Efficient Key Encapsulation to Multiple Parties

N.P. Smart

Dept. Computer Science, University of Bristol,
Merchant Venturer's Building, Woodland Road,
Bristol, BS8 1UB, United Kingdom
nigel@cs.bris.ac.uk

Abstract. We present the notion of an mKEM, which is a Key Encapsulation Mechanism (KEM) which takes multiple public keys as input. This has applications where one wishes to encrypt a single large document to a set of multiple recipients, as when one sends an encrypted email to more than one person. We present a security definition and show that the naive approach to implementing an mKEM is secure under this definition. We then go on to present a more efficient construction of an mKEM, which is secure in the random oracle model.

1 Introduction

Public key cryptography has been traditionally concerned with two parties communicating. In the traditional scenario one party, Alice, wishes to communicate securely with one other party, Bob. Alice obtains Bob's authentic public key and then encrypts the data she wishes to send to Bob. Bob, knowing the associated private key, is able to decrypt the ciphertext to obtain Alice's message. Since public key algorithms are very slow, if Alice wishes to send a large amount of data she first encrypts a per message symmetric "session key" to Bob using Bob's public key algorithm and then encrypts the actual data using a fast symmetric cipher keyed by the session key. Such a combination of public key and symmetric techniques is called a hybrid encryption algorithm.

This hybrid technique has been strengthened in recent years with the use of the KEM-DEM philosophy, see [4] and [5]. 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

$$C \leftarrow DEM_K(M).$$

Given knowledge of K one can also recover M via

$$M \leftarrow DEM_K^{-1}(C).$$

To transport the key K to the recipient of the ciphertext 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,

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

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

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

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.

However, as soon as one moves away from the traditional two-party setting problems occur. Suppose Alice now wishes to send a large file to two parties (say Bob and Charlie), for example she may wish to encrypt an email to Bob and Charlie, or encrypt a file on her system such that either Bob or Charlie can decrypt it. ¿From one's own experience one notices that very few emails are sent to a single recipient, hence such a one-to-many model is clearly of importance.

A number of possible solutions exist to this problem. All of which have disadvantages. In the first naive solution one simply encrypts the data twice, once for Bob and once for Charlie, using their respective public key schemes. This is clearly wasteful especially if the data to be encrypted is large. A more efficient solution would be to encrypt the data once with a symmetric encryption key K and then encrypt this key under Bob and Charlie's public keys, i.e. the ciphertext would look like

$$\mathcal{E}_{\mathrm{pk}_B}(K) \| \mathcal{E}_{\mathrm{pk}_C}(K) \| \mathcal{E}_K(M).$$

Whilst this is probably sufficient for two users, this can become very expensive for a large number of users.

In addition it is unclear what security definition one is using for such a scheme. The work of Bellare, Boldyreva and Micali [2] looks at the security of encryption schemes in the presence of many users but did not consider the fact that a "ciphertext" could correspond to different users. In their definition the above hybrid encryption to two parties would be considered as two encryptions, whereas we wish to treat it as a single encryption.

The use of the KEM-DEM philosophy in such a situation is also not applicable. After all the KEM produces the session key, hence application of a KEM for two users would result in two different session keys. What is required is a KEM like construction which takes as input n public keys and outputs a session key and an encapsulation of that session key under each of the input public keys. We would like such a multiple KEM (or mKEM) which is more efficient than the above concatenation of public key encryptions of a session key.

In this paper we propose such mKEMs and propose a security definition for them. We show that the above naive concatenation method is secure, in the standard model, under this definition, although inefficient. We then present a public key mKEM based on the Diffie–Hellman problem which is more efficient than repeated encryption of a session key using an analogous traditional public key system, yet is still secure in the random oracle model under the proposed security definition.

2 Notation

We let $v \leftarrow u$ for variables v and u to denote assignment. For a set S we let $v \leftarrow S$ denote the variable v being assigned the set S and $v \overset{R}{\leftarrow} S$ to denote v being assigned an element of the set S chosen uniformly at random.

If A is a, possibly probabilistic, algorithm then $v \leftarrow A$ denotes v being assigned the output of algorithm A with the probability distribution induced by A's input and internal random choices. If we wish to make explicit precisely what value r is used as the randomness in a probabilistic algorithm $A(x)$ with input x we write $A(x; r)$.

A function f is said to be negligible if for all polynomials p there exists a constant N_p such that $f(x) \leq \frac{1}{p(x)}$ for all $x \geq N_p$.

