A Designers Guide to KEMs

Alexander W. Dent[⋆]

Information Security Group,
Royal Holloway, University of London,
Egham Hill, Egham, Surrey, U.K.
alex@fermat.ma.rhul.ac.uk
http://www.isg.rhul.ac.uk/~alex/

Abstract. A generic or KEM-DEM hybrid construction is a formal method of combining a asymmetric and symmetric encryption techniques to give an efficient, provably secure public-key encryption scheme. This method combines an asymmetric KEM with a symmetric DEM, and each of these components must satisfy their own security conditions. In this paper we describe generic constructions for provably secure KEMs based on lower level primitives such as one-way trapdoor functions and weak key-agreement protocols.

1 Introduction

Whilst most dedicated public-key encryption algorithms are fine for sending short messages, many schemes have problems sending long or arbitrary length messages. Most of the normal "modes of operation" that allow a sender to send a long message using a public-key encryption algorithm directly are cripplingly inefficient and may well be weak.

One particular way to solve these problems is to use symmetric encryption with a randomly generated key to encrypt a message, and then use asymmetric cryptography to encrypt that (short) random key. This method has been considered cryptographic folklore for years and, as such, was not formally studied. This led to papers such as [2] which attacked schemes in which the set of symmetric keys is significantly smaller than the message space of the asymmetric scheme used to encrypt them. This folklore has recently been formalised in terms of a generic or KEM-DEM construction [3]. In this construction the encryption scheme is characterised into two parts: an asymmetric KEM and a symmetric DEM. A KEM (or key encapsulation mechanism) is a probabilistic algorithm that produces a random symmetric key and an an encryption of that key. A DEM (or data encapsulation mechanism) is a deterministic algorithm that encrypts a message of arbitrary length under the key given by the KEM.

[⋆] The information described in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. This paper has been supported by the Commission of the European Communities through the IST program under contract IST-1999-12324.

In this paper we provide generic methods for constructing KEMs from low level components such as encryption functions and key-agreement protocols. Furthermore we provide security proofs that show that the KEM will be highly secure given that these lower level components have certain weak security properties. Essentially this paper gives a toolbox to allow an algorithm designer to construct a KEM from almost any cryptographic problem.

2 Preliminaries

A KEM is a triple of algorithms:

– a key generation algorithm, $KEM.Gen$, which takes as input a security parameter 1^λ and outputs a public/private key-pair (pk, sk),
– a encapsulation algorithm, $KEM.Encap$, that takes as input a public-key pk and outputs an encapsulated key-pair (K, C) (C is sometimes said to be an encapsulation of the key K),
– a decapsulation algorithm, $KEM.Decap$, that takes as input an encapsulation of a key C and a secret-key sk, and outputs a key K.

Obviously if the scheme is to be useful we require that, with overwhelming probability, the scheme is sound, i.e. for almost all $(pk, sk) = KEM.Gen(1^\lambda)$ and almost all $(K, C) = KEM.Encap(pk)$ we have that $K = KEM.Decap(C, sk)$. We also assume that range of possible keys K is some set of fixed length binary strings, $\{0, 1\}^{KEM.Keylen(\lambda)}$.

A KEM is considered secure if there exists no attacker with a significant advantage in winning the following game played against a mythical system.

1. The system generates a public/private key-pair $(pk, sk) = KEM.Gen(1^\lambda)$ and passes pk to the attacker.
2. The attacker runs until it is ready to receive a challenge encapsulation pair. During this time the attacker may query a decapsulation oracle to find the key associated with any encapsulation.
3. The system prepares a challenge encapsulated key-pair as follows:
   (a) The system generates a valid encapsulated key-pair $(K_0, C) = KEM.Encap(pk)$.
   (b) The system selects an alternate key K_1 chosen uniformly at random from the set $\{0, 1\}^{KEM.Keylen(\lambda)}$.
   (c) The system selects a bit σ uniformly at random from $\{0, 1\}$.

   and then passes (K_σ, C) to the attacker.
4. The attacker is allowed to run until he outputs a guess σ' for σ. During this time the attacker may query an decapsulation to find the key associated with any encapsulation except the challenge encapsulation C.

The attacker is said to win this game if $\sigma' = \sigma$. We define an attacker's advantage Adv to be

$$Pr[\sigma' = \sigma] - 1/2$$

and the total advantage of the KEM to be the maximum advantage of any polynomial time attacker. If the total advantage of a KEM is negligible (as a function of λ) then the KEM is said to be IND-CCA2 secure.

Similarly, a DEM is a pair of algorithms:

– an encryption algorithm, $DEM.Encrypt$, that takes as input a message m and a key K, and produces a ciphertext C,
– and a decryption algorithm, $DEM.Decrypt$, the takes as input a ciphertext C and a key K, and produces a message m.

Again we require that a DEM is sound, that is for almost all keys K and messages m we have that $m = DEM.Decrypt(DEM.Encrypt(m, K), K)$. We also assume that the set of DEM keys is some set of fixed length binary strings, $\{0, 1\}^{DEM.Keylen(\lambda)}$.

