, we search L_D for a tuple matching the parameters $(\mathtt{ID}_m, U_{r_m}, U_{s_m})$. If it exists, we return the associated key k_m. Otherwise, we generate a random k_m, return it and update L_D by adding the entry $(\mathtt{ID}_m, U_{r_m}, U_{s_m}, k_m)$.

H_2 queries: $H_2(T_n, U_{r_n}, U_{s_n})$. We first check if T_n is the solution we are looking for by calling the DBDH oracle with (aP_1, bP_1, cP_2, T_n). If it returns 1, the solution is found, we output T_n and terminate. Otherwise, we proceed to search L_2 for a tuple matching the parameters (T_n, U_{r_n}, U_{s_n}). If it exists, we return the corresponding key k_n. If not, we search L_D for an entry such that $U_{r_m} = U_{r_n}$, $U_{s_m} = U_{s_n}$ and the DBDH oracle returns 1 when queried with $(U_{r_m}, U_{s_m}, cP_2, T_n)$. If this entry exists, we return the corresponding k_m. Otherwise, we generate a random k_n, return it and update L_2 by appending the entry $(T_n, U_{r_n}, U_{s_n}, k_n)$.

To complete the proof, it suffices to notice that B will only fail when the identities $\mathcal{I}^*$ chosen by A at the end of the first round do not match the random indices chosen at the beginning. Since this happens with probability at least q_T^{-n}, we conclude

$$\Pr[\mathtt{S} \wedge \mathtt{Ask}] \leq q_T^n \cdot \mathrm{Adv}_{\mathtt{GBDH}}(B, q_D^2 + 2q_D q_2 + q_2). \tag{3}$$

The total number of DBDH oracle calls ($q_D^2 + 2q_D q_2 + q_2$) follows from the way we simulate the decapsulation and H_2 oracles. Theorem 1 now follows from (2) and (3).

We note that the tightness of the security reduction in Theorem 1 decreases exponentially with the number of recipients. For practical purposes, and in line with the discussions on practice-oriented provable security in [2, 9], it is important to emphasise that our result provides theoretical security guarantees only in scenarios where messages are encrypted to a small number of recipients. A slightly better result, although still exponential in n, can be obtained by optimising the simulation in the proof above. We leave it as an open problem to find a sub-exponential security reduction for this type of scheme.

7 Efficiency Considerations

Execution time and bandwidth usage are two important factors affecting the efficiency of a KEM. In this section we present a comparison, with respect to these factors, between the simple scheme described in Section 5.1 and the scheme proposed in Section 5.2.

The former scheme requires n identity based encryption computations for encapsulation, and one identity based decryption computation for decapsulation. Assuming that we use the Full-Ident scheme of Boneh and Franklin [5], this means n pairing computations for encapsulation and another for decapsulation. On the other hand, as it can be seen from the description of the latter scheme, it takes only one pairing computation to encapsulate and two more to decapsulate. Note that one could essentially eliminate the pairing computation needed in encapsulation by pre-computing the value $\hat{t}(M_{\mathtt{pk}}, P_2)$.

The bandwidth usage of the scheme in Section 5.2 is $2\ell_1 + \ell_2$ bits for each recipient, where ℓ_1 and ℓ_2 are the bit lengths of the representations of elements of G_1 and G_2. Using the Full-Ident encryption scheme in [5], the simpler scheme uses $\ell_1 + 2\ell_k$ bits per user where ℓ_k is the length of the session key. For example, using supersingular elliptic curves over characteristic three we could take $\ell_k = 128$ and $\ell_1 = \ell_2 \approx 190$ bits. This means a bandwidth usage of 570 bits for the second scheme versus 446 bits for the first one.

To conclude this discussion we note the n-fold gain in computational weight obtained in Section 5.2 is achieved at the cost of a small increase in bandwidth usage, and also by reducing the security of the scheme to the hardness of GBDH, which is a stronger assumption than the hardness of BDH used in [5].

Acknowledgements

The authors would like to thank Nigel Smart and John Malone-Lee for their input during this work, and the anonymous reviewers whose comments helped to improve it.

References

1. M. Abdalla, M. Bellare and P. Rogaway. DHAES : An encryption scheme based on the Diffie-Hellman problem. *Cryptology ePrint Archive*, Report 1999/007. 1999.
2. M. Bellare. Practice-priented provable-security. *Proceedings of First International Workshop on Information Security*, LNCS 1396. Springer-Verlag, 1998.
3. M. Bellare, A. Boldyreva and S. Micali. Public-key encryption in the multi-user setting: security proofs and improvements. *Advances in Cryptology – EUROCRYPT 2000*, LNCS 1807:259-274. Springer-Verlag, 2000.
4. K. Bentahar, P. Farshim, J. Malone-Lee and N. P. Smart. Generic constructions of identity-based and certificateless KEMs. *Cryptology ePrint Archive*, Report 2005/058. 2005.
5. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32:586–615. 2003.
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. W. Dent. A designer's guide to KEMs. *Coding and Cryptography*, LNCS 2898:133-151. Springer-Verlag, 2003.
8. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and Systems Sciences*, 28:270–299. 1984.
9. N. Koblitz and A. Menezes. Another look at provable security. *Cryptology ePrint Archive*, Report 2004/152. 2004.
10. B. Lynn. Authenticated identity-based encryption. *Cryptology ePrint Archive*, Report 2002/072. 2002.
11. A. Miyaji, M. Nakabayashi and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *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. *Public Key Cryptography – PKC 2001*, LNCS 1992:104–118. Springer-Verlag, 2001.
13. C. Rackoff and D. R. Simon Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. *Advances in Cryptology – CRYPTO 1991*, LNCS 576:433-444. Springer-Verlag, 1991
14. N. P. Smart. Access control using pairing based cryptography. *Topics in Cryptology – CT-RSA 2003*, LNCS 2612:111–121. Springer-Verlag, 2003.
15. N. P. Smart. Efficient key encapsulation to multiple parties. *Security in Communication Networks*, LNCS 3352:208–219. Springer-Verlag, 2005.
16. V. Shoup. A proposal for an ISO standard for public key encryption (version 2.1). *Preprint*. 2001.