3 Security of a KEM

A KEM (key encapsulation mechanism) is a public key scheme which allows a sender to generate an encryption of a random session key, and allows the holder of the correct private key to recover the session key from the ciphertext. We let $\mathbb{D}$ denote a set of domain parameters which could consist of only the security parameter k written in unary 1^k, or could consist of a public group and generator as in ElGamal systems.

More formally we define a KEM [4] is a triple of algorithms:

- $\mathcal{G}_{KEM}(\mathbb{D})$ which is a probabilistic key generation algorithm. On input of $\mathbb{D}$ this algorithm outputs a public/private key pair $(\mathtt{pk}, \mathtt{sk})$.
- $\mathcal{E}_{KEM}(\mathtt{pk})$ which is a probabilistic encapsulation algorithm. On input of a public key $\mathtt{pk}$ this algorithm outputs an encapsulated key-pair (K, C), where $K \in \mathbb{K}$ is the session key and C is an encapsulation of the key K under the public key $\mathtt{pk}$. In other words C is a ciphertext of the message K. We assume that the space $\mathbb{K}$ of all keys output by $\mathcal{E}$ are of some fixed length.
- $\mathcal{D}_{KEM}(C, \mathtt{sk})$ 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 $\mathtt{sk}$.

For such a scheme to be useful we require that it is complete in the following sense,

$$\Pr\left((\mathtt{pk}, \mathtt{sk}) \leftarrow \mathcal{G}_{KEM}(\mathbb{D}), (K, C) \leftarrow \mathcal{E}_{KEM}(\mathtt{pk}) : K = \mathcal{D}_{KEM}(C, \mathtt{sk})\right) = 1.$$

Security of a KEM is defined in the following way. We assume an adversary $\mathcal{A}$ which runs in two stages. In the first stage it is allowed to produce encapsulations and (depending on the precise security definition we require) it may be allowed access to a decapsulation oracle on encapsulations of its choosing. At the end of this stage it returns some state information.

In the second stage it is provided with a challenge encapsulation C^*, its state information from the first stage plus two keys K_0 and K_1. The adversaries goal in the second stage is to decide which key K_b is encapsulated by C. In this second stage it may also have access to an decapsulation oracle, but if it does it is not allowed to request the decapsulation of the challenge C^*.

Consider the following game played with such an adversary:

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathcal{G}_{KEM}(\mathbb{D})$.
$s \leftarrow \mathcal{A}^1(\mathtt{pk})$.
$b \overset{R}{\leftarrow} \{0, 1\}$.
$(K_b, C^*) \leftarrow \mathcal{E}_{KEM}(\mathtt{pk})$.
$K_{1-b} \overset{R}{\leftarrow} \mathbb{K}$.
$b' \leftarrow \mathcal{A}^2(C^*, \{K_0, K_1\}, s)$.
Output whether $b = b'$.

The adversary is said to win the game if $b = b'$. The advantage of an adversary is defined to be

$$\mathrm{Adv}_{\mathcal{A}} = |\Pr(b = b') - 1/2|.$$

If the maximum advantage over all possible adversaries $\mathcal{A}$ is a negligible function of the security parameter k then we say that the KEM is IND-xxx secure, where xxx denotes what access $\mathcal{A}$ is allowed to a decapsulation oracle. If $\mathcal{A}$ is not allowed any access to such an oracle then we say the scheme is IND-CPA secure, if it is only allowed access during stage one then we say the scheme is IND-CCA1 secure and if it is allowed access in both stages (subject to the earlier restriction on requesting the decapsulation of C^*) then we say the scheme is IND-CCA2 secure.

A KEM needs to be used with a DEM (data encapsulation mechanism) to provide a hybrid encryption algorithm. A DEM is a symmetric encryption algorithm which takes a symmetric key k and a message (resp. ciphertext) and provides the corresponding ciphertext (resp. message). Security definitions can be provided for such DEMs, which are independent of the security definition of the associated KEM. In [4] Cramer and Shoup show that a combined KEM-DEM encryption scheme is secure in the standard IND-CCA2 sense (for a public key encryption scheme) if the DEM is secure and the KEM is secure in the IND-CCA2 sense above. Hence, the goal is to define KEM's which are IND-CCA2 secure.

4 Review of Dent's Construction

In this section we recap on some prior work on classical KEM's, in particular a construction of a secure KEM given a public key encryption algorithm which is secure in the sense of OW-CPA [3].

