Proceedings of the World Congress on Engineering 2008 Vol I
WCE 2008, July 2 - 4, 2008, London, U.K.

A New KEM-Based Generic Construction

Neyire Deniz Sarier [*]

Abstract—**Recently, various hybrid encryption schemes have been proposed, which could be considered based on one-way or OW-PCA secure KEMs. However, OW-PCA secure KEMs require the security of the encryption scheme to be based on stronger assumptions (e.g. GAP assumptions). Hence, it is more advantagous to construct a generic conversion which can adapt to all type of KEMs resulting in an efficient and highly secure encryption scheme.**

In this paper, we will describe a new generic construction that converts a one-way secure KEM to an efficient IND-CCA secure PKE scheme. Having no restriction for the application field of the KEM, our construction yields to a wider class of hybrid encryption schemes. The resulting hybrid scheme is more efficient than the Fujisaki-Okamoto construction in terms of the number of the hash functions and the tightness of the security reduction. Besides, the ciphertext size is shorter than REACT. We also show an application of our construction on bilinear pairings.

Keywords: KEM, OW, OW-PCA, PKE, IND-CCA

1 Introduction

In order to encrypt arbitrary long messages, hybrid encryption schemes are devised which consist of two independent layers. The first layer is called as the public key layer where simply Public Key Encryption (PKE) is performed to generate a uniformly distributed shared key and the corresponding ciphertext for that key. This construction could be considered as a simpler form of an encryption scheme and is defined as Key Encapsulation Mechanism (KEM). The second layer uses the shared key to encrypt the actual message via symmetric key encryption and this operation performed on the symmetric layer is called as Data Encapsulation Mechanism (DEM). It is required that the key space of the KEM is the same as the key space of the DEM. This two-layered approach is proposed in the work of Shoup and it is referred in the literateur as KEM/DEM framework. In recent years, a number of generic constructions of KEMs have been introduced, some of which do not fit Shoup's framework. Generally, KEMs that are secure in a strong sense are obtained from standard PKE schemes, which only need to be weakly secure.

Apart from the advantages in message sizes and efficiency, KEM/DEM framework provides the highest security level for the hybrid encryption scheme, namely Indistinguishability under Chosen Ciphertext Attack (IND-CCA). Clearly, this could be achieved by combining an IND-CCA secure KEM with an IND-CCA secure DEM, however there are efficient IND-CCA secure hybrid encryption schemes, where the underlying KEM is not IND-CCA secure as in the case of Kurosawa-Desmedt hybrid scheme. Besides, Baek et al described two conversions from a weakly secure KEM to IND-CCA secure encryption scheme in the random oracle model. Here, the KEM is based on the security of One-Wayness against Plaintext Checking Attack (OW-PCA), which is a stronger condition than one-wayness, where the special plaintext checking oracle is not required. Hence, it is more advantagous to construct a generic conversion which can adapt to all type of KEMs resulting in an efficient and highly secure hybrid encryption scheme. In this point of view, we aim to introduce a new generic construction based on a one-way KEM and obtain an efficient and IND-CCA secure PKE scheme.

1.1 Related Work

The first KEM/DEM framework originated in the work of Shoup [10], who described different type of KEMs and DEMs, both satisfying the IND-CCA security. Kurosawa et al presented the notion of a Tag-KEM, which is the generalization of a KEM and developed the Tag-KEM/DEM framework [2]. The Tag-KEMs are obtained either from PKEs or from KEMs and in [2], hybrid encryption schemes based on Fujisaki-Okamoto conversion [7], Bellare-Rogaway Scheme [4] and REACT [8] are described. Besides, Baek et al defined two different transformations that convert any OW-PCA KEM into an PKE scheme secure in the sense of IND-CCA in random oracle model [3]. Their constructions are related to the schemes REACT and DHIES [1] and they prove that if the underlying KEM is not OW-PCA secure, REACT and DHIES are not IND-CCA secure. OW-PCA security of these hybrid schemes generally reduces to a GAP problem, which is a weaker problem than a computational problem since a GAP problem is solving a computational problem by accessing the corresponding decisional oracle. At last, Bentahar et al extend the concept of KEM to the setting of Identity Based and Certificateless Encryption [5].

[*]Submitted on 22 March 2008
Bonn-Aachen International Center for Information Technology
B-IT, Computer Security Group, Dahlmannstr. 2, D-53113
Bonn, Germany. Email: denizsarier@yahoo.com

1.2 Our Contribution

We will describe a new generic construction that converts a one-way secure KEM to an efficient IND-CCA secure PKE scheme. Hence, our scheme converts a very weak primitive to an asymmetric encryption scheme that is highly secure in the random oracle model. As a warm up, we will extend the Bellare Rogaway's generic construction, which applies to any trapdoor one-way permutation f. However, this first attempt results in a malleable scheme when the hash functions are replaced by a real function such as SHA. Hence, with a simple modification, we obtain an IND-CCA secure hybrid encryption scheme without increasing the ciphertext length and with a tighter security reduction than Fujisaki-Okamoto transformation. Our construction requires one less hash function than Fujisaki-Okamoto transformation and has a shorter ciphertext compared to a KEM-based encryption scheme that employs REACT. Finally, we require the KEM to be one-way secure thus, having no restriction for the application field of the KEM, our construction yields to a wider class of hybrid schemes.

