A Universally Composable Secure Channel Based on the KEM-DEM Framework

Waka Nagao[1] Yoshifumi Manabe[1,2] Tatsuaki Okamoto[1,2]

[1] Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Kyoto, 606-8501 Japan
[2] NTT Labs, Nippon Telegraph and Telephone Corporation
1-1 Hikari-no-oka Yokosuka, 239-0847 Japan

Abstract. For ISO standards on public-key encryption, Shoup introduced the framework of KEM (Key Encapsulation Mechanism), and DEM (Data Encapsulation Mechanism), for formalizing and realizing *one-directional* hybrid encryption; KEM is a formalization of asymmetric encryption specified for key distribution, and DEM is a formalization of symmetric encryption. This paper investigates a more general hybrid protocol, *secure channel*, using KEM and DEM, such that KEM is used for distribution of a session key and DEM, along with the session key, is used for multiple *bi-directional* encrypted transactions in a session. This paper shows that KEM semantically secure against adaptively chosen ciphertext attacks (IND-CCA2) and DEM semantically secure against adaptively chosen plaintext/ciphertext attacks (IND-P2-C2) along with secure signatures and ideal certification authority are sufficient to realize a *universally composable* (UC) secure channel. To obtain the main result, this paper also shows several equivalence results: UC KEM, IND-CCA2 KEM and NM-CCA2 (non-malleable against CCA2) KEM are equivalent, and UC DEM, IND-P2-C2 DEM and NM-P2-C2 DEM are equivalent.

1 Introduction

1.1 Background

Key Encapsulation Mechanism (KEM) is a key distribution mechanism in public-key cryptosystems, that was proposed by Shoup for ISO standards on public-key encryption [11].

The difference between KEM and public-key encryption (PKE) is as follows: PKE's encryption procedure, on input plaintext M and receiver R's public-key PK_R, outputs ciphertext C, while KEM's encryption procedure, on input receiver R's public-key PK_R, outputs ciphertext C and key K, where C is sent to R, and K is kept secret inside the sender, and employed in the subsequent process of data encryption. PKE's decryption procedure, on input C and secret-key SK_R, outputs plaintext M, while KEM's decryption procedure, on input C and secret-key SK_R, outputs key K. Although KEM is a mechanism for key distribution and the applications of KEM are not specified, the most typical application is hybrid encryption, where a key shared via a KEM is employed for symmetric-key encryption. Shoup also formulated the symmetric-key encryption as the Data Encapsulation Mechanism (DEM)[11].

Shoup defined the security, "indistinguishable (semantically secure) against adaptively chosen-ciphertext attacks," for KEM and DEM, respectively, (we call them IND-CCA2-KEM and IND-CCA2-DEM, respectively), and showed that hybrid encryption (HPKE) implemented by combining KEM with IND-CCA2-KEM and DEM with IND-CCA2-DEM is a PKE with IND-CCA2-PKE [8, 11]. [3]

Since the KEM-DEM hybrid encryption specified by Shoup is *one-directional* (or equivalent to public-key encryption in functionality), it is applicable for secure email and single direction transactions. However, in many secure protocols (e.g., SSL, IPSec, SSH), asymmetric and symmetric encryption schemes are employed in a different manner as a *secure channel* such that an asymmetric encryption scheme is used for distribution of a session key while a symmetric encryption scheme with the session key is used for many bi-directional encrypted transactions in a session.

The KEM-DEM framework can be modified for such a hybrid usage, secure channel; KEM can be used for key distribution of a session key and DEM with the session key is used for secure communications in a session. Since the KEM-DEM framework will be standardized in a near future, it is a promising way to employ the above-mentioned modified KEM-DEM framework to realize a secure channel. However, no research has been done on the security requirements of KEM and DEM such that a secure channel based on the modified KEM-DEM framework can guarantee a sufficient level of security, although KEM with IND-CCA2-KEM and DEM with IND-CCA2-DEM have been shown to be sufficient for an IND-CCA2-PKE single-directional KEM-DEM-hybrid scheme [8, 11]. That is, we have the following problems:

- What are the security requirements of KEM and DEM to construct a secure channel?
- How to define the satisfactory level of security of a secure channel? (since it cannot be characterized by just public-key encryption, but should require more complicated security definition.)

1.2 Our Results

This paper answers the above-mentioned problems:

- This paper shows that KEM with IND-CCA2-KEM and DEM with IND-P2-C2-DEM along with secure signatures and ideal certification authority are sufficient to realize a universally composable secure channel.
- We follow the definition of a *universally composable* secure channel by Canetti and Krawczyk [6]. There are two major merits in using the universal composability paradigm. Firstly, the paradigm provides a clear and unified (or standard) approach to defining the security of any cryptographic functionality including a *secure channel*. Second, our concrete construction of a secure channel based on the KEM-DEM

[3] Originally, the notion of IND-CCA2 was defined for PKE. The way to provide analogous definitions and to use the same name, "indistinguishable (semantically secure) against adaptively chosen-ciphertext attacks", for KEM and DEM follows that of [8]. In this paper, however, we explicitly distinguish them by the terms, IND-CCA2-PKE, IND-CCA2-KEM, and IND-CCA2-DEM.

framework guarantees not only stand-alone security but also universal composable security. Since a secure protocol like SSL, IPSec and SSH is often employed as an element of a large-scale security system, the universal composability of a secure protocol is especially important.

In order to obtain the above-mentioned main result, we firstly show that UC KEM, IND-CCA2 KEM and NM-CCA2 KEM are equivalent, and that UC DEM, IND-P2-C2 DEM and NM-P2-C2 DEM are equivalent. We then present that UC KEM and UC KEM as well as UC signatures and ideal certification authority are sufficient for realizing a UC secure channel.

Although in this paper we consider only protocols for a single session, the same result for the multi-session case is obtained automatically via the UC with joint state (JUC) [7].

1.3 Related Works

Canetti and Krawczyk [6] showed a UC secure channel protocol consisting of an authenticated Diffie-Hellman key exchange scheme, message authentication code, and pseudorandom generator. Accordingly, their results are specific to their construction, which uses an authenticated Diffie-Hellman key exchange scheme, message authentication code and pseudorandom generator. Our result is based on the general notions of KEM, DEM and signatures, but not on any specific scheme.

The equivalence of UC PKE and IND-CCA2 PKE has been suggested by Canetti [3], and the equivalence of NM-CCA2 PKE and IND-CCA2 PKE has been shown by Bellare et.al. [1, 2]. The relationship among several security notions of symmetric encryptions has been investigated by Katz and Yung [10]. However, no results have been reported on the equivalence among UC KEM, IND-CCA2 KEM and NM-CCA2 KEM, and that among UC DEM, IND-CCA2 DEM and NM-CCA2 DEM.

2 The KEM-DEM Framework

We describe probabilistic algorithms and experiments with standard notations and conventions. For probabilistic algorithm A, $A(x_1, x_2, \cdots; r)$ is the result of running A that takes as inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots)$ denote the experiment of picking r at random and letting y equal the output of $A(x_1, x_2, \cdots; r)$. If S is a finite set, then $x \leftarrow S$ denotes the experiment of assigning to x an element uniformly chosen from S. If α is neither an algorithm nor a set, then $x \leftarrow \alpha$ indicates that we assign α to x. We say that y can be output by $A(x_1, x_2, \cdots)$ if there is some r such that $A(x_1, x_2, \cdots; r) = y$.

2.1 Key Encapsulation Mechanism

Formally, a key encapsulation mechanism KEM is given by the triple of algorithms $KEM.KeyGen()$, $KEM.Encrypt(pk, options)$ and $KEM.Decrypt(sk, C_0)$, where:

1. $KEM.KeyGen()$, the key generation algorithm, is a polynomial time and probabilistic algorithm that takes a security parameter $k \in N$ (provided in unary) and returns a pair (pk, sk) of matching public and secret keys.
2. $KEM.Encrypt(pk, options)$, the encryption algorithm, is a polynomial time and probabilistic algorithm that takes as input a public key pk, along with an optional $options$ argument, and outputs a key/ciphertext pair (K, C_0). The role of $options$ is analogous to that in public-key encryption.
3. $KEM.Decrypt(sk, C_0)$, the decryption algorithm, is a polynomial time and deterministic algorithm that takes as input secret key sk and ciphertext C_0, and outputs key K or special symbol $\perp$ ($\perp$ implies that the ciphertext was invalid).