We first turn our attention to probabilistic public key encryption schemes. We formally define these as a triple of algorithms:

- $\mathcal{G}(\mathbb{D})$ which is a probabilistic key generation algorithm. On input of $\mathbb{D}$, the domain parameters, this algorithm outputs a public/private key pair $(\mathtt{pk}, \mathtt{sk})$.
- $\mathcal{E}(m, \mathtt{pk})$ which is a probabilistic public key encryption algorithm. On input of a public key $\mathtt{pk}$ and a message $m \in \mathcal{M}$ this algorithm output a ciphertext c, it makes use of a random value drawn from a space $\mathcal{R}$.
- $\mathcal{D}(c, \mathtt{sk})$ which is a decryption algorithm. This takes as input a ciphertext c and a private key $\mathtt{sk}$ and outputs the associated message m or a special symbol $\perp$ representing the case where c is an invalid ciphertext with respect to the private key $\mathtt{sk}$.

For such a scheme to be useful we require that it is sound in the following sense,

$$\Pr\Big((\mathtt{pk}, \mathtt{sk}) \leftarrow \mathcal{G}(\mathbb{D}), m \overset{R}{\leftarrow} \mathcal{M}, c \leftarrow \mathcal{E}(m, \mathtt{pk}) : m = \mathcal{D}(c, \mathtt{sk})\Big) = 1.$$

We also require that the scheme is truly probabilistic in that the proportion of values of r, used as input into $\mathcal{E}(m, \mathtt{pk}; r)$, that encrypt a given message to a given ciphertext is negligible as a function of the security parameter.

We shall require the security notion of OW-CPA for public key schemes, which we recap on now. We assume an adversary $\mathcal{A}$ which takes a challenge ciphertext c^* and a public key and is asked to produce the associated plaintext. The scheme said to be OW-CPA secure if no adversary exists which wins the following game with probability greater than a negligible function of the security parameter k.

```
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathcal{G}(\mathbb{D})$.
$m \overset{R}{\leftarrow} \mathcal{M}$.
$c^* \leftarrow \mathcal{E}(m, \mathtt{pk})$.
$m' \leftarrow \mathcal{A}(\mathtt{pk}, c^*)$.
Output whether $m = m'$.
```

The adversary is not given access to any decryption oracles, but is clearly allowed to encrypt arbitrary messages of its choice since it has access to $\mathtt{pk}$.

Dent [3] proposes a KEM, derived from a OW-CPA probabilistic public key algorithm $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, a hash function H with codomain $\mathcal{R}$ (the space of randomness used by algorithm $\mathcal{E}$) and a key derivation function KDF with domain $\mathcal{M}$. Dent's scheme is described as follows:

```
$\mathcal{G}_{KEM}(\mathbb{D})$: $\mathcal{G}_{KEM} = \mathcal{G}$.

$\mathcal{E}_{KEM}(\mathtt{pk})$:
    $m \overset{R}{\leftarrow} \mathcal{M}$.
    $r \leftarrow H(m)$.
    $C \leftarrow \mathcal{E}(m, \mathtt{pk}; r)$.
    $K \leftarrow KDF(m)$.
    Output $(K, C)$.

$\mathcal{D}_{KEM}(C, \mathtt{sk})$:
    $m \leftarrow \mathcal{D}(C, \mathtt{sk})$.
    If $m = \perp$ then output $\perp$ and halt.
    $r \leftarrow H(m)$.
    Check that $C = \mathcal{E}(m, \mathtt{pk}; r)$,
            if not output $\perp$ and halt.
    $K \leftarrow KDF(m)$.
    Output $K$.
```

One then has the following result, when one models the functions H and KDF as random oracles,

Theorem 1 (Dent [3]). *If $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a OW-CPA probabilistic public key encryption algorithm then in the random oracle model the KEM*

$$(\mathcal{G}_{KEM}, \mathcal{E}_{KEM}, \mathcal{D}_{KEM})$$

derived from $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is secure in the sense of IND-CCA2.

5 mKEMs

We now extend the notion of KEM to deal with the case where one wants to encrypt a large amount of data to multiple people, say n people. In such a situation it makes sense to apply the DEM once and so one requires a mechanism which creates the symmetric key for the DEM and an encapsulation which encapsulates the key to many parties at once. We call such a system an mKEM for "multiple KEM".