We show also an application of our construction on the Sakai Kasahara Key Construction, which uses bilinear pairings.

1.3 Outline of the Paper

In section 2, we will state the definitions of KEM, DEM and PKE together with the security notions associated to them. Next, we present the new construction together with an application in section 3 and give the security proof. Finally, we conclude our proposals in section 4.

2 Definitions and Building Blocks

In order to introduce the new definitions and security notions, at first, we give some notations and conventions. Given a set S, $x \overset{R}{\leftarrow} S$ defines the assignment of a uniformly distributed random element from the set S to the variable x. S_r and M represents the random coin space and the message space. A function $\epsilon(k)$ is defined as negligible for any constant c and $k > k_0$, if there exists $k_0 \in N$ such that $\epsilon < (1/k)^c$.

2.1 Key Encapsulation Mechanism (KEM)

In Shoup's model, a KEM consists of three algorithms: Key generation, encryption and decryption algorithms. The formal definition of the KEM is as follows:

- Key Generation Algorithm $Keygen(l)$: A probabilistic algorithm that takes a security parameter $l \in \mathbb{N}$ and generates a public and a secret key pair (pk, sk).
- Key Encryption Algorithm $Enc_{pk}^{KEM}(r)$: A deterministic algorithm that takes as input the recipient's public key pk and a random string $r \in S_r$, outputs a pair (k, c), where $k \in S_k$ is a random session key and c is the encapsulation of k. The key space is defined as S_k, which is the key space of the DEM.
- Key Decryption Algorithm $Dec_{sk}^{KEM}(c)$: The decryption algorithm that recovers k from c.

For soundness, we require that for any (c, k) that are generated by $Enc_{pk}^{KEM}(r)$, $Dec_{sk}^{KEM}(c) = k$ must hold.

A one-way secure KEM requires that the adversary cannot recover the random session key k from the ciphertext c with non-negligible probability. The formal definition is as follows.

Definition 1 (One-Wayness (OW)):
Given a KEM $= (Keygen(l), Enc_{pk}^{KEM}(r), Dec_{sk}^{KEM}(c))$ and a probabilistic polynomial time adversary A in the following experiment:

```
Experiment OW($l$, KEM, A)
$(pk, sk) \leftarrow Keygen(l)$
$(k, c) \leftarrow Enc_{pk}^{KEM}(r)$
$k' \leftarrow A(pk, c)$
If $k' = k$ return 1
else return 0
```

The success of the attacker A with running time t in breaking the one-wayness of KEM is defined as

$$Succ_{A,KEM}^{OW} = \Pr[\mathrm{OW}(l, \mathrm{KEM}, \mathrm{A})=1]$$

Remark 2.1. *Baek et al defined the security notion OW-PCA for a KEM with a stronger requirement, which means that an attacker breaks the one-wayness of the KEM by accessing a Plaintext Checking (PC) oracle that decides on input of a pair of (k, c) whether c encrypts k. The definition of this security notion is given in Appendix. In our paper, the new generic construction does not require the KEM to be OW-PCA.*

2.2 Data Encapsulation Mechanism (DEM)

In hybrid encryption, a KEM is combined with a DEM, which is a symmetric encryption scheme. A DEM consists of two algorithms; $Enc(k, m)$ and $Dec(k, c)$, where $k \in S_k$ and $m \in M$, which are defined by the security parameter l of the KEM. In our generic construction we will implement the DEM by a one-time pad, where the DEM only needs to be IND-CPA secure and the resulting PKE scheme will be secure against IND-CCA.

2.3 Public Key Encryption (PKE)

A PKE scheme consists of three algorithms: Key generation, encryption and decryption algorithms that are defined as follows:

- Key Generation Algorithm $Keygen(l)$: A probabilistic algorithm that takes a security parameter $l \in \mathbb{N}$ and generates a public and a secret key pair (pk, sk).
- Encryption Algorithm $Encrypt_{pk}^{PKE}(m)$: A probabilistic encryption algorithm that takes as input the recipient's public key pk and a message $m \in M$ and encrypts the message into the ciphertext C.
- Decryption Algorithm $Decrypt_{sk}^{PKE}(C)$: The decryption algorithm that recovers m from C.

For soundness, $Decrypt_{sk}^{PKE}(C) = m$ must hold given any $((pk, sk), m)$ that are generated by the above functions.

For the PKE scheme we require the highest level of security, namely IND-CCA, where the adversary cannot extract one bit of information about the message from the ciphertext. In this security notion, the attacker has access to the decryption oracle O except for the query on the challenge. Basically, a decryption oracle is an oracle decrypting ciphertexts for an adversary. Clearly, PC oracle remains weaker than a decryption oracle because it is generally easier to check the solution of a problem than to compute it. The formal definition of this notion is as follows.

