Tag-KEM/DEM: A New Framework for Hybrid Encryption

Masayuki Abe[1], Rosario Gennaro[2] and Kaoru Kurosawa[3]

[1] NTT Information Sharing Platform Laboratories, NTT Corporation, Japan
abe.masayuki@lab.ntt.co.jp
[2] IBM T.J.Watson Research Center, USA
Rosario@us.ibm.com
[3] Ibaraki University, Japan
kurosawa@cis.ibaraki.ac.jp

Abstract. This paper presents a novel framework for generic construction of hybrid encryption schemes which produces more efficient schemes than before. A known framework introduced by Shoup combines a key encapsulation mechanism (KEM) and a data encryption mechanism (DEM). While it is believed that both of the components must be secure against chosen ciphertext attacks, Kurosawa and Desmedt showed a particular example of KEM that might not be CCA but can be securely combined with CCA DEM yielding more efficient hybrid encryption scheme. There are also many efficient hybrid encryption schemes in various settings that do not fit to the framework. These facts serve as motivation to seek another framework that yields more efficient schemes.
In addition to the potential efficiency of the resulting schemes, our framework will provide insightful explanation about existing schemes that do not fit to the previous framework. This could result in finding improvements for some schemes. Moreover, it allows immediate conversion from a class of threshold public-key encryption to a hybrid one without considerable overhead, which is not possible in the previous approach.

1 Introduction

A fundamental task of cryptography is to protect the secrecy of messages transmitted over public communication lines. For this purpose we use *encryption schemes* which use some secret information (a key) to encode messages in a way that an eavesdropper cannot decode it. However, as networks become more open and accessible, it becomes apparently clear that an adversary may not be limited to eavesdropping, but may take a more active role. She may try to interact with honest parties, by, for example, sending ciphertexts to them (possibly related to the ciphertexts she intends to decrypt) and analyze their response. Such active attacks can be proven to be much more powerful and hard to combat than passive ones (see for example [4]).

To model this type of attacks, the notion of *chosen-ciphertext security* was introduced by Naor and Yung [18] and developed by Rackoff and Simon [20], and Dolev, Dwork, and Naor [13]. Security against a chosen ciphertext attack (CCA security, in short) means that, even if the adversary is allowed to query a *decryption oracle* on ciphertext of her choosing, then she obtains no information about messages encrypted in other ciphertexts. The first CCA-secure cryptosystems were presented in [18, 20, 13], but they were quite impractical, as they rely on generic techniques for non-interactive zero-knowledge. In a breakthrough result, Cramer

and Shoup in [9] presented the first truly practical CCA-secure cryptosystem, whose security was based on the hardness of the decisional Diffie-Hellman problem. This construction was generalized in [10], using a new cryptographic primitive called *projective hash functions*.

Public-key encryption schemes often limit the message space to a particular group, which can be restrictive when one wants to encrypt arbitrary messages. For this purpose *hybrid* schemes are devised, composed by the two parts. First a *Key Encapsulation Mechanism* (KEM) is invoked: a random group element is encrypted and then mapped via a key derivation function into a random key. Then a *Data Encapsulation Mechanism* is performed: the previous key is used to encrypt the message using a symmetric encryption scheme. A formal treatment is found in [23, 11] and we refer it as KEM/DEM framework.

As in the literature, it is believed that both KEM and DEM must be CCA-secure to obtain CCA-secure hybrid encryption. It indeed looks quite reasonable because, if either of them are malleable, the adversary may be able to alter the corresponding part of the ciphertext and uses decryption oracle to get useful information. Recently in [17], Kurosawa and Desmedt introduced a hybrid encryption scheme which is a modification of the hybrid scheme presented in [21]. Their scheme is interesting from a theoretical point of view; When one looks at it as a KEM/DEM scheme, we do not know if their KEM is CCA-secure. Nevertheless, the resulting scheme is CCA-secure and more efficient than before both in computation and bandwidth. Thus the Kurosawa-Desmedt scheme points out that to obtain CCA-secure hybrid encryption, requiring both KEM/DEM to be CCA-secure, while being a sufficient condition, may not be a necessary one, and might indeed be an overkill. There are other hybrid encryption schemes in the literature, e.g.,[3, 19], which are very efficient, mostly in the random oracle model, but do not fit to the CCA-secure KEM/DEM framework.