A DEM is considered secure if there exists no attacker with a significant advantage in winning the following game played against a mythical system.

1. The system chooses a key K uniformly at random from the set possible DEM keys: $\{0, 1\}^{DEM.Keylen(\lambda)}$.
2. The attacker is allowed to run until it outputs two messages m_0 and m_1. During this time the attacker is NOT given access to a decryption oracle.
3. The system prepares a challenge ciphertext as follows
   (a) The system picks a bit σ uniformly at random from the set $\{0, 1\}$.
   (b) The system forms the ciphertext $C = DEM.Encrypt(m_\sigma, K)$.

   and passes this ciphertext back to the attacker.
4. The attacker then runs until outputs a guess σ' for σ. During this time it is allowed to query a decryption oracle to find the decryption of any ciphertext under the key K.

The attacker is said to win the game if $\sigma' = \sigma$. We define an attacker's advantage to be

$$Pr[\sigma' = \sigma] - 1/2$$

and the total advantage of the DEM to be the maximum advantage of any polynomial time attacker. If the total advantage is negligible (as a function of λ) then the DEM is said to be secure in the IND-CCA model.

A generic or KEM-DEM construction is an asymmetric encryption scheme made up of a KEM and a DEM (with $KEM.Keylen = DEM.Keylen$) in the following manner.

– Key-generation is provided by $KEM.Gen$.

– Encryption of a message m is given by the following algorithm:
  1. Generate an encapsulated key-pair $(K, C_1) = KEM.Encap(pk)$.
  2. Encrypt the message m by $C_2 = DEM.Encrypt(m, K)$.
  3. Output the ciphertext $C = (C_1, C_2)$.

– Decryption of a ciphertext C is given by the following algorithm:
  1. Parse C as (C_1, C_2).
  2. Compute the encapsulated key $K = KEM.Decap(C_1, sk)$.
  3. Decrypt the message $m = DEM.Decrypt(m, K)$.
  4. Output the message m.

It has been shown that the generic public-key encryption scheme constructed from an IND-CCA2 secure KEM and an IND-CCA secure DEM is itself IND-CCA2 secure in the standard sense [3].

3 Constructing KEMs from trapdoor one-way functions

We start by showing that the construction ideas used in RSA KEM [7] generalise to almost all one-way public-key encryption algorithms. Consider an encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ where

– $\mathcal{G}$ is a key-generation algorithm that takes as input a security parameter λ and outputs a public/private key-pair (pk, sk),
– $\mathcal{E}$ is the encryption algorithm that takes as input a message $m \in \mathcal{M}$ and the public-key and outputs a ciphertext $C \in \mathcal{C}$,
– and $\mathcal{D}$ is the decryption algorithm that takes as input a ciphertext $C \in \mathcal{C}$ and the secret-key sk and outputs either a message $m \in \mathcal{M}$ or the error symbol $\perp$.

Definition 1. *A public-key encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is said to be one-way if the probability that a polynomial time attacker $\mathcal{A}$ can invert a randomly generated ciphertext C is negligible as a function of λ. Such a cryptosystem is often referred to as being secure in the OW-CPA model*[1].

We begin by restricting ourselves to the case where the encryption function acts as a permutation on the message space $\mathcal{M}$ (hence $\mathcal{M} = \mathcal{C}$). One may use the scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ along with a suitable key-derivation function $KDF(\cdot)$ to construct a KEM in the following manner.

– Key generation is given by the key generation algorithm of the public-key encryption scheme (i.e. $KEM.Gen = \mathcal{G}$).

– Encapsulation is given by:
  1. Generate an element $r \in \mathcal{M}$ uniformly at random.
  2. Set $C := \mathcal{E}(r, pk)$.
  3. Set $K := KDF(r)$.
  4. Output (K, C).

[1] OW for "one-way" and CPA for "chosen plaintext attack". The term chosen plaintext attack is used because the attacker is not allowed to make decryption queries.

– Decapsulation of an encapsulation C is given by:
  1. Set $r := \mathcal{D}(C, sk)$.
  2. Set $K := KDF(r)$.
  3. Output K.

Theorem 1. *Suppose $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a one-way public-key encryption scheme such that, for all public-keys pk, the encryption function $\mathcal{E}(\cdot, pk)$ acts as a permutation on the set of messages. Then the KEM derived from $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is, in the random oracle model, IND-CCA2 secure.*

Proof. We intend to show that if there exists an attacker $\mathcal{A}$ that successfully breaks the scheme, i.e. breaks the scheme with non-negligible advantage, then there exists an algorithm that inverts the one-way encryption scheme with some non-negligible probability. In order to do this we need to slightly tweak the environment in which we run the attacker $\mathcal{A}$.