We require that for all (pk, sk) output by $KEM.KeyGen(1^k)$, and for all C_0 output by $KEM.Encrypt(pk, options)$, $KEM.Decrypt(sk, C_0) = K$ ($|K|$ is denoted $KEM.OutputKeyLen$ — the length of the key output by $KEM.Encrypt$ and $KEM.Decrypt$). A function $\epsilon : N \rightarrow R$ is negligible if for every constant $c \geq 0$ there exists an integer k_c such that $\epsilon(k) \leq k^{-c}$ for all $z \geq k_c$. We write vectors in boldface, as in $\boldsymbol{x}$. We also denote the number of components in $\boldsymbol{x}$ by $|\boldsymbol{x}|$, and the i-th component by $\boldsymbol{x}[i]$, so that $\boldsymbol{x} = (\boldsymbol{x}[1], \cdots, \boldsymbol{x}[|\boldsymbol{x}|])$. Additionally, we denote a component of a vector as x $\in \boldsymbol{x}$ or x $\notin \boldsymbol{x}$, which mean, respectively, mean that x is in or is not in the set $\{ \boldsymbol{x}[i] : 1 \leq i \leq |\boldsymbol{x}|\}$. Such notions provide convenient descriptions. For example, we can simply write $\boldsymbol{x} \leftarrow KEM.Decrypt(\boldsymbol{y})$ as the shorthand form of $1 \leq i \leq |\ \boldsymbol{y}\ |$ do $\boldsymbol{x}[i] \leftarrow KEM.Decrypt(\boldsymbol{y}[i])$. We will consider relations of amity t where t is polynomial in the security parameter k. Rather than writing $R(x_1, \cdots, x_t)$ we write $R(x, \boldsymbol{x})$, meaning the first argument is special and the rest are bunched into vector $\boldsymbol{x}$ with $|\boldsymbol{x}| = t - 1$.

Attack types of KEM We state following three attack types of KEM. First, we state CPA (Chosen Plaintext Attack). CPA is an attack type that an adversary is allowed to access to only encryption oracle but not decryption oracle. Secondly, we state CCA1 (Chosen Ciphertext Attack). CCA1 is an attack type that an adversary is allowed to access to both encryption and decryption oracle. However the adversary cannot access to decryption oracle after getting target ciphertext. Thirdly, we state CCA2 (Adaptive Chosen Ciphertext Attack). CCA2 is an attack type that an adversary is allowed to access to both encryption and decryption oracle even if after the adversary gets target ciphertext.

Indistinguishability of KEM We use IND-ATK-KEM to describe the security notion of indistinguishability for KEM against ATK $\in$ {CPA, CCA1, CCA2}[11]. We redescribe the security notion of IND-CCA2-KEM by considering following attack scenario. First, the key generation algorithm is run to generate the public and private key for the protocol. The adversary can get the public key, but not the private key. Secondly, the adversary generates some queries of plaintexts/ciphertexts and sends the queries to encryption/decryption oracle. Each oracle encrypts/decrypts the queries and returns the results of ciphertexts/plaintexts to the adversary. If the algorithm fails, this information is informed to the adversary, and the attack continues. Thirdly, encryption oracle does the following:

1. Runs the encryption algorithm, generating pair (K^*, C_0^*).
2. Generates a random string $\widetilde{K}$ of length $KEM.OutputKeyLen$.
3. Chooses $b \in \{0,1\}$ at random.
4. If $b = 0$, outputs (K^*, C_0^*), otherwise outputs ($\widetilde{K}$, C_0^*).

Fourth, the adversary generates plaintexts/ciphertexts to get information from each oracle on the condition of the ciphertext $C_0 \neq C_0^*$. Finally, the adversary outputs $\hat{b} \in \{0,1\}$.

Let Π_{KEM} = ($KEM.KeyGen$, $KEM.Encrypt$, $KEM.Decrypt$) be an encryption protocol and let A be an adversary. The advantage of Π_{KEM} for adversary A, $Adv_{A,\Pi_{\mathrm{KEM}}}^{\text{IND-ATK}}$ is defined as follows:

$$Adv_{A,\Pi_{\mathrm{KEM}}}^{\text{IND-ATK}}(k) = |\Pr[\hat{b} = b] - \tfrac{1}{2}|.$$

Π_{KEM} is secure in the sense of IND-ATK if $Adv_{A,\Pi_{\mathrm{KEM}}}^{\text{IND-ATK}}(k)$ is negligible for any PPT adversary A.

Non-Malleability of KEM We state formal definition of non-malleability for KEM in Fig.1 following [1], which we call NM-KEM. We also use NM-ATK-KEM to describe the security notion of non-malleability for KEM against ATK $\in$ {CPA, CCA1, CCA2}. Let $A = (A_1, A_2)$ be an adversary. (We state two more definitions in the full paper version.)

Fig. 1. NM-KEM Definition

$$Adv_{A,\Pi_{\mathrm{KEM}}}^{\text{NM-ATK}}(k) \equiv \Pr[Expt_{A,\Pi_{\mathrm{KEM}}}^{\text{NM-ATK}}(k) = 1] - \Pr[\widetilde{Expt}_{A,\Pi_{\mathrm{KEM}}}^{\text{NM-ATK}}(k) = 1]$$

where

$Expt_{A,\Pi_{\mathrm{KEM}}}^{\text{NM-ATK}}(k)$	$\widetilde{Expt}_{A,\Pi_{\mathrm{KEM}}}^{\text{NM-ATK}}(k)$
$(pk, sk) \leftarrow KEM.KeyGen(1^k)$	$(pk, sk) \leftarrow KEM.KeyGen(1^k)$
$(\mathcal{K}, s) \leftarrow A_1^{O_1}(pk)$	$(\mathcal{K}, s) \leftarrow A_1^{O_1}(pk)$
$(K^*, C_0^*) \leftarrow KEM.Encrypt(pk) \wedge K^* \in \mathcal{K}$	$K^* \leftarrow \mathcal{K}$
$(R, \boldsymbol{C_0}) \leftarrow A_2^{O_2}(s, C_0^*)$	$(\widetilde{K}, \widetilde{C_0}) \leftarrow KEM.Encrypt(pk) \wedge \widetilde{K} \in \mathcal{K}$
$\boldsymbol{K} \leftarrow KEM.Decrypt(sk, \boldsymbol{C_0})$	$(R, \widetilde{\boldsymbol{C_0}}) \leftarrow A_2^{O_2}(s, \widetilde{C_0})$
return 1 iff $(C_0^* \notin \boldsymbol{C_0}) \wedge R(K^*, \boldsymbol{K})$	$\widetilde{\boldsymbol{K}} \leftarrow KEM.Decrypt(sk, \widetilde{\boldsymbol{C_0}})$
	return 1 iff $(\widetilde{C_0} \notin \widetilde{\boldsymbol{C_0}}) \wedge R(K^*, \widetilde{\boldsymbol{K}})$

and
If ATK = CPA then $O_1 = \varepsilon$ and $O_2 = \varepsilon$.
If ATK = CCA1 then $O_1 = KEM.Decrypt(sk, \cdot)$ and $O_2 = \varepsilon$.
If ATK = CCA2 then $O_1 = KEM.Decrypt(sk, \cdot)$ and $O_2 = KEM.Decrypt(sk, \cdot)$.