Our Contribution. Prompted by the above observation, we set out to investigate another KEM/DEM framework that yields more efficient hybrid encryption and captures wider variety of existing schemes. Our results can be summarized as follows:

- We introduce a Tag-KEM which also takes as input a *tag*. Though such a notion is known in the literature, e.g., [23], we give an extended syntax and show, somewhat surprisingly, that if one uses a CCA-secure Tag-KEM in a novel way we show then it is sufficient for the DEM to be secure simply against a passive attacker to yield CCA-secure hybrid encryption.
- We present several constructions of CCA-secure Tag-KEMs based on various combination of assumptions on the tools used to build them. A class of KEM that is strictly less secure than CCA-secure ones but can yield CCA-secure Tag-KEM is shown. Importantly, we show the KEM by Kurosawa and Desmedt belongs to this class, which comes up with theoretical understanding of their scheme. This answers to the open question of [17].
- We show that the Tag-KEM/DEM framework provides a simple way to create threshold versions of CCA-secure hybrid encryption schemes, which is not possible in the KEM/DEM framework.

– Finally, we show how several schemes in the literature can be casted in our Tag-KEM/DEM framework. Furthermore we show that some of those schemes can actually be simplified when considered as instances of our framework.

2 Definitions

2.1 Key Encapsulation Mechanism with Tags

In Shoup's model, a KEM consists of three algorithms as public-key encryption does, except that the encryption algorithm takes only pk and outputs a random one-time key and its encryption. (See Appendix A.1.) The encryption function may also take a string as an input associated to every ciphertext. In our model, we divide the encryption function into two functions in such a way that the first one selects a random key and the second one encrypts the key along with a given tag. We call a KEM that meets this model a Tag-KEM. Formally:

$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^\lambda)$ A probabilistic algorithm that generates public-key pk and private-key sk. The public-key defines all relative spaces, i.e., spaces for tags and encapsulated keys denoted by $\mathcal{T}$ and $\mathcal{K}_K$.

$(\omega, dk) \leftarrow \mathsf{TKEM.Key}(pk)$ A probabilistic algorithm that outputs one-time key $dk \in \mathcal{K}_D$ and internal state information ω. $\mathcal{K}_D$ is the key-space of DEM.

$\psi \leftarrow \mathsf{TKEM.Enc}(\omega, \tau)$ A probabilistic algorithm that encrypts dk (embedded in ω) into ψ along with τ, where τ is called a tag.

$dk \leftarrow \mathsf{TKEM.Dec}_{sk}(\psi, \tau)$ A decryption algorithm that recovers dk from ψ and τ. For soundness,$\mathsf{TKEM.Dec}_{sk}(\psi, \tau) = dk$ must hold for any sk, dk, ψ, and τ, associated by the above three functions.

Tag-KEM is a generalization of KEM because if the tag is a fixed string, it is a KEM. Note that, in the above syntactic definition, τ is not included in ψ and explicitly given to $\mathsf{TKEM.Dec}$. Such explicit treatment of τ has some notational advantages when we consider an adversary who tries to alter the tag without affecting to the ciphertext.

Tags associated to PKE or KEM can be found in the literature (e.g. see [24, 23]), but their syntactic definition and the purpose are different from those of ours; A tag is supposed to carry an identity of the encryptor and has to be fixed before DEM key is selected. Despite the limitations, their particular implementation fits also to our model without essential modification.

The security of a Tag-KEM requires that the adversary should fail to distinguish whether a given dk is the one embedded in cipehrtext (ψ, τ) or not, with adaptive access to the decryption oracle. Let $\mathcal{O}$ be the decryption oracle, $\mathsf{TKEM.Dec}_{sk}(\cdot, \cdot)$. Let A_T be a polynomial-time oracle machine that plays the following game.

[GAME.TKEM]

Step 1. $(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^\lambda)$
Step 2. $\upsilon_1 \leftarrow {A_T}^{\mathcal{O}}(pk)$
Step 3. $(\omega, dk_1) \leftarrow \mathsf{TKEM.Key}(pk)$, $dk_0 \leftarrow \mathcal{K}_D$, $\delta \leftarrow \{0,1\}$.
Step 4. $(\tau, \upsilon_2) \leftarrow {A_T}^{\mathcal{O}}(\upsilon_1, dk_\delta)$
Step 5. $\psi \leftarrow \mathsf{TKEM.Enc}(\omega, \tau)$
Step 6. $\tilde{\delta} \leftarrow {A_T}^{\mathcal{O}}(\upsilon_2, \psi)$

In Step 6, A_T is restricted not to ask (ψ, τ) to decryption oracle $\mathcal{O}$. Variable υ_1, υ_2 are state information of the adversary. Variable dk_δ is set to either dk_0 or dk_1 according to the value of $\delta \in \{0,1\}$. Such convention is used throughout the paper unless otherwise noted. We define $\epsilon_{\mathrm{tkem},A_T} = \left|\Pr[\tilde{\delta} = \delta] - \frac{1}{2}\right|$ and $\epsilon_{\mathrm{tkem}} = \max_{A_T}(\epsilon_{\mathrm{tkem},A_T})$ where maximum is taken over all machines. We say that a Tag-KEM is CCA-secure if ϵ_{tkem} is negligible in λ.