Let Game 1 be the normal IND game, as described in Section 2. Let Game 2 be similar to Game 1 but with the challenge key-pair (K_σ, C) chosen at the start (although not given to the attacker). If the attacker $\mathcal{A}$ queries the decryption oracle at any point with the ciphertext C then the decryption oracle responds with the error symbol $\perp$. Notice that the only difference in the two games occurs if $\mathcal{A}$ queries the decryption oracle with the ciphertext C before he has been given challenge key-pair (K_σ, C). In order to analyse the effects of this change of environment on the advantage of $\mathcal{A}$ we need the following simple lemma.

Lemma 1. *If A, B and E are events in some probability space and $Pr[A \mid \neg E] = Pr[B \mid \neg E]$ then $|Pr[A] - Pr[B]| \leq Pr[E]$.*

Let A be the event that $\mathcal{A}$ wins in Game 1, i.e. that he correctly identifies σ from (K_σ, C), and let B be the event that $\mathcal{A}$ wins in Game 2. Let E be the event that $\mathcal{A}$ asks for the decryption of C before the challenge is issued. Note that if E does not occur then $\mathcal{A}$ receives exactly the same information from the oracles in both games and so $Pr[A|\neg E] = Pr[B|\neg E]$. Hence

$$|Pr[A] - Pr[B]| \leq Pr[E].$$

Suppose that $\mathcal{A}$ makes at most q_D decryption queries. Since C is chosen uniformly at random from the set of possible ciphertexts and $\mathcal{A}$ has no knowledge of C before being issued with the challenge ciphertext, we have that

$$Pr[E] \leq \frac{q_D}{|\mathcal{M}|}.$$

Since the underlying encryption scheme is one-way we must have that $\frac{1}{|\mathcal{M}|}$ is negligible as a function of λ. Hence $Pr[B]$ is only negligible less than $Pr[A]$. So,

Lemma 2. *If there exists an attacker for a KEM with non-negligible advantage in Game 1 then there exists an attacker for that KEM with non-negligible advantage in Game 2.*

We may now proceed along more standard lines and prove that if there exists an attacker $\mathcal{A}$ that breaks the KEM in Game 2 with advantage Adv then we can construct an algorithm that inverts the one-way public-key encryption scheme with probability comparable to Adv. In doing this we will model the key derivation function KDF as a random oracle. Consider the following algorithm:

1. Prepare two lists $KDFList$ and $DecList$, initially setting them both to be empty.
2. We receive a public-key pk from the system and pass this straight to $\mathcal{A}$. We also receive a challenge ciphertext C^* to invert which we do not yet pass to $\mathcal{A}$.
3. Allow $\mathcal{A}$ to run until it requests a challenge encapsulated key-pair. During this time $\mathcal{A}$ can ask the system for the decapsulation of a encapsulated key C or to evaluate the key-derivation function $KDF(\cdot)$ (modelled as a random oracle) on a given input. If $\mathcal{A}$ asks for the decapsulation of the ciphertext C we perform the following steps:
   (a) Check to see if $C = C^*$. If so, output $\perp$ and halt.
   (b) Check to see if there exists an entry $(C, K) \in DecList$. If so, output K and halt.
   (c) Generate a key K uniformly at random from the set $\{0,1\}^{KEM.Keylen}$ and add (C, K) to $DecList$.
   (d) Output K.

   If $\mathcal{A}$ asks the system to evaluate the key derivation function $KDF(\cdot)$ on the input X then we perform the following steps:
   (a) Check to see if there exists an entry of the form $(X, K) \in KDFList$. If so, output K and halt.
   (b) Generate a key K uniformly at random from the set $\{0,1\}^{KEM.Keylen}$.
   (c) Check to see if $X \in \mathcal{M}$. If not, add (X, K) to $KDFList$ then output K and halt.
   (d) Let $C = \mathcal{E}(X, pk)$. If $C = C^*$ then output X as the inverse of C^* and terminate the entire program.
   (e) Check to see if there exists an entry of the form $(C, K') \in DecList$. If so, add (X, K') to $KDFList$ then output K' and halt.
   (f) Add (C, K) to $DecList$ and output K.
4. In order to construct a challenge key-pair we generate a key K^* uniformly at random from the set $\{0,1\}^{KEM.Keylen}$. We pass the pair (K^*, C^*) to $\mathcal{A}$ as the challenge.
5. Allow $\mathcal{A}$ to run until it outputs a bit σ'. During this time we answer all queries $\mathcal{A}$ makes in exactly the same way as before.
6. Lastly, if the program has not yet terminated, then we generate a guess X uniformly at random from $\mathcal{M}$ and output X as the inverse of C^*.