Π_{KEM} is secure in the sense of NM-ATK-KEM, where ATK$\in$ {CPA, CCA1, CCA2}, if for every polynomial $p(k)$, A runs in $p(k)$, outputs a valid key space $\mathcal{K}$ in $p(k)$, and outputs relation R computable in $p(k)$, and $Adv^{\text{NM-ATK}}_{A,\Pi_{\text{KEM}}}(k)$ is negligible. We insist that the adversary is unsuccessful if some ciphertext $\boldsymbol{C_0}[i]$ does not have a valid decryption (that is, $\perp \in \boldsymbol{K}$).

Equivalence results We can obtain the equivalence of all three formal definitions and a following Theorem 1 between IND-CCA2-KEM and NM-CCA2-KEM. (See more details and proofs in the full paper version.)

Theorem 1 *(IND-CCA2-KEM ⇔ NM-CCA2-KEM)*
If encryption scheme Π_{KEM} is secure in the sense of IND-CCA2-KEM, then Π_{KEM} is secure in the sense of NM-CCA2-KEM.

2.2 Data Encapsulation Mechanism

Formally, a data encapsulation mechanism DEM is given by a pair of algorithms $DEM.Encrypt(K, M)$ and $DEM.Decrypt(K, C)$, where:

1. The encryption algorithm $DEM.Encrypt(K,\ M)$ takes as input a secret key K, and a plaintext M. It outputs a ciphertext C. Here, K, M and C are byte strings, and M may have arbitrary length, and K's length is $DEM.KeyLen$.
2. The decryption algorithm $DEM.Decrypt(K, C)$ takes as input secret key K and ciphertext C. It outputs plaintext M.

DEM must satisfy the soundness, $DEM.Decrypt(K,\ DEM.Encrypt(K,\ M))$ = M.

Attack types of DEM We state following six attack types of DEM. In the first, we consider the first three attack types, these are for access to encryption oracle. First, we state P0, that is an attack type with no access to encryption oracle by adversary. Secondly, we state P1 (Chosen Plaintext Attack). P1 is an attack type with access to encryption oracle. However the adversary cannot access to encryption oracle after getting target ciphertext. Thirdly, we state P2 (Adaptive Chosen Plaintext Attack). In this type, an adversary can access to encryption oracle even if after the adversary gets target ciphertext. Moreover, we consider the last three attack types, these are for access to decryption oracle. First, we state C0, that is an attack type with no access to decryption oracle by adversary. Secondly, we state C1 (Chosen Ciphertext Attack). C1 is an attack type with access to decryption oracle. However the adversary cannot access to decryption oracle after getting target ciphertext. Thirdly, we state C2 (Adaptive Chosen Ciphertext Attack). In this type, an adversary can access to decryption oracle even if after the adversary gets target ciphertext.

Indistinguishability of DEM We state formal definition of indistinguishability for DEM in Fig.2 following [10], which we call IND-DEM. We also use IND-PX-CY-DEM to describe the security notion of indistinguishability for DEM against ATK $\in$ $\{$CPA, CCA1, CCA2$\}$.

Let Π_{DEM} = $(DEM.Encrypt, DEM.Decrypt)$ be an encryption scheme over message space M and let A = $(A_1,\ A_2)$ be an adversary. We insist that $A_1(1^k)$ outputs $\{x_0, x_1\} \in M$ with $|x_0|$ = $|x_1|$, where k is security parameter. Furthermore, when Y = 2, we insist that A_2 does not ask for the decryption of challenge ciphertext y.

Π_{DEM} is secure in the sense of IND-PX-CY for $\{$X, Y$\} \in \{0, 1, 2\}$ if $Adv^{\text{IND-PX-CY}}_{A,\Pi_{\mathrm{DEM}}}(\cdot)$ is negligible for any PPT adversary A.

Fig. 2. IND-DEM Definition

$$Adv^{\text{IND-PX-CY}}_{A,\Pi_{\mathrm{DEM}}}(k) \equiv 2 \cdot \Pr[Expt^{\text{IND-PX-CY}}_{A,\Pi_{\mathrm{DEM}}}(k)] - 1$$

where $Expt^{\text{IND-PX-CY}}_{A,\Pi_{\mathrm{DEM}}}(k)$

$$K \leftarrow \{0, 1\}^k; (x_0, x_1, s) \leftarrow A_1^{O_1, O_1'}(1^k); b \leftarrow \{0, 1\}; y \leftarrow DEM.Encrypt(K, x_b);$$

$$g \leftarrow A_2^{O_2, O_2'}(1^k, s, y); \text{return 1 iff} \quad g = b$$

and
If X = 0 then $O_1(\cdot) = \varepsilon$ and $O_2(\cdot) = \varepsilon$.
If X = 1 then $O_1(\cdot) = DEM.Encrypt(K, \cdot)$ and $O_2(\cdot) = \varepsilon$.
If X = 2 then $O_1(\cdot) = DEM.Encrypt(K, \cdot)$ and $O_2(\cdot) = DEM.Encrypt(K, \cdot)$.
If Y = 0 then $O_1'(\cdot) = \varepsilon$ and $O_2'(\cdot) = \varepsilon$.
If Y = 1 then $O_1'(\cdot) = DEM.Decrypt(K, \cdot)$ and $O_2'(\cdot) = \varepsilon$.
If Y = 2 then $O_1'(\cdot) = DEM.Decrypt(K, \cdot)$ and $O_2'(\cdot) = DEM.Decrypt(K, \cdot)$.

Non-Malleability of DEM We state formal definition of non-malleability for DEM in Fig.3 following Bellare[2] and Katz[10], which we call NM-DEM. We also use NM-PX-CY-DEM to describe the security notion of non-malleability for DEM for $\{$X, Y$\}$ $\in \{0, 1, 2\}$.

In Fig.3, M is a distribution over messages and R is some relation and k is security parameter. We require that $|x|$ = $|x'|$ for all x, x' in the support of M. We also require that the vector of ciphertexts $\boldsymbol{y}$ output by A_2 should be non-empty. Furthermore, when Y = 2, we insist that A_2 does not ask for the decryption of y.

Π_{DEM} is secure in the sense of NM-PX-CY for $\{$X, Y$\} \in \{0, 1, 2\}$ if $Adv^{\text{NM-PX-CY}}_{A,\Pi_{\mathrm{DEM}}}(k)$ is negligible for any PPT adversary A.

We obtain that the two above security notions of DEM yield the following Theorem 2. (Proof is in the full paper version.)

Theorem 2 *(NM-P2-C2-DEM $\Leftrightarrow$ IND-P2-C2-DEM)*
Encryption scheme Π_{DEM} is secure in the sense of NM-P2-C2 if and only if Π_{DEM} is secure in the sense of IND-P2-C2.

Fig. 3. NM-DEM Definition

$$Adv^{\text{NM-PX-CY}}_{A,\Pi_{\text{DEM}}}(k) \equiv \Pr[Expt^{\text{NM-PX-CY}}_{A,\Pi_{\text{DEM}}}(k)=1] - \Pr[\widetilde{Expt}^{\text{NM-PX-CY}}_{A,\Pi_{\text{DEM}}}(k)=1]$$

where

$Expt^{\text{NM-PX-CY}}_{A,\Pi_{\text{DEM}}}(k)$	$\widetilde{Expt}^{\text{NM-PX-CY}}_{A,\Pi_{\text{DEM}}}(k)$
$K\leftarrow\{0,1\}^k$	$K\leftarrow\{0,1\}^k$
$(M,s)\leftarrow A_1^{O_1,O'_1}(1^k)$	$(M,s)\leftarrow A_1^{O_1,O'_1}$
$x\leftarrow M$	$(x,\tilde{x})\leftarrow M$
$y\leftarrow DEM.Encrypt(K,x)$	$\tilde{y}\leftarrow DEM.Encrypt(K,\tilde{x})$
$(R,\boldsymbol{y})\leftarrow A_2^{O_2,O'_2}(s,y)$	$(R,\widetilde{\boldsymbol{y}})\leftarrow A_2^{O_2,O'_2}(s,\widetilde{y})$
$\boldsymbol{x}\leftarrow DEM.Decrypt(K,\boldsymbol{y})$	$\widetilde{\boldsymbol{x}}\leftarrow DEM.Decrypt(K,\widetilde{\boldsymbol{y}})$
return 1 iff $(y\notin\boldsymbol{y}) \ \wedge\ R(x,\boldsymbol{x})$	return 1 iff $(\widetilde{y}\notin\widetilde{\boldsymbol{y}}) \ \wedge\ R(x,\widetilde{\boldsymbol{x}})$