Note, a trivial solution would be to generate a session key K for the DEM and then encrypt this to the various intended receivers by encrypting using an IND-CCA2 public key encryption algorithm. This would produce n distinct ciphertexts $c_1, \ldots, c_n$, each encrypting K for n different users. We would then define the key encapsulation as

$$C = c_1 \| c_2 \| \cdots \| c_n.$$

Our goal however is to do this in a more efficient manner, where one measures efficiency either in terms of computing resources or in terms of length of the resulting encapsulation C.

Note, in the above trivial system one would need to specify which ciphertext component c_i corresponded to which user u_i. Hence, the ciphertext should actually contain some information specifying which ciphertext corresponds to which user, i.e. we need to have something like

$$C = u_1 \| c_1 \| u_2 \| c_2 \cdots \| u_n \| c_n.$$

In our future discussion we shall drop this explicit reference to which users which component corresponds to. Instead we shall pass the list of recipients to the decryption function as an optional additional parameter.

Just as for a KEM, we define an mKEM formally as a triple of algorithms, $(\mathcal{G}_{mKEM}, \mathcal{E}_{mKEM}, \mathcal{D}_{mKEM})$, by adapting the earlier definition, we have

- $\mathcal{G}_{mKEM}(\mathbb{D})$ which is a probabilistic key generation algorithm. On input of $\mathbb{D}$, the domain parameters, this algorithm outputs a public/private key pair $(\mathtt{pk}, \mathtt{sk})$.
- $\mathcal{E}_{mKEM}(\mathcal{P})$ which is a probabilistic encapsulation algorithm. On input of a set of public key $\mathcal{P} = \{\mathtt{pk}_1, \ldots, \mathtt{pk}_n\}$ this algorithm outputs an encapsulated key-pair (K, C), where $K \in \mathbb{K}$ is the session key and C is an encapsulation of the key K under the public keys $\{\mathtt{pk}_1, \ldots, \mathtt{pk}_n\}$.

– $\mathcal{D}_{mKEM}(C, sk, \mathcal{P})$ which is a decapsulation algorithm. This takes as input an encapsulation C and a private key $\mathtt{sk}$, plus optionally the set of all recipients $\mathcal{P}$, 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 $\mathtt{sk}$.

Completeness is now defined as follows.

$$\Pr\left(\begin{array}{ll}(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathcal{G}_{mKEM}(\mathbb{D}) \forall i \in \{1, \dots, n\} & \\ (K, C) \leftarrow \mathcal{E}_{mKEM}(\{\mathtt{pk}_1, \dots, \mathtt{pk}_n\}), & \\ j \overset{R}{\leftarrow} \{1, \dots, n\} & : K = \mathcal{D}_{mKEM}(C, \mathtt{sk}_j)\end{array}\right) = 1.$$

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

```
$(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathcal{G}_{mKEM}(\mathbb{D}) \forall i \in \{1, \dots, n\}$.
$\mathcal{P}' \leftarrow \{\mathtt{pk}_1, \dots, \mathtt{pk}_n\}$
$\{s, \mathcal{P}\} \leftarrow \mathcal{A}^1(\mathcal{P}')$, where $\mathcal{P} \subseteq \mathcal{P}'$ and $m = \#\mathcal{P} \leq n$.
$b \overset{R}{\leftarrow} \{0, 1\}$.
$(K_b, C^*) \leftarrow \mathcal{E}_{mKEM}(\mathcal{P})$.
$K_{1-b} \overset{R}{\leftarrow} \mathbb{K}$.
$b' \leftarrow \mathcal{A}^2(C^*, \{K_0, K_1\}, s)$.
Output whether $b = b'$.
```

Notice in stage one the adversary picks a set $\mathcal{P}$ of public keys on which it wants to be challenged. Unlike the case of security models for multisignatures we do not allow the adversary to generate their own keys to be challenged on, after all if we allowed the adversary to generate its own public keys to be challenged on it could simply remember the corresponding private keys and decapsulate honestly.

One needs to be careful about the oracle access to the decapsulation oracle. To see why consider our earlier trivial mKEM with two parties, the challenge is given by

$$C^* = c_1 \| c_2.$$

However, using a traditional CCA2 definition of security an adversary could produce the encapsulation

$$C = c_1$$

and ask the decapsulation oracle to return the associated private key. Since $C \neq C^*$ this is a valid oracle query, which would result in the adversary breaking the system. However, we feel such an oracle query is too lenient for our purposes. We therefore restrict decapsulation oracle queries in the second stage to be only allowed if the resulting key is different from the key encapsulated by C^*. Such a restricted oracle access is used in other works to deal with the public key encryption algorithms which suffer from benign malleability, see for example [5].