We can bound the probability of success of this algorithm in inverting the challenge ciphertext C^* in terms of the probability of success of the attacker $\mathcal{A}$ in winning Game 2. Let F be the event that $\mathcal{A}$, at some point, queries the key derivation function oracle with the value $X^* = \mathcal{D}(C^*, sk)$. Thus the probability of inverting C^* is given by:

$$Pr[Output\ X^*] = Pr[Output\ X^*|F]Pr[F] + Pr[Output\ X^*|\neg F]Pr[\neg F].$$

Trivially we can see that

$$Pr[Output\ X^*|F] = 1.$$

It therefore suffices to bound $Pr[F]$. If F does not occur then the attacker can have no knowledge of $KDF(X^*) = KEM.Decap(C^*, sk)$ as $KDF(\cdot)$ is a random oracle, hence its advantage in that situation is 0. It is easy to show that, since $\mathcal{A}$ has advantage Adv, the probability that F occurs is at least Adv. Therefore,

$$\begin{aligned} Pr[OutputX^*] &= Pr[OutputX^*|F]Pr[F] + Pr[OutputX^*|\neg F]Pr[\neg F] \\ &\geq Pr[F] \\ &\geq Adv \end{aligned}$$

In particular, if $\mathcal{A}$ has a non-negligible advantage then there exists an algorithm that inverts the one-way function with non-negligible probability. Putting this all together we get:

Lemma 3. *If there exists an attacker for the KEM with non-negligible advantage then there exists an algorithm that inverts the underlying encryption scheme with non-negligible probability. Conversely, if the underlying encryption scheme is one-way then the KEM is IND-CCA2 secure.*

Hence Theorem 1 holds. □

Now suppose that $\mathcal{E}(\cdot, pk)$ is not a permutation (i.e. $\mathcal{M} \neq \mathcal{C}$) but instead a deterministic function, i.e. $\mathcal{E}(\mathcal{M}, pk) \subseteq \mathcal{C}$. In this situation a decryption query always reveals some information about a ciphertext: it reveals if that ciphertext lies in $\mathcal{E}(\mathcal{M}, pk)$ or not. If such queries reveal information about the secret key then the above construction will be weak even if the underlying encryption scheme is one-way.

Definition 2. *A public-key encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is said to be secure in the OW-CPA$^+$ model if it is one-way even when the attacker has access to an oracle $\mathcal{O}$ that decides if a ciphertext C lies in $\mathcal{E}(\mathcal{M}, pk)$.*

This follows an idea first proposed in a paper by Joye, Quisquater and Yung [4]. There paper attacks an early version of EPOC-2, which is a hybrid cipher based on the Okamoto-Uchiyama cryptosystem [6]. This underlying cipher is secure in the OW-CPA model but not in the OW-CPA$^+$ model. Given a scheme that is secure in the OW-CPA$^+$ model then the same construction given above can be used to construct a secure KEM.

Lemma 4. *Suppose $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a deterministic public-key encryption scheme that is secure in the OW-CPA$^+$ model. Then the above construction for a KEM is secure in the IND-CCA2 sense.*

Sketch Proof The proof is very similar to the proof of Lemma 1. The only difference is that, when we are running the algorithm that breaks the one-way encryption scheme, we answer decryption queries from $\mathcal{A}$ by first checking whether that ciphertext has a valid decryption. If the ciphertext has no valid decryption then we return $\perp$, otherwise we simulate decryptions as before. $\square$

4 Constructing KEMs from probabilistic encryption schemes

We now also open ourselves up to the possibility that the encryption algorithm is not deterministic but probabilistic. The first thing to note is that our current construction is not sufficient in the case of probabilistic encryption. Even if an encryption scheme is secure in the OW-CPA$^+$ model, it might still be possible (in the case of probabilistic encryption) to find, for any given ciphertext C, a ciphertext $C' \neq C$ such that $\mathcal{D}(C', sk) = \mathcal{D}(C, sk)$. This is sufficient to break the KEM in the adaptive chosen ciphertext model. Hence we propose a new construction, based on an encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$.

- Key generation for the KEM is provided by the key generation algorithm for the underlying scheme, $\mathcal{G}$.
- Encapsulation is provided by the following algorithm.
  1. Generate an element r uniformly at random from the set $\mathcal{M}$.
  2. Set $C := \mathcal{E}(r, sk)$.
  3. Set $K := KDF(\bar{r}||C)$, where $\bar{r}$ is a fixed length representation of the element $r \in \mathcal{M}$.
  4. Output (K, C).
- Decapsulation of an encapsulation C is given by the following algorithm.
  1. Set $r := \mathcal{D}(C, sk)$.
  2. Set $K := KDF(\bar{r}||C)$ where $\bar{r}$ is a fixed length representation of the element r.
  3. Output K.

Now, for the security proof, to maintain consistency between decryption queries and random oracle queries we need to have access to an oracle that can decide whether a given ciphertext C is an encryption of a given message m. We specifically mention this oracle instead of including it inside some kind of attack model because this oracle is usually supplied as part of the intractability assumptions (for example, the Gap problems [5]) or is easily simulated in the random oracle model. Alternatively we could note that a scheme that is one-way even when the attacker has access to a decryption oracle (the OW-CCA2 model [1]) will certainly be secure in the OW-CPA$^+$ model, even if the attacker has access to a model that can decide if a given ciphertext is the encryption of a given message or not.

Theorem 2. *Suppose $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a public-key encryption scheme that is OW-CPA$^+$ secure when the attacker has access to an oracle that checks whether a ciphertext C (different from the challenge ciphertext) is an encryption of a message m. Then the KEM constructed from that scheme is, in the random oracle model, IND-CCA2 secure.*

Sketch Proof See Appendix A.