and
If X = 0 then $O_1(\cdot)=\varepsilon$ and $O_2(\cdot)=\varepsilon$.
If X = 1 then $O_1(\cdot)=DEM.Encrypt(K,\cdot)$ and $O_2(\cdot)=\varepsilon$.
If X = 2 then $O_1(\cdot)=DEM.Encrypt(K,\cdot)$ and $O_2(\cdot)=DEM.Encrypt(K,\cdot)$.
If Y = 0 then $O'_1(\cdot)=\varepsilon$ and $O'_2(\cdot)=\varepsilon$.
If Y = 1 then $O'_1(\cdot)=DEM.Decrypt(K,\cdot)$ and $O'_2(\cdot)=\varepsilon$.
If Y = 2 then $O'_1(\cdot)=DEM.Decrypt(K,\cdot)$ and $O'_2(\cdot)=DEM.Decrypt(K,\cdot)$.

3 Universally Composable KEM Is Equivalent to IND-CCA2 KEM

3.1 The Key Encryption Mechanithm Functionality $\mathcal{F}_{\text{KEM}}$

We define key encapsulation mechanism (KEM) functionality $\mathcal{F}_{\text{KEM}}$ in Fig.4. $\mathcal{F}_{\text{KEM}}$ is a functionality of KEM-key-generation, KEM-encryption and KEM-decryption. Here note that there is no functionality of data transmission between parties in $\mathcal{F}_{\text{KEM}}$.

3.2 UC KEM is Equivalent to IND-CCA2 KEM

Let KEM= $(KEM.KeyGen, KEM.Encrypt, KEM.Decrypt)$ be a key encapsulation mechanism. Consider the following transformation from KEM to protocol π_{KEM} that is constructed for realizing $\mathcal{F}_{\text{KEM}}$:

1. Upon input (KEM.KeyGen, sid) within some party P_j, P_j obtains the public key pk and secret key sk by running the algorithm $KEM.KeyGen()$, then outputs (KEM Key, sid, pk).
2. Upon input (KEM.Encrypt, sid, pk') within some party P_i, P_i obtains pair $(K^*, {C_0}^*)$ of a key and a ciphertext by running the algorithm $KEM.Encrypt(pk')$ and outputs (Encrypted Shared Key, sid, pk', $K^*, {C_0}^*$). (Note that it does not necessarily hold that pk'= pk).

Fig. 4. The Key Encapsulation Mechanism Functionality

Functionality $\mathcal{F}_{\mathrm{KEM}}$

$\mathcal{F}_{\mathrm{KEM}}$ proceed as follows, running with parties $P_1, \ldots, P_n$ and an adversary S.

KEM.KeyGen
In the first activation, expect to receive (KEM.KeyGen, sid) from some party P_j. Then,

1. Send (KEM.KeyGen, sid) to S.
2. Upon receiving (KEM Key, sid, pk) from S, send (KEM Key, sid, pk) to P_j.
3. If this is the first activation then record the pair (P_j, pk), otherwise pk is discarded.

KEM.Encrypt
Upon receiving (KEM.Encrypt, sid, pk') from some party P_i, proceed as follows:

– Check the memory, if $pk' = pk$, and if P_j is not corrupted, then proceeds as follows:
  1. Send (KEM.Encrypt, sid, pk') to S.
  2. Receive (Encrypted Shared Key, sid, pk', C_0) from S.
  3. If C_0 is stored in memory then halt.
  4. Choose Shared Key $K \xleftarrow{R} \{0,1\}^*$ randomly.
  5. Send (Encrypted Shared Key, sid, pk', K, C_0) to P_i.
  6. Store the pair (K, C_0) in memory.
– Otherwise (includes $pk' \neq pk$ or pk is not yet recorded, or P_j is corrupted),
  1. Send (KEM.Encrypt with Key, sid, pk') to S.
  2. Receive (Encrypted Shared Key, sid, pk', K, C_0) from S.
  3. Send (Encrypted Shared Key, sid, pk', K, C_0) to P_i.

KEM.Decrypt
Upon receiving (KEM.Decrypt, sid, C_0') from P_j (and P_j only), hand (KEM.Decrypt, sid, C_0') to S. Upon receiving (Shared Key, sid, K') from S, proceed as follows:

1. If a pair (K, C_0') exists in memory, send (Shared Key, sid, K) to P_j.
2. Otherwise, send (Shared Key, sid, K') to P_j.

3. Upon input (KEM.Decrypt, $sid, {C_0}^*$) within P_j, P_j obtains $K^* = KEM.Decrypt(sk$, ${C_0}^*)$ and output (Shared Key, sid, K^*).

Theorem 3 *π_{KEM} securely realizes $\mathcal{F}_{\mathrm{KEM}}$ with respect to non-adaptive adversaries if and only if KEM is indistinguishable against adaptive chosen ciphertext attacks (IND-CCA2 KEM).*

Proof. ("only if" part) Because NM-CCA2-KEM equals to IND-CCA2-KEM by Theorem 1, we prove that if π_{KEM} is not NM-CCA2-KEM secure, then π_{KEM} does not securely realize $\mathcal{F}_{\mathrm{KEM}}$. More details, we prove that we can construct an environment Z and a real life adversary A such that for any ideal process adversary (simulator) S,

Z can tell whether it is interacting with A and π_{KEM} or with S in the ideal process for $\mathcal{F}_{\mathrm{KEM}}$ by using the adversary G that breaks NM-CCA2-KEM.

Z proceeds as follows:

1. Activates key receiver P_j with (KEM.KeyGen, sid), and obtains pk.
2. Activates P_i with (KEM.Encrypt, sid, pk), and obtains $(K^*, {C_0}^*)$.
3. Activates G with pk and ${C_0}^*$, obtains (R, $\boldsymbol{C_0}$), where R is some relation.
4. Activates P_j with (KEM.Decrypt, sid, $\boldsymbol{C_0}[i]$) for each i, and obtains $\boldsymbol{K'}[i]$.
5. Return 1 iff $R(K^*, \boldsymbol{K'})$.

When Z interacts with A and π_{KEM}, Z obtains corresponding pair $(K^*, {C_0}^*)$ in Step 2. In this case, Z returns 1 in Step 5. On the other hand, Z interacts with S in the ideal process for $\mathcal{F}_{\mathrm{KEM}}$, Z obtains non-corresponding pair $(K^\star, {C_0}^*)$ in Step 2, where $K^\star \stackrel{R}{\longleftarrow} \{0,1\}^*$ by $\mathcal{F}_{\mathrm{KEM}}$ and ${C_0}^*$ is generated by S. For ${C_0}^*$, G successfully obtains (R, $\boldsymbol{C_0}$). However Z cannot output 1 in Step 5 because there is no relation $R(K^\star, \boldsymbol{K'})$.

("if" part) We show that if π_{KEM} does not securely realize $\mathcal{F}_{\mathrm{KEM}}$, then π_{KEM} is not IND-CCA2-KEM. More details, we assume that for any simulator S there is an adversary and an environment Z that can distinguish with non-negligible probability whether it interacts with S in the ideal process for $\mathcal{F}_{\mathrm{KEM}}$ or with parties running π_{KEM} and the adversary A in the real-life world. Then we prove that π_{KEM} is not IND-CCA2-secure by using the distinguishable environment Z.

We will show that Z can distinguish only when receiver P_j is not corrupted. We discuss all the cases as follows.