Definition 2 (IND-CCA):
Given a PKE scheme and a probabilistic polynomial time adversary A=(A_1, A_2) in the following experiment:

Experiment IND-CCA(l, PKE, A)
$(pk, sk) \leftarrow Keygen(l)$
$(m_0, m_1, s) \leftarrow A_1^O(pk)$ with $|m_0| = |m_1|$
$b \overset{R}{\leftarrow} 0, 1;$
$c \leftarrow Encrypt_{pk}^{PKE}(m_b)$
$b' \leftarrow A_2^O(C, s)$
If $b' = b$ return 1
else return 0

The advantage of the attacker A with running time t is defined as

$$Adv_{A,PKE}^{IND-CCA} = |Pr[b' = b] - \tfrac{1}{2}|$$

Hence, a PKE scheme is IND-CCA secure if the advantage of A is negligible in the security parameter l. Here, s is a state information.

3 A New Generic Construction

In this section, we present a new generic construction based on a one-way KEM and obtain an IND-CCA secure PKE scheme. For this purpose, we start by extending the Bellare Rogaway's generic construction for a KEM that we call as Conversion *P1*. Next, we will modify *P1* to overcome the disadvantages of *P1* and obtain an efficient and non-malleable hybrid encryption scheme.

Conversion *P1*: In [2], Kurosawa et al present a hybrid encryption scheme in the Tag-KEM/DEM framework by modifying the Bellare-Rogaway generic construction. However, their scheme does not change the encryption function of Bellare and Rogaway's scheme, which consists of a one-way permutation f and two random oracles H and G.

More specifically, a ciphertext C of Bellare and Rogaway's scheme has the following form:

$$C = (c_1, c_2, c_3) = (f(r), G(r) \oplus m, H(r||m))$$

In this scheme, when one-time pad is generalized to any one-time secure DEM and the tag $c_3 = H(r||m)$ is modified to $H(r||c_2)$, one obtains the construction in [2]. Here, the session key of the DEM is $G(r)$, which is computed by applying the secret key to c_1, namely f^{-1} and obtaining the random element r.

Our first generic construction $P1$ also uses two hash functions but the random session key and its ciphertext is computed differently. In our new construction $P1$, which takes as input any OW-PCA KEM and turns it to a IND-CCA secure encryption scheme in the random oracle model, we need two hash functions H_2 and H_3, where $H_2 : S_k \rightarrow \{0,1\}^n$ and $H_3 : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$. Here, n is the bit-length of the plaintext and k_1 is a security parameter. Specifically,

- Key Generation Algorithm $Keygen(l)$: A probabilistic algorithm that takes a security parameter $l \in \mathbb{N}$ and generates a public and a secret key pair (pk, sk).
- Encryption Algorithm $Encrypt_{pk}^{PKE}(m)$: This algorithm first runs the key encryption algorithm $Enc_{pk}^{KEM}(r)$ of the KEM and outputs the pair (c, k), where k is the random session key and c is the encapsulation of k. Next, the message m is encrypted using the DEM, which is one-time pad for our construction. The DEM outputs $c_2 = m \oplus H_2(k)$ and to provide non-malleability, an additional component of $c_3 = H_3(m, k)$ is used. Here, k is the shared key computed by the KEM. Finally, the ciphertext $C = (c_1, c_2, c_3) = (c, m \oplus H_2(k), H_3(m, k))$ is the output of the $Encrypt_{pk}^{PKE}(m)$.
- Decryption Algorithm $Decrypt_{sk}^{PKE}(C)$: The decryption algorithm recovers m from C first by applying the secret key sk to c_1 to obtain k'. Next, $H_2(k')$ is computed and xored with c_2. Finally, it is checked whether $H_3(m', k') = c_3$. If they are equal, m' is returned, otherwise C is rejected.

The difference between the first construction $P1$ and the construction in [2] is that, when the secret key is applied to c_1, namely to the encapsulation of the session key k, we do not fully invert the KEM and obtain the random element r as in [2], but only compute the random key k.

The construction *P1* is based on a OW-PCA secure KEM and can be proven IND-CCA secure in the random oracle model with a tight reduction cost. However, this could only be achieved for the encryption of fixed length messages since *P1* can be proven insecure for some instantiations of the hash function such as SHA, which is an iterative hash function. More specifically, if the hash function H_3 is instantiated with SHA as $H_3(m,k) = SHA(1||k||m)$ and the hash function $H_2(m)$ is instantiated as $H_2(k) = SHA(0||k)$, one can easily construct the encryption of $m||m'$ by using the ciphertext $(C = c_1, m \oplus SHA(0||k), SHA(1||k||m))$ that was obtained previously by the adversary.

3.1 Modified Generic Construction: $P2$

To overcome the message size limitation of the construction $P1$, we present a modification of $P1$, which we define as $P2$. This new construction takes as input any one-way KEM and results in an IND-CCA secure hybrid encryption scheme in the random oracle model. Since the security assumption on the KEM is changed to one-wayness, there is no need for a PC oracle resulting in a construction that is applicable for a wider class of primitives. The ciphertext size does not increase and the only additional operation is an xoring. Specifically,