5 Constructing KEMs from key-agreement functions

One can not only build KEMs from encryption algorithms but also from simple key-agreement protocols. This should not be surprising considering the similarities between the two types of protocol! For our purposes we define a key-agreement scheme to be a triple of algorithms $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{G}$ is a parameter generation algorithm that takes as input a security parameter 1^λ and outputs a public/private key-pair (pk, sk) that will be used to generate a shared key.
- $\mathcal{E}$ is a deterministic encapsulation algorithm that takes as input a public-key pk and a fixed length random seed $r \in \mathcal{R}$, and outputs a pair encapsulated key-pair (K, C), where K is a key and C is an encapsulation of that key.
- $\mathcal{D}$ is an decapsulation algorithm that takes an encapsulation C and a secret key sk, and outputs a key K (or the error symbol $\perp$.

Notice that these definitions are very similar to the definition of a KEM itself. This is not accidental! However there are fundamental differences between KEMs and key agreement protocols. Whilst a KEM and a simple key-agreement protocol are functionally very similar, they have very different security requirements. In particular a key-agreement protocol is designed to protect a key for a long period of time whereas a KEM is designed only to protect a key that is used once. This means we can use a very weak notion of security for the underlying key agreement protocol and still get full security in the KEM.

For our purposes we will define the security of a key agreement protocol as follows.

Definition 3. *A key-agreement protocol $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is said to be one-way secure if, given a randomly generated encapsulation C, the probability that any polynomial time attacker $\mathcal{A}$ can find the corresponding key $K = \mathcal{D}(C, sk)$ is negligible as a function of λ.*

We shall refer to a one-way secure key-agreement protocol as being secure in the OW-CPA model. If the scheme is secure even when an attacker has access to an oracle that checks whether a given encapsulation C is valid (or if $\mathcal{D}(C, sk) = \perp$), then that scheme is said to be secure in the OW-CPA$^+$ model.

It might also be necessary to allow the attacker access to an oracle that determines if (K, C) is a valid encapsulation or not. However we chose to leave this out of the formal model, as before, partly to respect the model defined by Joye *et al.* and partly because this oracle is usually easily simulated in the random oracle model or available to the attacker due to the intractability assumption (as is the case in the Gap Diffie-Hellman assumption [4]).

We define a generic KEM based on a key-agreement protocol $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ as follows.

– Key generation is given by the key generation algorithm $\mathcal{G}$ of the key-agreement protocol.

– The encapsulation algorithm is given by:
  1. Generate an element r uniformly at random from the set $\mathcal{R}$.
  2. Set $(K_{raw}, C) := \mathcal{E}(r, sk)$.
  3. Set $K := KDF(\bar{K}_{raw}||C)$ where $\bar{K}_{raw}$ is a fixed length binary representation of K_{raw}.
  4. Output (K, C).

– The decapsulation algorithm, for a ciphertext C, is given by:
  1. Set $K_{raw} := \mathcal{D}(C, sk)$.
  2. Set $K := KDF(\bar{K}_{raw}||C)$ where $\bar{K}_{raw}$ is a fixed length binary representation of K_{raw}.
  3. Output K.

Theorem 3. *Suppose that $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a key-agreement protocol that is secure in the OW-CPA$^+$ model, even when the attacker has access to an oracle that decides whether a given encapsulation C (different from the challenge encapsulation) is an encapsulation of a given key K_{raw}. Then the KEM constructed in the above manner from this key-agreement scheme is IND-CCA2 secure.*

The proof of this theorem is similar to that of Theorem 2.

6 Relaxing the security requirement

We may reduce the need for the scheme to be secure in the OW-CPA$^+$ model to the need for scheme to simply be secure in the OW-CPA model if we are prepared to reduce the efficiency of the cipher. The technique we use is based on the masking approach used by PSEC-KEM [7]. We will consider constructing this KEM using a probabilistic encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$. The case where the KEM is constructed from a key-agreement protocol is analogous.

Suppose that $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a public-key encryption scheme and that $KDF(\cdot)$ and $MGF(\cdot)$ are suitably independent hash based functions (KDF is a key-derivation function and MGF is a mask-generation function). We also use a hash function ϕ that maps suitably sized bit strings to elements of the message space $\mathcal{M}$ of the asymmetric encryption scheme. Consider a KEM built in the following manner:

– Key-generation is given by $\mathcal{G}$, i.e. $KEM.Gen = \mathcal{G}$.

– Encapsulation is given by:
  1. Generate a suitably large bit string $r \in \{0,1\}^N$.
  2. Split r into two strings $r_1 \in \{0,1\}^{n_1}$ and $r_2 \in \{0,1\}^{n_2}$ where $n_1+n_2 = N$.
  3. Set $r_2' := \phi(r_1)$.
  4. Set $C_1 := \mathcal{E}(r_2', pk)$.
  5. Set $C_2 := r \oplus MGF(\bar{r_2}||C_1)$ where $\bar{r_2}$ is a fixed length representation of r_2'.
  6. Set $C := (C_1, C_2)$.
  7. Set $K := KDF(r_1)$.
  8. Output (K, C).