(Case 1: Receiver P_j is corrupted.) In this case, we can make simulator S such that the environment Z cannot distinguish the real life world from the ideal process world. Once A corrupts P_j, simulator S corrupts dummy party $\widetilde{P_j}$. However receiver P_i is not corrupted, that is, P_i is honest. Simulator S proceeds as follows:

1. When S receives (KEM.KeyGen, sid), it obtains (pk, sk) by running KEM.KeyGen(), and returns pk to $\mathcal{F}_{\mathrm{KEM}}$.
2. When S receives (KEM.Encrypt with Key, sid, pk), then S generates a corresponding pair (K, C_0) and returns C_0 to $\mathcal{F}_{\mathrm{KEM}}$.
3. When S receives (KEM.Decrypt, sid, C_0), S generates key K and returns K to $\mathcal{F}_{\mathrm{KEM}}$.

In this case Z cannot distinguish the real world from the ideal world because S can reconstruct by using the simulated copy of A. Note that, A can do stopping the protocol π_{KEM}. Even if this situation happens, Z cannot distinguish the real world from the ideal world, because S can also stop the protocol.

(Case 2: P_j is not corrupted.) We look at the generated key and ciphertext by P_i in each world.

- In the real life world, π_{KEM} runs among the honest parties, P_i generates corresponding pair (K^*, C_0^*) by running the algorithm $KEM.Encrypt(pk)$.
- In the ideal process world, when $\widetilde{P_i}$ sends (KEM.Encrypt, sid, pk) to $\mathcal{F}_{KEM}$, $\mathcal{F}_{KEM}$ obtains C_0 from S, and $\mathcal{F}_{KEM}$ chooses shared key $K \stackrel{R}{\longleftarrow} \{0,1\}^*$ at random. Then sends (Encrypted Shared Key, sid, pk, K, C_0) to P_i.

It is easily seen that C_0 is not concerned to the key K (because $\mathcal{F}_{KEM}$ randomly generates the key K). In the real world, Z obtains the corresponding pair $(K^*, {C_0}^*)$. However, in the ideal world, Z obtains the non-corresponding pair (K, C_0). Consequently, we can construct environment Z that can distinguish the real world from the ideal world.

Recall the formal settings, there are three types of messages between Z and A. That is, Z sends A a message either to corrupt parties, or to report on messages sending, or to deliver some message. In this protocol, no party corruption occurs during execution since we consider non-adaptive adversaries. Furthermore, parties don't send messages each other. Therefore, there are no request to report on or deliver messages. So, the way that S affects the output of Z is only the communication via $\mathcal{F}_{\mathrm{KEM}}$. As a result, S proceeds as follows:

1. When S receives a message (KEM.KeyGen, sid) from $\mathcal{F}_{\mathrm{KEM}}$, it runs the key generation algorithms KEM.KeyGen(), obtains the public key pk and the secret key sk, and returns pk to $\mathcal{F}_{\mathrm{KEM}}$.
2. When S receives a message (KEM.Encrypt, sid, pk) from $\mathcal{F}_{\mathrm{KEM}}$, then it generates C_0 from the output of the algorithm $KEM.Encrypt(pk)$, and returns C_0 to $\mathcal{F}_{\mathrm{KEM}}$.
3. When S receives a message (KEM.Encrypt with Key, sid, pk) from $\mathcal{F}_{\mathrm{KEM}}$, then it generates key (K, C_0) = $KEM.Encrypt(pk)$, and returns (K, C_0) to $\mathcal{F}_{\mathrm{KEM}}$.
4. When S receives a message (KEM.Decrypt, sid, C_0) from $\mathcal{F}_{\mathrm{KEM}}$, it obtains K = $KEM.Decrypt(sk, C_0)$ and returns K to $\mathcal{F}_{\mathrm{KEM}}$.

We assume that there is an environment Z that can distinguish the interaction in the real life world from that in the ideal process world. We prove that we can construct an adversary F that breaks IND-CCA2-KEM by using the distinguishable environment Z. Precisely, for some value of the security parameter z for Z, we assume that there is an environment Z such that $IDEAL_{F,S,Z}$(z) - $REAL_{\pi_{\mathrm{KEM}},A,Z}$(z) $> \sigma$, then we show that F correctly guesses the bit b with probability $\frac{1}{2} + \frac{\sigma}{2l}$ in the CCA2 game, where l is the total number of times invoking encryption oracle.

F is given a public key pk, and is allowed to query to decryption oracle and encryption oracle. First, F chooses a number $h \xleftarrow{R} \{1, \ldots, l\}$ at random. Secondly, F simulates Z on the following simulated interaction with a system running π_{KEM}. Let K_i and C_{0i} denote the i-th key and ciphertext that Z asks to encrypt in this simulation, respectively.

1. When Z activates some party P_j with (KEM.KeyGen, sid), F lets P_j output the value pk from F's input.
2. For the first $h-1$ times that Z asks some party P_i to generate shared key K_i, F lets P_i return (K_i, C_{0i}) by using algorithm (K_i, C_{0i}) = KEM.Encrypt(pk).
3. The h-th time that Z asks to generate key K_h, F queries its encryption oracle with pk, then obtains corresponding pair X= (K_h, C_{0h}) or non-corresponding pair X = (K'_h, C_{0h}) from encryption oracle. Accordingly, F hands X to Z as the test pair.
4. For the remaining $l-h$ times that Z asks P_i to generate shared key K_i, F lets P_i return (K_i, C_{0i}), where $K_i \xleftarrow{R} \{0,1\}^*$ randomly and C_0 from the output of algorithm $KEM.Encrypt(pk)$.

5. Whenever Z activates decryptor P_j with (KEM.Decrypt, sid, C_0), where $C_0 = C_{0i}$ for some i, F lets P_i return the corresponding key K_i for any i. If C_0 is different from all the C_{0i}'s, then F queries C_0 to its decryption oracle, obtains value v, and lets P_j return v to Z.
6. When Z halts, F outputs whatever Z outputs and halts.

We apply a standard hybrid argument for analyzing the success probability of F. Let the random variable D_i denote the output of Z from an interaction that is identical to an interaction with S in the ideal process, except that the first i pairs are computed with correctly generation, and the last pair are computed with non-corresponding generation. We can see that D_0 is identical to the output of Z in the ideal process world, and D_l is identical to the output of Z in the real life world. (This follows from the fact that the mechanism KEM guarantees that KEM.Decrypt(sk, C_0) = K, where C_0 = KEM.Encrypt(pk), this is called "soundness".) Furthermore, in the simulation of F, if the value C_{0h} that F obtains from its encryption oracle is an encryption of K_h then the output of the simulated Z has the distribution of D_{h-1}. If C_{0h} does not correspond to the encryption of the key then the output of the simulated Z has the distribution of D_h. As discussed above, we can construct attacker F by using the distinguishable environment Z. We can conclude that if π_{KEM} does not securely realize $\mathcal{F}_{\mathrm{KEM}}$, then π_{KEM} is not IND-CCA2-KEM.

□

4 Universally Composable DEM Is Equivalent to IND-P2-C2 DEM

4.1 The KEM-DEM functionality $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$

We define KEM-DEM functionality $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ in Fig.5 and Fig.6. $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ is a functionality of hybrid usage of KEM and DEM, KEM-key-generation, KEM-encryption, KEM-decryption, DEM-encryption and DEM-decryption. Information obtained in KEM-encryption and KEM-decryption is transfered to DEM-encryption and DEM-decryption inside $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$. Here note that there is no functionality of data transmission between parties in $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$.

4.2 UC DEM Is Equivalent to IND-P2-C2 DEM

First, we define a protocol $\pi_{\mathrm{KEM\text{-}DEM}}$ in Fig.7 that is constructed on an algorithm DEM = $(DEM.Encrypt, DEM.Decrypt)$ in the $\mathcal{F}_{\mathrm{KEM}}$-hybrid model. We say that the underlying DEM is *UC secure* if and only if $\pi_{\mathrm{KEM\text{-}DEM}}$ securely realizes $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ in the $\mathcal{F}_{\mathrm{KEM}}$-hybrid model.

Therefore, the following theorem implies that UC DEM is equivalent to IND-P2-C2 DEM.