We say an mKEM is (m, n)-IND-CCA2 secure, for an integers n and m with $m \leq n$, if the advantage of the adversary winning the above game is negligible as a function of the security parameter. We assume the adversary is allowed access to decapsulation oracle queries in both stages, subject to the above restriction on C^*.

6 Constructions of mKEMs

We start this section by giving a generic construction which mirrors the naive construction of the introduction. Then we go on to provide a more efficient construction based on the ElGamal encryption function.

6.1 A Generic Construction

We let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ denote a public key encryption algorithm which is IND-CCA2 secure, and let KDF denote a key derivation function with domain $\mathcal{M}$.

We define a KEM from $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ as follows, where $\mathcal{M}$ is the message space of $\mathcal{E}$ and $\mathcal{R}$ is the space of randomness used by $\mathcal{E}$,

```
$\mathcal{G}_{mKEM}(\mathbb{D})$:
    $(\texttt{sk}, \texttt{pk}) \leftarrow \mathcal{G}(\mathbb{D})$.
    Output $(\texttt{pk}, \texttt{sk})$.
```

```
$\mathcal{E}_{mKEM}(\{\texttt{pk}_1, \ldots, \texttt{pk}_n\})$:
    $m \overset{R}{\leftarrow} \mathcal{M}$.
    $r_i \overset{R}{\leftarrow} \mathcal{R}$ for all $i$.
    $c_i \leftarrow \mathcal{E}(m, \texttt{pk}_i; r_i)$ for all $i$.
    $K \leftarrow KDF(m)$.
    $C \leftarrow (c_1, \ldots, c_n)$.
    Output $(K, C)$.
```

```
$\mathcal{D}_{mKEM}(C, \texttt{sk}_i)$:
    Parse $C$ as $(c_1, \ldots, c_n)$.
    $m \leftarrow \mathcal{D}(c_i, \texttt{sk}_i)$.
    If $m = \perp$ then output $\perp$ and halt.
    $K \leftarrow KDF(m)$.
    Output $K$.
```

Theorem 2. *If $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is IND-CCA2 secure as a public key encryption scheme and n grows as a polynomial function of the security parameter then the mKEM $(\mathcal{G}_{mKEM}, \mathcal{E}_{mKEM}, \mathcal{D}_{mKEM})$ is (m, n)-IND-CCA2 for all $m \leq n$.*

Proof. Since $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is IND-CCA2 secure as a public key encryption algorithm it is secure in the multi-user setting described in [2].

We recap on the security definition from [2]. The adversary is given n public keys $\texttt{pk}_1, \ldots, \texttt{pk}_n$ and is given access to a left-right oracle $\mathcal{O}_{LR}$ which on input of $\{\{m_0, m_1\}, \texttt{pk}_i\}$ will output the encryption of m_b under $\texttt{pk}_i$ for some fixed hidden bit b. The adversary is given access to a decryption oracle $\mathcal{O}_D$ for all the public keys $\texttt{pk}_i$, subject to the constraint it is not allowed to ask for the decryption of the result of a call to the left-right oracle $\mathcal{O}_{LR}$.

We assume an adversary $\mathcal{A}$ against the mKEM $(\mathcal{G}_{mKEM}, \mathcal{E}_{mKEM}, \mathcal{D}_{mKEM})$ and show how one can use this to produce an adversary $\mathcal{B}$ against $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ in the above multi-user setting. Thus we will derive a contradiction.

Algorithm $\mathcal{B}$ takes as input the n public keys $\mathcal{P}' = \{\texttt{pk}_1, \ldots, \texttt{pk}_n\}$. These are then passed into algorithm $\mathcal{A}^1$. We answer the decapsulation oracle queries of $\mathcal{A}^1$ using the decryption provided to $\mathcal{B}$ in an obvious way, i.e. on input of $C = (c_1, \ldots, c_n)$ with respect to some public key $\texttt{pk}_i$ we execute

```
$m \leftarrow \mathcal{O}_D(c_i, \texttt{pk}_i)$.
If $m = \perp$ then output $\perp$ and halt.
```

```
    $K \leftarrow KDF(m)$.
    Output $K$.
```

Algorithm $\mathcal{A}^1$ eventually will terminate and will return a list of public keys

$$\mathcal{P} = \{\mathtt{pk}_{i_1}, \ldots, \mathtt{pk}_{i_m}\} \subseteq \mathcal{P}',$$

and a state s.