- Key Generation Algorithm $Keygen(l)$: A probabilistic algorithm that takes a security parameter $l \in \mathbb{N}$ and generates a public and a secret key pair (pk, sk).
- Encryption Algorithm $Encrypt_{pk}^{PKE}(m)$: This algorithm first runs the probabilistic key encryption algorithm $Enc_{pk}^{KEM}(r)$ of the KEM and outputs the pair (c, k), where k is the random session key and c is the encapsulation of k. Next, the message m is encrypted as $c_2 = m \oplus H_2(k)$ and to provide non-malleability, an additional component of $c_3 = r \oplus H_3(m,k)$. Here, r is the random string that is used by the $Enc_{pk}^{KEM}(r)$ algorithm of the KEM and the range of H_3 is the group that the random element r is selected from. Namely, $r \in S_r$ and $H_3 : \{0,1\}^* \to S_r$. Finally, the ciphertext $C = (c_1, c_2, c_3) = (c, m \oplus H_2(k), r \oplus H_3(m,k))$ is the output of the $Encrypt_{pk}^{PKE}(m)$.
- Decryption Algorithm $Decrypt_{sk}^{PKE}(C)$: The decryption algorithm recovers m' from C first by applying the secret key sk to c_1 to obtain k'. Secondly, $H_2(k')$ is computed and xored with c_2. Next, $H_3(m', k')$ is computed and xored with c_3 to obtain the random element r'. Finally, $Enc_{pk}^{KEM}(r')$ of the KEM is run and compared to c_1. If they are equal, m' is returned, otherwise C is rejected.

Theorem 3.1. *Suppose there exists an adversary A which distinguishes $P2$ within a time bound τ_A and with advantage ϵ in less than q_{H_2}, q_{H_3} and q_D random oracle calls. Then there exists an algorithm R which inverts KEM with probability $\epsilon' \geq \frac{2\epsilon}{q_{H_2}+q_{H_3}} - \frac{q_D}{2^{k_1}}$ and in time $\tau_R \leq \tau_A + q_D q_{H_3}\sigma$, where σ is the time to compute $Enc_{pk}^{KEM}(r)$.*

Proof. Given the PKE scheme $P2 = (Keygen(l), Encrypt_{pk}^{PKE}(m), Decrypt_{sk}^{PKE}(C))$, the goal of the reduction algorithm R is to invert the KEM given (pk, c^*) and obtain k^*. During this procedure, the reduction will use the adversary A who tries to distinguish the encryption $C_\beta = (c^*, m_\beta \oplus h_2^*, h_3^*)$ given by the algorithm R where h_2^* and h_3^* are picked at random from the ranges of H_2 and H_3 by R. The proof is as follows:

1. The algorithm R is given pk as the public key and c^* as the challenge ciphertext of the KEM, who will recover the shared key k^* using the adversary A.
2. R returns A (pk, H_2, H_3) as the public parameters of $P2$ and simulates the challenger for A by answering the random oracle and decryption queries of A. Here H_2, H_3 are random oracles controlled by R.
3. H_2-queries: On each new input k, R picks a random $h_2 \in \{0,1\}^n$, returns that value to A and inserts the tuple (k, h_2) to the H_2List.
4. H_3-queries: On each new input (k, m), R picks a random $h_3 \in \{0,1\}^{k_1}$, returns that value to A and inserts the tuple (k, m, h_3) to the H_3List.
5. Decryption queries: On each new input (c_1, c_2, c_3),
   - R computes for each entry (k_i, m_i, h_{3i}) in the H_3List the value $r_i = h_{3i} \oplus c_3$.
   - For each r_i, R checks whether KEM$(r_i) = (c_1, k_i)$. If not, R returns reject.
   - Otherwise, R computes $H_2(k_i)$ using the simulation of H_2 as above and checks whether $m_i \oplus H_2(k_i) = c_2$. If not, R returns reject, else R returns m_i.
6. Challenge: Using the value c^*, R generates the challenge ciphertext for A as follows:
   - Upon receiving the equal length messages (m_0, m_1), R chooses at random $\beta \in \{0,1\}$.
   - R picks at random h_2^* and h_3^* from $\{0,1\}^n$ and $\{0,1\}^{k_1}$, where h_3^* represents $r^* \oplus H_3(m_\beta, k^*)$.
   - R generates the challenge ciphertext as $C = (c^*, m_\beta \oplus h_2^*, h_3^*)$.

7. R answers A's random oracle and decryption queries as before.

8. A outputs its guess β'. R will pick at random an entry from $[H_2\text{List} \vee H_3\text{List}]$ and returns the k to the challenger.

$\square$

3.2 Security Analysis

When we analyze the simulation of the random oracles, the answers to the random oracles queries are uniformly distributed. The simulation of the decryption oracle is nearly perfect but there are cases when a valid ciphertect $C = (c_1, c_2, c_3)$ is rejected if the corresponding tuple (m, k) has not been queried to the oracle H_3 since the computation of r depends on the tuples listed in H_3List. Consequently, the adversary needs either to guess a right value for the output of H_3 without querying H_3 and for r at the same time, in other words, he needs to guess a correct value for c_3. And the probability that A guesses a correct value for c_3 is only $\frac{1}{2^{|k_1|}}$. Here k_1 is a security parameter such that $q \geq 2^{k_1}$ and q is the order of the group S_r, which is the range of H_3.

Let H be the event that algorithm A issues a query for $H_2(k^*)$ or $H_3(m, k^*)$, namely k^* appears in some tuple on the $[H_2\text{List} \vee H_3\text{List}]$. Due to the above written facts, $Pr[H]$ is equal in both the real and simulated attacks, even in the case when A does not issue a query for k^*. The decryption of the challenge is independent of A's view since $H_2(k^*)$ or $H_3(m, k^*)$ are independent of A' s view.