– Decapsulation of a ciphertext C is given by:
  1. Parse C as (C_1, C_2).
  2. Set $r_2' := \mathcal{D}(C_1, sk)$.
  3. Set $r := C_2 \oplus MGF(\bar{r_2}||C_1)$ where $\bar{r_2}$ is a fixed length representation of r_2'.
  4. Split r into two strings $r_1 \in \{0,1\}^{n_1}$ and $r_2 \in \{0,1\}^{n_2}$.
  5. Check that $r_2' = \phi(r_2)$. If not, output $\perp$ and halt.
  6. Set $K := KDF(r_1)$.
  7. Output K.

Theorem 4. *Suppose $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a public-key encryption scheme that is one-way (OW-CPA) secure and for which there exists an oracle that checks whether a ciphertext C (different from the challenge ciphertext) is an encryption of a message m. Then the KEM derived from this scheme (in the above manner) is IND-CCA2 secure.*

It is necessary for the security proofs that n_1 and n_2 are suitably large. Certainly the values

$$n_1 \geq \lambda \qquad n_2 \geq \lambda$$

are sufficient.

Sketch Proof of Theorem 4 See Appendix B.

7 Conclusion

This paper has provided generic constructions for key-encapsulation mechanisms based on some very simple cryptographic protocols. Indeed, these results mean that a KEM can be constructed based on almost any cryptographic problem. The abundance of KEMs and their variety serve to reinforce the idea that the KEM-DEM method of constructing a public-key encryption scheme is a very sound model. Indeed the easy with which one can construct secure KEMs, and so secure hybrid encryption schemes, suggests that this is an area with enormous practical potential.

Acknowledgements

I would like to thank Victor Shoup, Louis Granboulan and Kenny Paterson for some very useful discussions in this area.

References

1. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology – Crypto 98*, LNCS 1462, pages 26–45. Springer-Verlag, 1998.
2. D. Boneh, A. Joux, and A. Nguyen. Why textbook ElGamal and RSA encryption are insecure. In *Advances in Cryptology – Aisacrypt 2000*, LNCS 1976, pages 30–43. Springer-Verlag, 2000.
3. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. Technical report, http://shoup.net/, 2002.
4. 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*, LNCS 2020, pages 208–222. Springer-Verlag, 2001.
5. T. Okamoto and D. Pointcheval. The gap problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography*, LNCS 1992, pages 104–118. Springer-Verlag, 2001.
6. T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology – Eurocrypt 98*, LNCS 1403, pages 308–318. Springer-Verlag, 1998.
7. V. Shoup. A proposal for the ISO standard for public-key encryption (version 2.0). Available from `http://shoup.net/`, 2001.

A Sketch Proof of Theorem 2

This proof runs along lines that are very similar to the proof of Theorem 1. Suppose there exists an attacker $\mathcal{A}$ which can distinguish a valid key-pair from an invalid key-pair in the normal IND environment described in Section 2.

Let Game 1 be the normal IND environment for $\mathcal{A}$ (as described in Section 2). Let Game 2 be similar to Game 1 but with the challenge key-pair (K_σ, C) chosen at the start. If $\mathcal{A}$ queries the decryption oracle at any time with the ciphertext C then the decryption oracle responds with $\perp$.

Let A be the event that $\mathcal{A}$ wins (i.e. correctly identifies if (K_σ, C) is a valid key-pair or not) in Game 1. Let B be the event that $\mathcal{A}$ wins in Game 2. Then, exactly the same way as in the proof of Theorem 1, then we can show that

$$|Pr[A] - Pr[B]| \leq \frac{q_D}{|\mathcal{M}|}$$

where q_D is the number of decryption queries that $\mathcal{A}$ makes.

Hence, if there exists an attacker with non-negligible advantage in Game 1 then there exists an attacker with non-negligible advantage in Game 2.

Now we show that if there exists an attacker with advantage *Adv* in Game 2 then there exists an algorithm that inverts the encryption function with probability comparable to *Adv*. Again we model the key derivation function $KDF(\cdot)$ as a random oracle. Consider the following algorithm to invert the encryption function:

1. Prepare two lists $KDFList$ and $DecList$, initially setting both of them to be empty.
2. We receive a challenge ciphertext C^* to invert.
3. Firstly we run $\mathcal{A}$ with the proper public-key pk until it request a challenge encapsulated key-pair. During this time $\mathcal{A}$ can ask for the decapsulation of any encapsulated key C or to evaluate the key derivation function $KDF(\cdot)$ on a given input. If $\mathcal{A}$ asks for the decapsulation of an encapsulation C then we perform the following steps:
   (a) Check to see if $C = C^*$. If so, output $\perp$ and halt.
   (b) Check to see if there exists an entry $(C, K) \in DecList$, for some K. If so, output K and halt.
   (c) Check to see if $C \in \mathcal{E}(\mathcal{M}, pk)$ using the appropriate oracle. If not, output $\perp$ and halt.
   (d) Generate a new key K uniformly at random from the set $\{0,1\}^{KEM.Keylen}$ and add (C, K) to $DecList$.
   (e) Output K.