Theorem 4 *Protocol $\pi_{\mathrm{KEM\text{-}DEM}}$ securely realizes $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$ with respect to non-adaptive adversaries in the $\mathcal{F}_{\mathrm{KEM}}$-hybrid model if and only if DEM is indistinguishable against adaptive chosen plaintext/ciphertext attacks(IND-P2-C2 DEM).*

Fig. 5. The KEM-DEM Functionality

Functionality $\mathcal{F}_{\text{KEM-DEM}}$

$\mathcal{F}_{\text{KEM-DEM}}$ proceeds as follows, running with parties $P_1, \ldots, P_n$ and an adversary S.

KEM.KeyGen

In the first activation, expect to receive (KEM.KeyGen, sid) from some party P_j. Then,

1. Send (KEM.KeyGen, sid) to S.
2. Upon receiving (KEM Key, sid, pk) from S, send (KEM Key, sid, pk) to P_j.

KEM.Encrypt

Upon receiving (KEM.Encrypt, sid, pk') from some party P_i, proceed as follows:

- If an entry (P_i, C, active) is not in memory for any C,
  1. Send (KEM.Encrypt, sid, pk') to S, and receive (Encrypted Shared Key, sid, pk', C_0) from S.
  2. Send (Encrypted Shared Key, sid, pk', C_0) to P_i, and store the pair (pk', C_0) and (P_i, C_0, active) in memory.
- Otherwise, do nothing.

KEM.Decrypt

Upon receiving (KEM.Decrypt, sid, C'_0) from P_j (and P_j only), hand (KEM.Decrypt, sid, C'_0) to S. Upon receiving ok from S, proceed as follows:

- If an entry (P_j, C, active) is not in memory for any C, send ok to P_j and store the pair (P_j, C'_0, active) in memory.
- Otherwise, do nothing.

DEM.Encrypt

Upon receiving (DEM.Encrypt, sid, m) from party P_e ($e \in \{i, j\}$ only), proceed as follows:

- If (P_e, C_0, active) is stored in memory.
  - If both P_e are uncorrupted, then proceeds as follows:
    1. Send (DEM.Encrypt, sid, $|m|$) to S, where $|m|$ denotes the length of m and receive (DEM.Ciphertext, sid, c') from S.
    2. Send (DEM.Ciphertext, sid, c') to P_e, and store the entry (m, c', C_0) in memory.
  - Otherwise, proceeds as follows:
    1. Send(DEM.Encrypt, sid, m) to S, and receive (DEM.Ciphertext, sid, c') from S.
    2. Send (DEM.Ciphertext, sid, c') to P_e, and store the entry (m, c', C_0) in memory.
- Otherwise, do nothing.

Fig. 6. The KEM-DEM Functionality

Functionality $\mathcal{F}_{\text{KEM-DEM}}$ (continued)

DEM.Decrypt

Upon receiving (DEM.Decrypt, sid, c') from P_e ($e \in \{i, j\}$ only), hand (DEM.Decrypt, sid, c') to S. Upon receiving (DEM.Plaintext, sid, ϕ) from S, proceed as follows:

- If an entry (P_e, C, active) exists in memory for some C:
  1. If the entry (m, c', C) is stored in the memory, then send (DEM.Plaintext, sid, m) to P_j.
  2. Else, if P_i and P_j is not corrupted, and if (m, c', C) doesn't recorded in the memory, then store the entry ($\perp$, c', C) and send (DEM.Plaintext, sid, $\perp$) to P_e.
  3. Else, if an entry ($\perp$, c', C) is recorded, then send (DEM.Plaintext, sid, $\perp$) to P_e.
  4. Otherwise, send (DEM.Plaintext, sid, ϕ) to P_e, and record the entry (ϕ, c', C) in memory.
- Otherwise, do nothing.

Proof. (sketch) ("only if" part) Because NM-P2-C2-DEM equals to IND-P2-C2-DEM by Theorem 2, we prove that if π_{DEM} is not NM-P2-C2-DEM secure, then $\pi_{\text{KEM-DEM}}$ does not securely realize $\mathcal{F}_{\text{KEM-DEM}}$ in the $\mathcal{F}_{\text{KEM}}$ - hybrid model. More details, we prove that we can construct an environment Z and a real life adversary A such that for any ideal process adversary (simulator) S, Z can tell whether it is interacting with A and $\pi_{\text{KEM-DEM}}$ or with S in the ideal process for $\mathcal{F}_{\text{KEM-DEM}}$ by using the adversary which breaks NM-P2-C2-DEM. Note that A corrupts no party and Z sends no messages to A. We assume that there exists a successful attacker G for π_{DEM} in the sense of NM-P2-C2-DEM. Environment Z proceeds as usual, except that Z runs a copy of G.

Z proceeds as above, except that Z runs a simulated copy of G. For more details:

1. Activates key receiver P_j with (KEM.KeyGen, sid), then obtains pk.
2. Activates key encrypter P_i with (KEM.Encrypt, sid, pk), then obtains ${C_0}^*$.
3. Activates P_j with (KEM.Decrypt, sid, C_0).
4. Activates message encrypter P_i with (DEM.Encrypt, sid, m), then obtains c.
5. Activates G on c, obtains (R, $\boldsymbol{c}$), where R is some relation.
6. Activates P_j with (DEM.Decrypt, sid, $\boldsymbol{c}[i]$) for each i, and obtains $\boldsymbol{m'}[i]$.
7. Return 1 iff $R(m, \boldsymbol{m'})$.

When Z interacts with A and $\pi_{\text{KEM-DEM}}$, Z obtains ciphertext c in Step 4. In this case, Z return 1 in Step 7. Therefore when Z interacts with A and $\pi_{\text{KEM-DEM}}$, Z outputs 1 with non-negligible probability. On the other hand, Z interacts with S in the ideal process for $\mathcal{F}_{\text{KEM}}$, Z also obtains ciphertext c in Step 4. For ciphertext c, G successfully obtains (R, $\boldsymbol{c}$). However Z cannot output 1 in Step 7 because there is no relation $R(m, \boldsymbol{m'})$.

("if" part) We prove that if $\pi_{\text{KEM-DEM}}$ does not securely realize $\mathcal{F}_{\text{KEM-DEM}}$, then π_{DEM} is not IND-P2-C2-DEM. More details, we assume that there is an adversary A

Fig. 7. The KEM-DEM Protocol

Protocol $\pi_{\text{KEM-DEM}}$

Key Encapsulation Mechanithm KEM

KEM.KeyGen

1. Upon input (KEM.KeyGen, sid), P_j sends (KEM.KeyGen, sid_1) to $\mathcal{F}_{\text{KEM}}$.
2. Upon receiving (KEM Key, sid_1, pk) from $\mathcal{F}_{\text{KEM}}$, P_j outputs pk.

KEM.Encrypt
Upon input (KEM.Encrypt, sid, pk) within party P_i,

- If boolean variable active is not set,
  1. P_i sends (KEM.Encrypt, sid_1, pk) to $\mathcal{F}_{\text{KEM}}$.
  2. Upon receiving (Encrypted Shared key, sid_1, pk, K, C_0) from $\mathcal{F}_{\text{KEM}}$, then P_i outputs C_0 and stores the key K in memory and sets a boolean variable active in memory.
- Otherwise, do nothing.

KEM.Decrypt
Upon input (KEM.Decrypt, sid, C_0) within P_j,

- If boolean variable active is not set,
  1. P_j sends (KEM.Decrypt, sid_1, C_0) to $\mathcal{F}_{\text{KEM}}$.
  2. Upon receiving (Shared Key sid_1, K), P_j stores K in memory and outputs ok and sets a boolean variable active in memory.
- Otherwise, do nothing.

Data Encapsulation Mechanithm DEM

DEM.Encrypt
Upon input (DEM.Encrypt, sid, m) from P_e ($e \in \{i, j\}$), proceeds as follows:

- If the boolean variable is active in P_e's memory, P_e obtains ciphertext c = DEM.Encrypt(K, m) and outputs (DEM Ciphertext, sid, c).
- Otherwise do nothing.