Besides, $Pr[H] \geq 2\epsilon$ due to the following facts:

By the definition of $A \Rightarrow |Pr[\beta = \beta'] - \frac{1}{2}| \geq \epsilon$ (*)

If $k^* \notin [H_2\text{List} \vee H_3\text{List}] \Rightarrow Pr[\beta = \beta' | \neg H] = \frac{1}{2}$

$$\begin{aligned} Pr[\beta = \beta'] &= Pr[\beta = \beta' | H] Pr[H] + Pr[\beta = \beta' | \neg H] Pr[\neg H] \\ &\leq Pr[H] + Pr[\beta = \beta' | \neg H] Pr[\neg H] \\ &= Pr[H] + \tfrac{1}{2} Pr[\neg H] = \tfrac{1}{2} + \tfrac{1}{2} Pr[H] \end{aligned}$$

$$Pr[\beta = \beta'] \geq Pr[\beta = \beta' | \neg H] Pr[\neg H] = \tfrac{1}{2} - \tfrac{1}{2} Pr[H]$$

$$\begin{aligned} &\tfrac{1}{2} - \tfrac{1}{2} Pr[H] \leq Pr[\beta = \beta'] \leq \tfrac{1}{2} + \tfrac{1}{2} Pr[H] \\ &\Rightarrow -\tfrac{1}{2} Pr[H] \leq Pr[\beta = \beta'] - \tfrac{1}{2} \leq \tfrac{1}{2} Pr[H] \\ &\Rightarrow |Pr[\beta = \beta'] - \tfrac{1}{2}| \leq \tfrac{1}{2} Pr[H] \qquad (**) \end{aligned}$$

Combining (*) and (**), we obtain

$$\epsilon \leq |Pr[\beta = \beta'] - \tfrac{1}{2}| \leq \tfrac{1}{2} Pr[H] \Rightarrow Pr[H] \geq 2\epsilon$$

This result shows that the correct session key k^* takes place in the union list with probability at least 2ϵ. Since at the end of the simulation, a random tuple from the union list is selected, the probability that this tuple is the correct answer is at least $\frac{2\epsilon}{q_{H_2} + q_{H_3}}$ due to the $q_{H_2} + q_{H_3}$ total entries in the union list. When the total number of hash queries are taken as equal $q_{H_2} = q_{H_3} = q_H$, the correct session key k^* is in the union list with probability at least $\frac{\epsilon}{q_H}$.

Claim: The view of A in the simulation is identical to the view in the real attack, if A does not correctly guess the value of c_3. One can define the event $Guessc_3$ as A's correctly guessing the value of c_3, which occurs with probability $\frac{q_D}{2^{|k_1|}}$. This event results in a linear term that is subtracted from the final result. Hence,

$$Pr[SuccessR] > \tfrac{\epsilon}{q_H} - \tfrac{q_D}{2^{k_1}}.$$

When the reduction cost of the schemes $P2$ and $P1$ are compared, there is a loss of $\frac{1}{q_H}$ in $P2$ due to the removal of the PC oracle. However, breaking OW-PCA security means breaking its one-wayness by accessing an oracle that solves an easier problem, which corresponds to solving a GAP problem. Solving GAP problems could be easier than solving a computational problem or sometimes there might be no groups on which one can define the GAP problem. Consequently, $P2$ generates a flexible and efficient hybrid encryption scheme that is based on a one-way KEM and allows encryption of arbitrary size messages as opposed to P_1.

3.3 Application on Bilinear Pairings

An application of our generic construction $P2$ in identity based setting is presented using the Sakai Kasahara Key Construction which is based on bilinear pairings. In [6], Chen and Cheng present an efficient IBE scheme using this key construction and Fujisaki-Okamoto transformation. Their scheme SK-IBE uses four hash functions and security proof requires reductions to intermediate schemes with tightness of $O(\frac{\epsilon}{q_H^3})$. However, when our new construction P_2 is applied instead of the Fujisaki-Okamoto transformation, three hash functions is used and the same computational problem can directly be reduced to the scheme with the tightness $O(\frac{\epsilon}{q_H^2})$. The new scheme SK-IBE2 is presented as follows.

- Setup: Given a security parameter k_1, the parameters of the scheme is generated as follows.
  1. Generate two cyclic groups G and F of prime order q and a bilinear pairing $\hat{e} : G_1 \times G_2 \rightarrow F$. Pick a random generator $P_2 \in G_2^*$ and set $P_1 = \psi(P_2)$, where ψ is an isomorphism form G_2 to G_1.
  2. Pick a random $x \in \mathbb{Z}_q^*$ and compute xP_1.
  3. Pick two cryptographic hash functions $H_2 : F \rightarrow \{0,1\}^n$ and $H_3 : \{0,1\}^n \times F \rightarrow \mathbb{Z}_q^*$

The message space is $M = \{0,1\}^n$. The ciphertext space is $C = G_1^*$ x $\{0,1\}^n$ x $\mathbb{Z}_q^*$. The master public key is $(q, G_1, G_2, F, \psi, \hat{e}, n, P_1, P_2, xP_1, H_1, H_2, H_3)$ and the master secret key is x.