   If $\mathcal{A}$ asks for the evaluation of $KDF(X)$ then we perform the following steps:
   (a) Check to see if there exists an entry $(X, K) \in KDFList$, for some K. If so, output K and halt.
   (b) Check to see if we can parse X as $\bar{r}||C$ for some representation $r \in \mathcal{M}$ and $C \in \mathcal{C}$. If not, generate a key K uniformly at random from the set $\{0,1\}^{KEM.Keylen}$, add (X, K) to $KDFList$, output K and halt.
   (c) Check to see if $C = C^*$. If so, generate a key K uniformly at random from the set $\{0,1\}^{KEM.Keylen}$, add (X, K) to $KDFList$ then output K and halt.
   (d) Check to see if C is an encryption of the message r (using the appropriate oracle). If not, generate a key K uniformly at random from the set $\{0,1\}^{KEM.Keylen}$, add (X, K) to $KDFList$ then output K and halt.
   (e) Check to see if there exists an entry (C, K) in $DecList$. If so, add (X, K) to $KDFList$, output K and halt.
   (f) Generate a key K uniformly at random from the set $\{0,1\}^{KEM.Keylen}$. Add (X, K) to $KDFList$, add (C, K) to $DecList$ then output K and halt.
4. When $\mathcal{A}$ requests a challenge key-pair generate a key K^* uniformly at random from the set $\{0,1\}^{KEM.Keylen}$ and pass the pair (K^*, C^*) to $\mathcal{A}$.
5. Allow $\mathcal{A}$ to run until it outputs a bit σ'. During this time answer all the queries that $\mathcal{A}$ makes as before.
6. Lastly, select an entry (X, K) uniformly at random from $KDFList$ such that $X = \bar{r}||C^*$ and output r as the inverse of C^*. If no such pair exists then select $r \in \mathcal{M}$ uniformly at random and output r as the inverse of C^*.

As before we can show that the probability that the above algorithm outputs $r^* = \mathcal{D}(C^*, sk)$ is greater than or equal to the probability that the key derivation function is queried on the input $\bar{r^*}||C^*$. The probability that the key derivation function is queried on the input $\bar{r^*}||C^*$ is, in turn, greater than or equal to the advantage that $\mathcal{A}$ has in winning Game 2. Hence the probability that the above algorithm outputs the inverse of C^* is at least $\frac{1}{q_K}$.

So, if there exists an attacker with non-negligible advantage that correctly distinguishes between a valid and invalid KEM key-pair output then there exists an algorithm that inverts the underlying encryption function with non-negligible probability. Or, equivalently, if the underlying encryption scheme is one-way then the KEM IND-CCA2 secure.

B Sketch Proof of Theorem 4

The proof of Theorem 4 is, again, very similar to the proof of Theorem 1 but several orders of magnitude more difficult. We will give a sketch of the proof here.

Assume that there exists an attacker $\mathcal{A}$ which breaks the KEM in the IND-CCA2 sense. We will use this to construct an algorithm that inverts the underlying cryptosystem. Assume that $\mathcal{A}$ makes at most q_D queries to the decryption oracle, q_K queries of the key derivation function oracle, q_M queries of the mask generating function oracle and q_H queries of the hash function ϕ. In constructing the an algorithm that inverts the underlying cryptosystem we will be challenged to invert a random ciphertext C^* (we will use the superscript * to denote variables associated with the challenge ciphertext).

We start on lines very similar to those used in Appendix A. Let Game 1 be the game that an attacker plays normally. Let Game 2 be the game where the system prepares a challenge ciphertext (C^*, C'^*) at the start (where C'^* is generated uniformly at random from the set of binary strings of length N) and returns $\perp$ if the attacker requests the decryption of (C^*, C'^*) *at any time*. As before it is simple to show that if a scheme is secure in Game 1 then it is secure in Game 2.

Now consider what information the attacker gets back if he queries a ciphertext (C^*, C_2). Let C_2^1 denote the first n_1 bits of C_2 and let C_2^2 denote the last n_2 bits of C_2. If C_2^2 is the same as the last n_2 bits of of C'^* then the ciphertext will pass the integrity check in step 5 of the decryption algorithm and so the decryption oracle will output a key of some kind. If C_2^2 is not the same as the last n_2 bits of C'^* then, with probability $\frac{1}{|\mathcal{M}|}$ the integrity check will fail and the decryption oracle will output $\perp$.

Let Game 3 be similar to Game 2 but in Game 3, if the attacker requests the decryption of a ciphertext (C^*, C_2) where the last n_2 bits of C_2 are not the same as those of C'^* then the system returns $\perp$. Again we can bound the difference in the probability that an attacker succeeds in both of these games using Lemma 1. In this case the success probabilities differ by at most $\frac{q_D}{|\mathcal{M}|}$.