Algorithm $\mathcal{B}$ then computes two random messages $m_0, m_1 \in \mathcal{M}$ and computes $K_0 = KDF(m_0)$ and $K_1 = KDF(m_1)$. Then using the left-right oracles it computes

$$C^* = (c_{i_1}, \ldots, c_{i_m})$$

where

$$c_{i_j} = \mathcal{O}_{LR}(\{m_0, m_1\}, \mathtt{pk}_{i_j}).$$

One then executes $\mathcal{A}^2(C^*, \{K_0, K_1\}, s)$. The decapsulation oracle queries of $\mathcal{A}^2$ are answered as above on noting that any oracle query allowed in the game being played by $\mathcal{A}^2$ will be able to be answered by the oracle provided to $\mathcal{B}$.

Finally $\mathcal{A}^2$ will respond with its guess b' as to the hidden bit b, we let this bit b' be the output of $\mathcal{B}$. If $\mathcal{A}^2$ answers correctly then algorithm $\mathcal{B}$ will also answer correctly.

We note that the KDF function used need have very weak properties in the above construction. Namely, it maps a uniform distribution on its input space into a uniform distribution on its output space. There is no requirement on it being one-way or anything else. This should be constrasted with the construction in the next section where we require that the KDF is modelled as a random oracle.

6.2 An Efficient ElGamal Based mKEM

We now present an efficient mKEM based on the ElGamal encryption algorithm for a group G of prime order $q \approx 2^k$ with generator g. We let $\mathbb{D} = \{q, g, G\}$ denote the domain parameters of the scheme, ElGamal is then given by the triple of algorithms:

```
$\mathcal{G}(\mathbb{D})$:
    $\mathtt{sk} \xleftarrow{R} \mathbb{F}_q^*$.
    $\mathtt{pk} \leftarrow g^{\mathtt{sk}}$.
    Output $(\mathtt{pk}, \mathtt{sk})$.

$\mathcal{E}(m, \mathtt{pk}; r)$:
    $c_1 \leftarrow g^r$.
    $c_2 \leftarrow m \cdot \mathtt{pk}^r$.
    Output $(c_1, c_2)$.

$\mathcal{D}((c_1, c_2), \mathtt{sk})$:
    $m = c_2 / c_1^{\mathtt{sk}}$.
    Output $m$.
```

This is OW-CPA secure assuming the Diffie–Hellman problem for the group G is hard. Hence, the KEM derived from ElGamal by Dent's construction, is IND-CCA2 secure.

We now derive an mKEM from ElGamal by letting the key generation function be as in standard ElGamal. We then define encapsulation and decapsulation via

```
$\mathcal{E}_{mKEM}(\{\mathtt{pk}_1, \ldots, \mathtt{pk}_n\})$:
    $m \xleftarrow{R} G$.
    $r \leftarrow H(m)$.
    $c_0 \leftarrow g^r$.
    $c_i \leftarrow m \cdot \mathtt{pk}_i^r$ for $i = 1, \ldots, n$.
    $K \leftarrow KDF(m)$.
    $C \leftarrow (c_0, c_1, \ldots, c_n)$.
    Output $(K, C)$.
```

```
$\mathcal{D}_{mKEM}(C, \mathtt{sk}_i)$:
    Parse $C$ as $(c_0, c_1, \ldots, c_n)$.
    $m \leftarrow c_i / c_0^{\mathtt{sk}_i}$.
    $r \leftarrow H(m)$.
    If $c_0 \neq g^r$ the output $\perp$ and halt.
    $K \leftarrow KDF(m)$.
    Output $K$.
```

Theorem 3. *If n grows as a polynomial function of the security parameter k and the Diffie–Hellman problem in G is hard then, in the random oracle model the above mKEM is secure, for n users, in the sense of (m,n)-IND-CCA2 for mKEMs for all $m \leq n$.*

Proof. We let $(\mathcal{G}_{KEM}, \mathcal{E}_{KEM}, \mathcal{D}_{KEM})$ denote the ordinary KEM derived from the ElGamal system via Dent's transform for OW-CPA probabilistic public key algorithms. It is known, by Theorem 1, that in the random oracle model the scheme $(\mathcal{G}_{KEM}, \mathcal{E}_{KEM}, \mathcal{D}_{KEM})$ is secure in the IND-CCA2 sense assuming the Diffie–Hellman problem is hard.