- Extract: Given an identity $ID_A \in \{0,1\}^*$, the master public key and the master secret key, the algorithm returns $d_A = \frac{1}{x+H_1(ID_A)}P_2$.
- Encrypt: Given a plaintext $m \in M$ and an identity ID_A, the following steps are performed.
  1. Select a random element $r \in \mathbb{Z}_q^*$.
  2. Compute $L_A = H_1(ID_A)P_1 + xP_1$ and $k = \hat{e}(P_1, P_2)^r$.
  3. Set the ciphertext to $C = (U, V, W) = (rL_A, m \oplus H_2(k), r \oplus H_3(m, k))$.
- Decrypt: Given $C = (U, V, W)$ and d_A,
  1. Compute $k' = \hat{e}(U, d_A)$ and $m' = V \oplus H_2(k')$
  2. Compute $r' = W \oplus H_3(k', m')$
  3. If $r'L_A \neq U$, return reject, else return m' as the plaintext.

Theorem 3.2. *Suppose the hash functions H_1, H_2, H_3 are random oracles and there exists an IND-ID-CCA adversary A that runs in time at most t and has advantage ε against the scheme SK-IBE2 making at most $q_E > 0$ private key extraction queries, $q_D > 0$ decryption queries and $q_{H_1}, q_{H_2}, q_{H_3} > 0$ hash queries to H_1, H_2, H_3. Then there is an algorithm B that solves the k-BDHI problem where $k = q_{H_1}$ with advantage at least*

$$\frac{\varepsilon}{q_{H_1}(q_{H_2}+q_{H_3})} - \frac{q_D}{2^{k_1}}$$

The security proof is presented in Appendix and uses the same technique as in [9].

4 Conclusion

In this paper, we proposed a new generic construction P_2 that is based on a weakly secure KEM and result in IND-CCA secure hybrid encryption scheme. When compared to hybrid encryption schemes based on OW-PCA KEMs such as REACT, our construction achieves shorter ciphertext size than REACT scheme that is employed as in [3]. Besides, when compared to the Fujisaki-Okamoto transformation, our construction is more efficient in terms of the number of hash functions and the tightness of the reduction cost. A summarizing table is shown as below, where |c| is the size of the key encapsulation and |h| is the size of the output of the hash functions. The only disadvantage of our construction P_2 compared to REACT is the additional factor $1/q_H$ at the reduction cost, where q_H is the total number of H-queries. However, OW-PCA assumption on the KEM requires solving a GAP problem, whereas our constructions can be reduced to harder computational problems due to the removal of the PC oracle.

Table 1: Comparison of KEM-Based Constructions in Public Key Setting

Scheme	Assumption	Ciphertext size	Tightness
REACT*	OW-PCA	|c|+3 |h|	tight
P_1	OW-PCA	|c|+2 |h|	tight
Fujisaki-Okamoto	OW	|c|+2 |h|	$O(\epsilon/q_H^2)$
P_2	OW	|c|+2 |h|	$O(\epsilon/q_H)$

* REACT implemented as in [3]

Acknowledgement

The author is grateful to Prof. Dr. Joachim von zur Gathen and Dr. Damien Vergnaud for their supervision of her Master's Thesis. Also, the author is thankful to Dr. Pascal Paillier and Laila El Aimani for the fruitful discussions during crypt@b-it 2007.

References

[1] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2001.

[2] Masayuki Abe, Rosario Gennaro, and Kaoru Kurosawa. Tag-KEM/DEM: A New Framework for Hybrid Encryption. *J. Cryptology*, 21(1):97–130, 2008.

[3] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Efficient Multi-receiver Identity-Based Encryption and Its Application to Broadcast Encryption. In Serge Vaudenay, editor, *Public Key Cryptography - PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 380–397. Springer, 2005.

[4] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[5] Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel P. Smart. Generic Constructions of Identity-Based and Certificateless KEMs. *J. Cryptology*, 21(2):178–199, 2008.

[6] Liqun Chen and Zhaohui Cheng. Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme. In Nigel P. Smart, editor, *Cryptography and Coding, IMA Int. Conf.*, volume 3796 of

Lecture Notes in Computer Science, pages 442–459. Springer, 2005.

[7] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[8] Tatsuaki Okamoto and David Pointcheval. REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 159–175. Springer, 2001.

[9] Neyire Deniz Sarier. Identity Based Encryption: Security Notions and New IBE Schemes Based on Sakai Kasahara's Key Construction. Master's thesis, RWTH Aachen, 2007.

[10] Victor Shoup. Using Hash Functions as a Hedge against Chosen Ciphertext Attack. In *Advances in Cryptology - EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 275–288. Springer, 2000.

Appendix

In this appendix, we review the security notion of OW-PCA and give the security proof of the theorem 3.2.

OW-PCA Notion for a KEM

The security notion of OW-PCA is defined for encryption schemes in [9] and its analogous for KEMs is defined in [11]. We first begin with the definition of a PC oracle.