DEM.Decrypt
Upon input (DEM.Decrypt, sid, c) from P_e ($e \in \{i, j\}$), proceeds as follows:

- If the boolean variable is active in P_e's memory, P_e obtains m = DEM.Decrypt (K, c) and outputs (DEM Plaintext, sid, m).
- Otherwise do nothing.

such that for any simulator S, there is an environment Z can tell with non-negligible probability whether it is interacting with $\mathcal{F}_{\text{KEM-DEM}}$ and S in the ideal process world or with parties running $\pi_{\text{KEM-DEM}}$ and the adversary A in the real life world. Then, we prove that there is adversary F breaks IND-P2-C2-DEM by using distinguishable Z. Note that there are three cases of party corruption since we take account of non-adaptive adversaries.

Recall the formal settings, there are three types of messages between Z and A. That is, Z sends A a message either to corrupt parties, or to report on messages sending, or to deliver some message. In this protocol, no party corruption occurs during execution since we consider non-adaptive adversaries. Furthermore, parties don't send messages each other. Therefore, there are no request to report on or deliver messages. In fact, there is no communication between Z and A at all. So, the way that S affects the output of Z is only the communication via $\mathcal{F}_{\text{KEM-DEM}}$.

We will show that Z can distinguish is only when both sender P_i and receiver P_j are not corrupted. We discuss all the cases for the following simulator S as follows:

1. When S receives (KEM.KeyGen, sid), S obtains (pk, sk) by running KEM.KeyGen(), and returns (KEM Key, sid, pk) to $\mathcal{F}_{\text{KEM-DEM}}$.
2. When S receives (KEM.Encrypt, sid, pk), S generates a corresponding pair (K, C_0), and returns (Encrypted Shared Key, sid, pk, C_0) to $\mathcal{F}_{\text{KEM-DEM}}$.
3. When S receives (KEM.Decrypt, sid, C_0), S obtains key K by KEM.Decrypt(sk, C_0), and returns ok to $\mathcal{F}_{\text{KEM-DEM}}$.
4. When S receives (DEM.Encrypt, sid, $|m|$), S generates c' by output of DEM.Encry pt(K, $0^{|m|}$), and returns (DEM.Ciphertext, sid, c') to $\mathcal{F}_{\text{KEM-DEM}}$.
5. When S receives (DEM.Encrypt, sid, m), S generates c' by the output of DEM.Enc rypt(K, m) and returns (DEM.Ciphertext, sid, c') to $\mathcal{F}_{\text{KEM-DEM}}$.
6. When S receives (DEM.Decrypt, sid, c'), S generates ϕ by DEM.Decrypt(K, c'), and sends (DEM.Plaintext, sid, ϕ).

(Case 1: Sender P_i is corrupted.) In this case, once A corrupts P_i, simulator S corrupts dummy party $\widetilde{P_i}$. However receiver P_j is not corrupted, that is, P_j is honest. Environment Z cannot distinguish the real life world from the ideal process world for the above simulator S because S can reconstruct by using the simulated copy of A. Note that, A can do stopping the protocol $\pi_{\text{KEM-DEM}}$. Even if this situation is happened, Z cannot distinguish the real world from the ideal world, because S can also stop the protocol.

(Case 2: Receiver P_j is corrupted.) In this case, once A corrupts P_j, simulator S corrupts dummy party $\widetilde{P_j}$. However sender P_i is not corrupted, that is, P_i is honest. Environment Z cannot distinguish the real life world from the ideal process world by the above simulator S because simulator S can reconstruct by using the simulated copy of A.

(Case 3: No party is corrupted.) In this case, sender P_i and receiver P_j are not corrupted i.e., they are honest parties. We look at the generated key and ciphertext by P_i in each world.

– In the real life world, $\pi_{\text{KEM-DEM}}$ runs among the honest parties, P_i generates c by running the algorithm $DEM.Encrypt(K, m)$. Note that c is corresponding to m.
– In the ideal process world, $\mathcal{F}_{\text{KEM-DEM}}$ send (DEM.Encrypt, sid, $|m|$) to S. P_i obtains c' from S via $\mathcal{F}_{\text{KEM-DEM}}$. Note that c is non-corresponding to m because S sees only the length of m.

By applying a hybrid argument similar to the one in the proof of Theorem 3, we can obtain adversary F that attacks IND-P2-C2-DEM by using the environment Z that can distinguish the real world from the ideal world.

□

5 A Universally Composable Secure Channel Based on the KEM-DEM Framework

To realize secure channel functionality, $\mathcal{F}_{\mathrm{SC}}$, defined in [4], we define a secure channel protocol π_{SC} in Fig.8 in the ($\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$, $\mathcal{F}_{\mathrm{SIG}}$, $\mathcal{F}_{\mathrm{CA}}$)-hybrid model, where $\mathcal{F}_{\mathrm{SIG}}$ is a signature functionality [4], and $\mathcal{F}_{\mathrm{CA}}$ is certification authority functionality [4]. (Due to the page limitation, we omit the description of $\mathcal{F}_{\mathrm{SIG}}$ and $\mathcal{F}_{\mathrm{CA}}$. See [4] for the definitions.)

Combining with the previous theorems, the following theorem implies that IND-CCA2 KEM, IND-P2-C2 DEM, secure signatures and ideal CA are sufficient to securely realize $\mathcal{F}_{\mathrm{SC}}$.

Theorem 5 *Protocol π_{SC} securely realizes $\mathcal{F}_{\mathrm{SC}}$ in the ($\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$, $\mathcal{F}_{\mathrm{SIG}}$, $\mathcal{F}_{\mathrm{CA}}$)-hybrid model.*

Proof. (sketch) Let A be an adversary that interacts with parties running π_{SC} in the ($\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$, $\mathcal{F}_{\mathrm{SIG}}$, $\mathcal{F}_{\mathrm{CA}}$)-hybrid model, and S be an ideal process adversary (simulator) that interacts with the ideal process for $\mathcal{F}_{\mathrm{SC}}$. We construct S such that any environment Z cannot tell whether it is interacting with A in π_{SC} or with S in the ideal process for $\mathcal{F}_{\mathrm{SC}}$. S invokes a simulated copy of A, and proceeds as follows:

1. Inputs from Z are forwarded to A and outputs from A are forwarded to Z.
2. **(Simulating the interaction of A in the session set-up)** Upon receiving a message (sid, P_i, P_j) from $\mathcal{F}_{\mathrm{SC}}$ (which means that P_i and P_j have set-up a session), simulates for A the process of exchanging shared key between P_i and P_j. That is, play functionalities, $\mathcal{F}_{\mathrm{CA}}$, $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$, $\mathcal{F}_{\mathrm{SIG}}$, for A as follows: send to A (in the name of $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$) the message (KEM.KeyGen, sid_1, PK_i), obtain the response (KEM Key, sid_1, PK_i) from A; send to A (in the name of $\mathcal{F}_{\mathrm{CA}}$) the message (Registered, P_i, PK_i), obtain the response ok from A; send to A (in the name of $\mathcal{F}_{\mathrm{CA}}$) the message (Retrieve, P_i, P_j), obtain the response ok from A; send to A (in the name of $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$) the message (KEM.Encrypt, sid_1, PK_i), obtain the response (Encrypted Shared key, sid_1, PK_i, C_0) from A; send to A (in the name of $\mathcal{F}_{\mathrm{SIG}}$) the message (KeyGen, (P_j, sid')), obtain the response (Verification Key, $(P_j, sid'), PK_j$) from A; send to A (in the name of $\mathcal{F}_{\mathrm{CA}}$) the message (Registered, P_j, PK_j), obtain the response ok from A; send to A (in the name of $\mathcal{F}_{\mathrm{SIG}}$) the message (Sign, (P_j, sid'), C_0), obtain the response (Signature, (P_j, sid'), C_0, σ) from A; send to A (in the name of $\mathcal{F}_{\mathrm{CA}}$) the message (Retrieve, P_j, P_i), obtain the response ok from A; send to A (in the name of $\mathcal{F}_{\mathrm{SIG}}$) the message (Verify, (P_j, sid'), C_0, σ, PK_j), obtain the response (Verified, (P_j, sid'), C_0, ϕ) from A; send to A (in the name of $\mathcal{F}_{\mathrm{KEM\text{-}DEM}}$) the message (KEM.Decrypt, sid_1, C_0, PK_i), obtain the response ok from A.