We let $(\mathcal{G}_{mKEM}, \mathcal{E}_{mKEM}, \mathcal{D}_{mKEM})$ denote our mKEM. We shall assume we have an IND-CCA2 adversary $\mathcal{A}$ against this scheme in the random oracle model which works against n public keys. We shall show how to use $\mathcal{A}$ to create an adversary $\mathcal{B}$ against $(\mathcal{G}_{KEM}, \mathcal{E}_{KEM}, \mathcal{D}_{KEM})$. Since such an adversary is assumed, in the random oracle model, not to exist we can then conclude that $\mathcal{A}$ could not exist either.

We first describe algorithm $\mathcal{B}^1(\mathtt{pk})$. Let $\mathtt{pk}_1 = \mathtt{pk}$ denote the public key input into algorithm $\mathcal{B}^1$. We first generate some extra public keys via, $k_i \xleftarrow{R} \mathbb{F}_q^*$ and $\mathtt{pk}_i = \mathtt{pk} \cdot g^{k_i}$, for $i = 2, 3, \ldots, n$. We now pass the set $\mathcal{P}' = \{\mathtt{pk}_1, \ldots, \mathtt{pk}_n\}$ into $\mathcal{A}^1$. We then obtain a subset $\mathcal{P} \subseteq \mathcal{P}'$ and a state s'. We shall discuss how to answer all decapsulation oracle queries of $\mathcal{A}^1$ later. We let s denote the state

$$s = \{(k_2, \mathtt{pk}_2), \ldots, (k_n, \mathtt{pk}_n), \mathcal{P}, s'\}$$

and return s as the output of $\mathcal{B}^1(\mathtt{pk})$.

Algorithm $\mathcal{B}^2$ takes as input two keys K_0 and K_1, the state information s and an encapsulation C^* of one of the keys $K_b \in \{K_0, K_1\}$ from the algorithm $\mathcal{E}_{KEM}$ with respect to the public key $\mathtt{pk}$. We first need to create a valid encapsulation C_m^* of the key K_b with respect to the algorithm $\mathcal{E}_{mKEM}$ and the set of keys $\mathcal{P} = \{\mathtt{pk}_{i_1}, \ldots, \mathtt{pk}_{i_m}\}$. We have

$$C^* = (\mathfrak{c}_0^*, \mathfrak{c}_1^*) = (g^{\mathfrak{r}}, \mathfrak{m} \cdot \mathtt{pk}^{\mathfrak{r}}),$$

with $K_b = KDF(\mathfrak{m})$ and $\mathfrak{r} = H(\mathfrak{m})$, whereas

$$C_m^* = (c_0^*, c_1^*, \ldots, c_m^*) = (g^{\mathfrak{r}}, \mathfrak{m} \cdot \mathtt{pk}_{i_1}^{\mathfrak{r}}, \ldots, \mathfrak{m} \cdot \mathtt{pk}_{i_m}^{\mathfrak{r}}).$$

Hence, we set $c_0^* = \mathfrak{c}_0^*$ and for $j = 1, \ldots, m$

$$\begin{aligned} c_j^* &= \mathfrak{c}_1^* \cdot {\mathfrak{c}_0^*}^{k_{i_j}} \\ &= \mathfrak{m} \cdot \mathtt{pk}^r \cdot g^{\mathfrak{r} k_{i_j}} = \mathfrak{m} \cdot \left(\mathtt{pk} \cdot g^{k_{i_j}} \right)^{\mathfrak{r}} \\ &= \mathfrak{m} \cdot \mathtt{pk}_{i_j}^{\mathfrak{r}}, \end{aligned}$$

where we let $k_1 = 0$.

Having constructed C_m^* we can now pass $C_m^*, \{K_0, K_1\}$ and s' to algorithm $\mathcal{A}^2$. If $\mathcal{A}$ is a successful adversary against the mKEM then we will obtain with non-negligible probability a bit b' such that $K_b = K_{b'}$. Hence, by returning this bit b' as our output from the algorithm $\mathcal{B}^2$ we obtain an algorithm $\mathcal{B}$ which with non-negligible probability will break the security of the $(\mathcal{G}_{KEM}, \mathcal{E}_{KEM}, \mathcal{D}_{KEM})$ in the IND-CCA2 sense.

All that remains is to show how to answer the decapsulation oracle queries of algorithm $\mathcal{A}$. Recall we have a decapsulation oracle $\mathcal{O}_{KEM}$ for the scheme $(\mathcal{G}_{KEM}, \mathcal{E}_{KEM}, \mathcal{D}_{KEM})$ which will respond with respect to the key pk on all requests C except one is not allow to query it with $C = C^*$. The decapsulation queries of $\mathcal{A}$ must be answered correctly unless the query C_m corresponds to the key K_b.