Definition 3 (Plaintext Checking (PC) Oracle):
Given a KEM $= (Keygen(l), Enc(pk, r), Dec(sk, c))$, the PC oracle decides on input a random session key $k \in S_k$ and an encapsulation c of that key, whether c encrypts k. Obviously in the case of a deterministic scheme, PC oracle is useless since the decision could be made by reencryption.

Definition 4 (OW-PCA):
Given a KEM $= (Keygen(l), Enc(pk, r), Dec(sk, c))$ and a probabilistic polynomial time adversary A in the following experiment:

```
Experiment OW-PCA($l$, KEM, A)
$(pk, sk) \leftarrow Keygen(l)$
$r \overset{R}{\leftarrow} S_r$; $(k, c) \leftarrow Enc(pk, r)$
$k' \leftarrow A^{PC}(pk, c)$
If $k' = k$ return 1
else return 0
```

The success of the attacker A in breaking the OW-PCA of KEM is defined as follows. Here, A has the running time t and makes q_{PC} queries to the PC oracle.

$$Succ_{A,KEM}^{OW-PCA} = \Pr[\text{OW-PCA}(l, \text{KEM, A})=1]$$

Proof of Theorem 3.2

Algorithm B is given as input a random k-BDHI instance $(P_1, P_2, xP_2, x^2P_2, ..., x^kP_2)$, which computes $\hat{e}(P_1, P_2)^{1/x}$ using the attacker A as follows.

- B chooses an index I with $1 \leq I \leq q_{H_1}$ as its guess for the challenge identity that A attacks.
Here $h_0 \in \mathbb{Z}_q^*$ represents the $H_1(ID_I)$.
- B prepares the public parameters for A as follows [6]:
  1. B randomly chooses different $h_1, ..., h_{k-1} \in \mathbb{Z}_q^*$.
  2. B sets $f(z) = \prod_{i=1}^{k-1}(z + h_i)$. One can easily reformulate $f(z)$ to the form $f(z) = \sum_{i=0}^{k-1} c_i z^i$. The constant term c_0 is non-zero because $h_i \neq 0$ and different and c_i are computable from h_i.
  3. B computes $Q_2 = \sum_{i=0}^{k-1} c_i x^i P_2 = f(x)P_2$ and $xQ_2 = xf(x)P_2 = \sum_{i=0}^{k-1} c_i x^{i+1} P_2$.
  4. B computes $f_i(z) = \frac{f(z)}{z+h_i} = \sum_{j=0}^{k-2} d_j z^j$ and $\frac{1}{x+h_i}Q_2 = \frac{1}{x+h_i}f(x)P_2 = f_i(x)P_2 = \sum_{j=0}^{k-2} d_j x^j P_2$ for $i \in [1, k-1]$.
  5. B computes $Q_1 = \psi(Q_2)$.
  6. B sets $T' = \sum_{i=1}^{k-1} c_i x^{i-1} P_2$ and computes $T_0 = \hat{e}(\psi(T'), Q_2 + c_0 P_2)$.
- B passes the public parameters to A as $M_{pk} = (q, G_1, G_2, F, \psi, \hat{e}, n, Q_1, Q_2, xQ_1 - h_0 Q_1)$. Here, H_1,H_2 and H_3 are random oracles controlled by B. By setting $P_{pub} = xQ_1 - h_0 Q_1$ the private key for the challenge identity is $d_{ch} = \frac{1}{x}Q_2$ since $\hat{e}(h_0 Q_1 + P_{pub}), \frac{1}{x}Q_2) = \hat{e}(Q_1, Q_2)$. Also note that $\hat{e}((h_i + h_0)Q_1 + P_{pub}), \frac{1}{h_i+x}Q_2) = \hat{e}(Q_1, Q_2)$ for $i = 1, ..., k-1$. Hence M_{pk} represents valid public parameters for SK-IBE2.
- H_1-queries: Upon receiving a query ID_j to the random oracle H_1 for some $j \in [1, q_{H_1}]$, B checks if $(ID_j, h_j + h_0, \frac{1}{h_j+x}Q_2)$ exists in H_1List. If the entry exists in the list, B returns $h_j + h_0$. Otherwise, B does the following:

1. If the query is on the $I-th$ distinct ID, namely $ID_j = ID_I$, B stores $(ID_I, h_0, \perp)$ in H_1List and returns h_0.
2. Else, B inserts the tuple $(ID_j, h_j + h_0, \frac{1}{h_j+x}Q_2)$ to the H_1List and returns $h_j + h_0$ to A.

- H_2-queries: On each new input k_i, B chooses a random $\xi_i \in \{0,1\}^n$, inserts (k_i, ξ_i) to the H_2List and returns ξ_i to A.

- H_3-queries: On each new query (m_i, k_i) to the H_3 oracle, B chooses a random $h_{3i} \in \mathbb{Z}_q^*$, inserts the tuple (m_i, k_i, h_{3i}) to H_3List and returns h_{3i} to A.

- Extraction-query: Upon receiving a private key extraction query ID_i for some $i \in [1, q_E]$, B first checks H_1List. If ID_i does not exist in the list, B queries $H_1(ID_i)$ and checks the value d_i, which is the last entry of the corresponding tuple. If $d_i \neq \perp$, B returns $d_i = \frac{1}{h_i+x}Q_2$. Else, B aborts the game.