Fig. 8. The Secure Channel Protocol $\pi_{\rm SC}$

Protocol $\pi_{\rm SC}$

Session Set-up

1. Upon input (Establish-session, sid, P_j, initiator), P_i sends (KEM.KeyGen, sid_1) to $\mathcal{F}_{\text{KEM-DEM}}$, and stores (sid, P_j).
2. Upon receiving (KEM Key, sid_1, PK_i) from $\mathcal{F}_{\text{KEM-DEM}}$, P_i sends (Register, P_i, PK_i) to $\mathcal{F}_{\rm CA}$.
3. Upon input (Establish-session, sid, P_i, responder), P_j sends (Retrieve, P_i) to $\mathcal{F}_{\rm CA}$.
4. Upon receiving (Retrieve, P_i, PK_i) from $\mathcal{F}_{\rm CA}$, P_j sends (KEM.Encrypt, sid_1, PK_i) to $\mathcal{F}_{\text{KEM-DEM}}$, and receives (Encrypted Shared key, sid_1, PK_i, C_0) from $\mathcal{F}_{\text{KEM-DEM}}$.
5. P_j sends (KeyGen, (P_j, sid')) to $\mathcal{F}_{\rm SIG}$, receives (Verification Key, (P_j, sid'), PK_j).
6. P_j sends (Register, P_j, PK_j) to $\mathcal{F}_{\rm CA}$, then sends (Sign, P_j, C_0) to $\mathcal{F}_{\rm SIG}$, receives (Signature, (P_j, sid'), C_0, σ) from $\mathcal{F}_{\rm SIG}$.
7. P_j sends (sid, C_0, σ, P_j) to P_i, and set a boolean variable active.
8. Upon receiving (sid, C_0, σ, P_j), P_i checks whether (sid, P_j) is stored. If it is not stored, discard the message. Otherwise, P_i sends (Retrieve, P_j) to $\mathcal{F}_{\rm CA}$ and receives (Retrieve, P_j, PK_j), then sends (Verify, (P_j, sid'), C_0, σ, PK_j) to $\mathcal{F}_{\rm SIG}$ and receives (Verified, (P_j, sid'), C_0, f). If f is 1 then P_i goes to next step. Else finish the protocol.
9. P_i sends (KEM.Decrypt, sid_1, C_0) to $\mathcal{F}_{\text{KEM-DEM}}$. If ok is returned from $\mathcal{F}_{\text{KEM-DEM}}$, set a boolean variable active.

Data Exchange

1. Upon input (Send, sid, m), to P_e, if P_e is active (i.e., $e \in \{i, j\}$), P_e sends the message (DEM.Encrypt, sid_1, m) to $\mathcal{F}_{\text{KEM-DEM}}$.
2. Upon receiving (DEM.Ciphertext, c) from $\mathcal{F}_{\text{KEM-DEM}}$, P_e sends c to $P_{\bar{e}}$.
3. Upon receiving c, if $P_{\bar{e}}$ is active (i.e., $\bar{e} \in \{i, j\}$), $P_{\bar{e}}$ sends (DEM.Decrypt, sid_1, c) to $\mathcal{F}_{\text{KEM-DEM}}$.
4. $P_{\bar{e}}$ receives (DEM.Plaintext, m) from $\mathcal{F}_{\text{KEM-DEM}}$ and outputs m.

Session Ending

1. Upon input (Expire-session, sid), P_e sends (Expire-session, sid) to $P_{\bar{e}}$ and erases the session state (including all keys and local values) and terminates this protocol.
2. Upon receiving (Expire-session, sid), $P_{\bar{e}}$ erases the session state (including all keys and local values) and terminates this protocol.

3. **(Simulating the interaction of A in the data exchange)** Upon receiving a message (sid, P_e, u) ($e \in \{i, j\}$) from $\mathcal{F}_{\rm SC}$ (which means that P_e sent a message of length u to $P_{\bar{e}}$), simulates for A the process of exchanging shared key between P_i and P_j. That is, play functionality $\mathcal{F}_{\text{KEM-DEM}}$ for A as follows: send to A (in the name of $\mathcal{F}_{\text{KEM-DEM}}$) the message (DEM.Encrypt, $sid_1, |m|$), obtain the response (DEM.Ciphertext, c) from A; send to A (in the name of $\mathcal{F}_{\text{KEM-DEM}}$) the message (DEM.Decrypt, sid_1, c), obtain the response (DEM.Plaintext, ψ) from A.

4. **(Simulating the interaction of a corrupted party)** Simulating the interaction of a corrupted party can be done by simulating the functionalities and transmissions in the natural way. So, we omit the precise description here.
5. **(Simulating party corruption)** When A corrupts a party, S corrupts that party in the ideal process, and forwards the obtained information to A. This poses no problem since none of the parties maintains any secret information.

It is straightforward to verify that the simulation is perfect. That is, for any environment Z and A, it holds that the view of Z interacting with S and $\mathcal{F}_{\mathrm{SC}}$ is distributed identically to the view of Z interacting with A and parties running protocol π_{SC} in the $(\mathcal{F}_{\mathrm{KEM\text{-}DEM}}, \mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{CA}})$-hybrid model.

□

6 Conclusion

The KEM-DEM framework is a promising formulation for hybrid encryption based on symmetric and asymmetric encryption, and will be standardized in ISO in the near future. This paper studied the possibility of constructing a *UC secure channel* using the KEM-DEM framework. We presented that IND-CCA2 KEM and IND-P2-C2 DEM along with secure signatures and ideal certification authority are sufficient to realize a UC secure channel. This paper also shows several equivalence results: UC KEM, IND-CCA2 KEM and NM-CCA2 KEM are equivalent, and UC DEM, IND-P2-C2 DEM and NM-P2-C2 DEM are equivalent.

References

1. M.Bellare, A.Desai, D.Pointcheval, and P.Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes," Crypto'98 LNCS 1462.
2. M.Bellare and A.Sahai, "Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterisation," Crypto'99 LNCS 1666.
3. R. Canetti, "Universally Composable Security: A New paradigm for Cryptographic Protocols," 42nd FOCS, 2001. Full version available at http://eprint.iacr.org/2000/067.
4. R. Canetti, "Universally Composable Signature, Certification, and Authentication," August, 2004. http://eprint.iacr.org/2003/239/.
5. R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels," Eurocrypt 01, 2001. Full version at http://eprint.iacr.org/2001.
6. R. Canetti and H. Krawczyk, "Universally Composable Notions of Key Exchange and Secure Channels," Eurocrypt 02, LNCS, Springer, 2002. http://eprint.iacr.org/2002.
7. R. Canetti and T. Rabin, "Universal Composition with Joint State," Proceedings of Crypto 03, LNCS, Springer, 2003. available at http://eprint.iacr.org/2002.
8. R.Cramer and V.Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack, " http://shoup.net/papers/, 2001 Dec.
9. D.Dolev, C.Dwork, and M.Naor, "Non-Malleable Cryptography," 23rd STOC, 1991. Also Technical Report CS95-27, Weizmann Institute of Science, 1995.
10. J. Katz and M.Yung, "Characterization of Security Notions for Probabilistic Private-Key Encryption," to appear. Full version available at http://www.cs.umd.edu/~jkatz/.
11. V.Shoup, "A Proposal for an ISO Standard for Public Key Encryption (version 2.1)," ISO/IEC JTC1/SC27, N2563, http://shoup.net/papers/, 2001 Dec.