Suppose we are given the, possibly invalid, encapsulation

$$C_m = (c_0, c_1, \ldots, c_m) = (g^r, m \cdot \mathtt{pk}_{i_1}^{r_1}, \ldots, m \cdot \mathtt{pk}_{i_m}^{r_m})$$

and we are asked to decapsulate it with respect to the public key $\mathtt{pk}_{i_j}$. This should result in the key $K = KDF(m)$ if and only if $r = H(m)$ and $r_j = r$.

We first form the encapsulation $(\mathfrak{c}_0, \mathfrak{c}_1)$ with respect to the scheme KEM, via setting $\mathfrak{c}_0 = c_0$ and

$$\begin{aligned} \mathfrak{c}_1 &= c_{i_j} \cdot c_0^{-k_{i_j}} \\ &= m \cdot \mathtt{pk}_{i_j}^{r_j} \cdot g^{-r k_{i_j}} \\ &= m \cdot \mathtt{pk}^{r_j} \cdot g^{(r_j - r) k_{i_j}} \\ &= m \cdot \mathtt{pk}^r \text{ if } r_j = r. \end{aligned}$$

Note since we are not allowed to query $\mathcal{A}$'s decapsulation oracle with any encapsulation which corresponds to K_b we must have $m \neq \mathfrak{m}$.

The oracle $\mathcal{O}$ will not respond if $\mathfrak{c}_0 = \mathfrak{c}_0^*$ and $\mathfrak{c}_1 = \mathfrak{c}_1^*$. Such a situation would mean that $\mathcal{O}$ returns K_b, i.e. the encapsulation C_m is an encapsulation of K_b with respect to $\mathtt{pk}_{i_j}$, and such a query is invalid under the security model for mKEMs.

We can, therefore, assume that either $\mathfrak{c}_0 \neq \mathfrak{c}_0^*$ or $\mathfrak{c}_1 \neq \mathfrak{c}_1^*$. In either case the oracle $\mathcal{O}$ will compute

$$m' = m \cdot \mathtt{pk}^{r_j} \cdot g^{(r_j - r) k_{i_j}} \cdot \mathfrak{c}_0^{-\mathtt{sk}} = m \cdot \left(\mathtt{pk} g^{k_{i_j}} \right)^{r_j - r}.$$

For the oracle $\mathcal{O}$ to return $K = KDF(m')$ we must have $r = H(m')$. In which case $(\mathfrak{c}_0, \mathfrak{c}_1)$ is a (possibly badly formed) ciphertext for the KEM which never-the-less passes the validity check and C_m is a (possibly badly formed) ciphertext for the mKEM which also passes the validity check of the mKEM. Hence, $\mathcal{A}$s oracle should return K, unless $K = K_b$ in which case we have found a collision in KDF since

$$KDF(\mathfrak{m}) = KDF(m').$$

Such a collision will only occur with negligible probability since KDF is modelled as a random oracle.

7 Efficiency Comparison

We first compare our ElGamal based mKEM for n users against naive concatenation of n ElGamal ciphertexts together. We let $EG(n)$ denote a IND-CCA2 version of ElGamal (such as EC-IES/DH-IES [1]) applied n-times to encrypt a session key. We let $EG_{KEM}(n)$ denote the ElGamal based mKEM described in Section 6.2 applied to n public keys. We compare the number of group exponentiations performed in the following table:

	$EG_{KEM}(n)$	$EG(n)$
Encapsulation	$n+1$	$2n+2$
Decapsulation	2	1

Hence we see that our method is more efficient than simply concatenating n-ElGamal ciphertexts together. In addition our method only requires the transmission of $n+1$ group elements as opposed to $2n$ group elements for the naive method.

References

1. M. Abdalla, M. Bellare and P. Rogaway. DHAES : An encryption scheme based on the Diffie–Hellman problem. Preprint, 1999.
2. M. Bellare, A. Boldyreva and S. Micali. Public-key encryption in the multi-user setting : Security proofs and improvements. In *Advances in Cryptology – EuroCrypt 2000*, Springer-Verlag LNCS 1807, 259–274, 2000.
3. A.W. Dent. A designer's guide to KEMs. To appear *Coding and Cryptography*, Springer-Verlag LNCS 2898, 133–151, 2003.
4. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen-ciphertext attack. Preprint, 2002.
5. V. Shoup. A proposal for the ISO standard for public-key encryption (version 2.0). Preprint, 2001.