- Decryption query: Upon receiving a decryption query (C_i, ID_I) for some $i \in [1, q_D]$, where $C_i = (U_i, V_i, W_i) = (r_i x Q_1, m_i \oplus H_2(k_i), r_i \oplus H_3(m_i, k_i))$, B first looks to the H_1List. If ID_i is not on the list, then B queries $H_1(ID_i)$. If $d_i = \perp$, namely $ID_i = ID_I$, B does the following:

  1. B computes for each entry (k_i, m_i, h_{3i}) in the H_3List the value $r_i = h_{3i} \oplus c_3$.
  2. For each r_i, B checks whether $(r_i x Q_1) = U_i$ and $\hat{e}(Q_1, Q_2)^{r_i} = k_i$. If not, B returns reject.
  3. Else, B computes $H_2(k_i)$ using the simulation of H_2 as above and checks whether $m_i \oplus H_2(k_i) = V_i$. If not, B returns reject, else, B outputs m_i.

  Here we are rejecting basically a decryption query whose corresponding (m_i, k_i) has not been queried to the random oracle H_3.

  If $d_i \neq \perp$, namely $ID_i \neq ID_I$, B tries to perform the decryption by first computing $\hat{e}(U_i, d_i)$ and then querying $H_2(\hat{e}(U_i, d_i))$. A decryption query in the form of (C_i, ID_i) can easily be answered since the decryption key for ID_i is known to the adversary B.

- Challenge: Upon receiving the two equal length messages m_0, m_1 and the challenge identity ID_{ch}, B generates the challenge ciphertext as follows:

  1. If the I-th query on H_1 has been issued and $ID_I = ID_{ch}$ and so $d_{ch} = \perp$ B continues, else B aborts the game.
  2. Else if the tuple corresponding to ID_{ch} is on the H_1List and so $d_{ch} \neq \perp$, B aborts the game. Otherwise, B inserts the tuple $(ID_{ch}, h_0, \perp)$ into the list and continues. Note that, at this stage, $H_1(ID_{ch}) = h_0$ and $d_{ch} = \perp$.
  3. B chooses $\beta \in \{0,1\}$ and $r \in \mathbb{Z}_q^*$ at random and return rQ_1 as U^*.
  4. B chooses $h_2^* \in \{0,1\}^n$ at random, which is supposed to represent $H_2(k^*)$
  5. B chooses $W^* \in \mathbb{Z}_q^*$
  6. B returns $C^* = (U^*, m_\beta \oplus h_2^*, W^*)$.

- B answers A's random oracle queries, private key extraction and decryption queries as before. Note that following the rules, the adversary will not issue the extraction query on ID_{ch} only whose $d_{ch} = \perp$ and the decryption query on (ID_{ch}, C_{ch}). And thus, B always can answer other queries without aborting the game.

- A outputs its guess β'. B picks a random entry k_i from the [H_2List $\vee H_3$List], computes $T = k_i^{1/r}$ and returns $(T/T_0)^{1/c_0^2}$. It is clear that $\hat{e}(P_1, P_2)^{1/x} = (T/T_0)^{1/c_0^2}$ if $T = \hat{e}(Q_1, Q_2)^{1/x}$.

Security Analysis

The simulation of the decryption oracle is nearly perfect but there are cases when a valid ciphertect $C = (c_1, c_2, c_3)$ is rejected if the corresponding tuple (m, k) has not been queried to the oracle H_3. Consequently, the adversary needs either to guess a right value for the output of H_3 without querying H_3 and the correct value for r at the same time, or guess a correct value for c_3, which occur with probabilities of $\frac{1}{2^{k_1}}$ as previously.

Claim: The view of A in the simulation is identical to the view in the real attack, if A does not correctly guess the value of c_3. One can define the event $Guessc_3$ as A's correctly guessing the value of c_3 and the event H as the event that A queries the oracles H_2 or H_3 for $k = \hat{e}(Q_1, Q_2)^{r/x}$. Then, we have

$Pr[SuccessB] = Pr[H] = |Pr[\beta' = \beta | \neg Guessc_3] - \frac{1}{2}|$. Also by the definition of A, we have $|Pr[\beta' = \beta] - \frac{1}{2}| > \epsilon$.

$$\begin{aligned} Pr[SuccessB] &= |Pr[\beta' = \beta | \neg Guessc_3] - \tfrac{1}{2}| \\ &> |Pr[\beta' = \beta] - Pr[Guessc_3] - \tfrac{1}{2}| \\ &> \tfrac{\epsilon}{q_{H_1}(q_{H_2}+q_{H_3})} - Pr[Guessc_3] \end{aligned}$$

Due to the total decryption queries A makes, $Pr[Guessc_3] \leq \frac{q_D}{2^{k_1}}$. Thus, for $q_{H_1} = q_{H_2} = q_{H_3} = q_H$

$$Pr[SuccessB] > \frac{\epsilon}{q_{H_1}(q_{H_2}+q_{H_3})} - \frac{q_D}{2^{k_1}} \approx \frac{\epsilon}{q_H^2}.$$

An improvement on the efficiency of the SK-IBE2 could be the obtained by dropping the H_2 function and directly multiplying the message m to the session key $k = \hat{e}(P_1, P_2)^r$. Then the messages are selected from the group F.