We can now construct an algorithm that uses a successful attacker $\mathcal{A}$ against the KEM to invert the underlying encryption function. The algorithm is constructed in a manner similar to that used in the proof of Theorem 1 and Theorem 2, the only problem arises in constructing the simulators. We give the specification of the simulators below. Remember that, when constructing the simulators, we have access to an oracle $\mathcal{O}$ that decides whether a given ciphertext C is an encryption of a given message m in the underlying encryption scheme.

The simulators require several lists, $MGFList$, $KDFList$, $PhiList$ and $DecList$, which should be initially empty. It also requires a string $Mask^*$ that has been generated uniformly at random from $\{0,1\}^{n_1}$. The simulation of the key derivation function and the mask generating function ϕ are easy. If the attacker requests the evaluation of the key derivation function KDF on the input X then the following steps are performed:

1. Check to see if there exists a pair $(X, K) \in KDFList$ for some K. If so, output K and halt.
2. Otherwise generate a bit string K uniformly at random from the output set $\{0,1\}^{KEM.Keylen(\lambda)}$, add (X, K) to $KDFList$ and output K.

If the attacker requests the evaluation of the hash function ϕ on the input X then the following steps are performed:

1. Check to see if there exists a pair $(X, Hash) \in PhiList$ for some $Hash$. If so, output $Hash$ and halt.
2. Otherwise generate an element $Hash \in \mathcal{M}$ uniformly at random, add $(X, Hash)$ to $PhiList$ and output $Hash$.

If the attacker requests the decryption of a ciphertext (C_1, C_2) then the following steps are performed:

1. Check to see if $(C_1, C_2, Mask, K) \in DecList$ for some values of $Mask$ and K. If so, output K and halt.
2. Check to see if $(C_1, C_2) = (C^*, C'^*)$. If so, output $\perp$ and halt.
3. If $C_1 = C^*$ and the last n_2 bits of C_2 are not equal to the last n_2 bits of C'^* then output $\perp$ and halt.
4. If $C_1 = C^*$ and the last n_2 bits of C_2 are equal to the last n_2 bits of C'^* then set X to be equal to the XOR of the first n_1 bits of C_2 and $Mask^*$. Then evaluate $K = KDF(X)$, output K and halt.
5. Check to see if $(C_1, C_2', Mask, K') \in DecList$ for any values of C_2', $Mask$ and K'. If so, let X be the first n_1 bits of $C_2 \oplus Mask$ and evaluate $K = KDF(X)$. Add $(C_1, C_2, Mask, K)$ to $DecList$, output K and halt.
6. Check to see if there exists $(\bar{r}||C_1, Mask) \in MGFList$ with r the decryption of C. If so, let X be the first n_1 bits of $C_2 \oplus Mask$ and evaluate $K = KDF(X)$. Add $(C_1, C_2, Mask, K)$ to $DecList$, output K and halt.
7. Generate $Mask$ uniformly at random from $\{0,1\}^N$. Let X be the first n_1 bits of $C_2 \oplus Mask$ and evaluate $K = KDF(X)$. Add $(C_1, C_2, Mask, K)$ to $DecList$ and output K.

Lastly, if the attacker requests the evaluation of the mask generating function MGF on the input X then the following steps are performed:

1. Check to see if $(X, Mask) \in MGFList$ for any value of $Mask$. If so, output $Mask$ and halt.
2. Check to see if X can be parsed as $\bar{r}||C$ for some $r \in \mathcal{M}$ and $C \in \mathcal{C}$. If not, generate $Mask$ uniformly at random from the set $\{0,1\}^N$ and add $(X, Mask)$ to $MGFList$ then output $Mask$ and halt.
3. Check to see if C is an encryption of r (using the appropriate oracle). If not, generate $Mask$ uniformly at random from the set $\{0,1\}^N$ and add $(X, Mask)$ to $MGFList$ then output $Mask$ and halt.
4. Check to see if there exists $(C, C', Mask, K) \in DecList$ for some values C', $Mask$ and K. If so add $(X, Mask)$ to $MGFList$, output $Mask$ and halt.
5. Generate $Mask$ uniformly at random from $\{0,1\}^N$. Add $(X, Mask)$ to $MGFList$ and output $Mask$.

Here the simulation of the mask generating function checks that its output will be consistent with that of the decryption oracle simulator and vice versa.

We will eventually select our final output X, our guess for the inverse of C^*, from the set of inputs to the mask-generating function MGF of the form $\bar{X}||C^*$. Let $X^* = \mathcal{D}(C^*, sk)$. It can be shown that if the mask generating function has not been queried on the input $\bar{X}||C^*$ then the probability that the key derivation function has been queried on the correct input is negligible and hence an attackers advantage is zero. Therefore we can conclude that if that attacker has a non-negligible advantage then there is a non-negligible probability that the mask generating function has been queried on the input $\bar{X^*}||C^*$ and so the probability that we output X^* is $\frac{1}{q_M}^{th}$ of that probability.