2.2 Data Encapsulation Mechanism

A DEM is a symmetric encryption scheme that consists of two algorithms, $\mathsf{DEM.Enc}$ and $\mathsf{DEM.Dec}$ associated to key-space $\mathcal{K}_D$ defined by λ. We consider $\mathcal{K}_D$ is $\{0,1\}^\lambda$ and message space is $\{0,1\}^*$.

$\chi \leftarrow \mathsf{DEM.Enc}_{dk}(m)$ An encryption algorithm that encrypts m into ciphertext χ by using symmetric-key $dk \in \mathcal{K}_D$.

$m \leftarrow \mathsf{DEM.Dec}_{dk}(\chi)$ A corresponding decryption algorithm that recovers message m from input ciphertext χ. Obvious soundness condition applies.

We only require passive security for DEM. Let A_D be a polynomial-time machine that plays the following game.

[GAME.DEM]

Step 1. $(m_0, m_1, \upsilon) \leftarrow A_D(1^\lambda)$.
Step 2. $dk \leftarrow \mathcal{K}_D$, $\xi \leftarrow \{0,1\}$, $\chi \leftarrow \mathsf{DEM.Enc}_{dk}(m_\xi)$.
Step 3. $\tilde{\xi} \leftarrow A_D(\upsilon, \chi)$

The messages, m_0 and m_1 must be the same length. Let $\epsilon_{\mathrm{dem},A_D} = \left|\Pr[\tilde{\xi} = \xi] - \frac{1}{2}\right|$ and $\epsilon_{\mathrm{dem}} = \max_{A_D}(\epsilon_{\mathrm{dem},A_D})$ where maximum is taken over all machines. We say that a DEM is one-time secure if ϵ_{dem} is negligible in λ. One-time pad is a simple example that fulfills this security notion.

2.3 Public-Key Encryption

A public-key encryption scheme consists of three algorithms, $\mathsf{PKE.Gen}$, $\mathsf{PKE.Enc}$, and $\mathsf{PKE.Dec}$:

$(pk, sk) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$ A probabilistic algorithm that on input the security parameter λ, generates public and private keys (pk, sk). The public-key defines the message space $\mathcal{M}$.

$c \leftarrow \mathsf{PKE.Enc}_{pk}(m)$ A probabilistic algorithm that encrypts a message $m \in \mathcal{M}$ into a ciphertext c.

$m \leftarrow \mathsf{PKE.Dec}_{sk}(c)$ An algorithm that decrypts c. It outputs either $m \in \mathcal{M}$ or a special symbol $\perp \notin \mathcal{M}$. An obvious soundness condition applies.

Let A_E be a polynomial-time oracle machine that plays the following game. By $\mathcal{O}$, we denote decryption oracle, $\mathsf{PKE.Dec}_{sk}(\cdot)$

[GAME.PKE]

Step 1. $(pk, sk) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$
Step 2. $(m_0, m_1, \upsilon) \leftarrow {A_E}^{\mathcal{O}}(pk)$
Step 3. $b \leftarrow \{0,1\}$, $c = \mathsf{PKE.Enc}_{pk}(m_b)$.
Step 4. $\tilde{b} \leftarrow {A_E}^{\mathcal{O}}(\upsilon, c)$

In Step 4, A_E is restricted not to ask c to $\mathcal{O}$. In addition, m_0 and m_1 must be of the same length. We define $\epsilon_{\mathrm{pke},A_E} = \left|\Pr[\tilde{b} = b] - \frac{1}{2}\right|$ and $\epsilon_{\mathrm{pke}} = \max_{A_E}(\epsilon_{\mathrm{pke},A_E})$ where maximum is taken over all ppt machines. We say that a PKE is CCA-secure if ϵ_{pke} is negligible in λ.

3 Generic Construction of Hybrid PKE

In GAME.TKEM, it is important to see that the same ψ can be asked to the decryption oracle as long as τ is different. Therefore, to conform CCA-security, the pair (ψ, τ) must be non-malleable, which means that CCA-secure Tag-KEM provides integrity to τ. We exploit this property to protect the DEM part so as to be non-malleable.

Now in our construction of hybrid PKE, we require that Tag-KEM accepts any string as a tag, i.e., $\mathcal{T} = \{0,1\}^*$. First of all, PKE.Gen is the same as TKEM.Gen; Given security parameter λ, it outputs pubic-key pk and private-key sk. Encryption and decryption functions are as follows.

Function: $\mathsf{PKE.Enc}_{pk}(m)$	**Function:** $\mathsf{PKE.Dec}(sk, c)$
$(\omega, dk) \leftarrow \mathsf{TKEM.Key}(pk)$	$(\psi, \chi) \leftarrow c$
$\chi \leftarrow \mathsf{DEM.Enc}_{dk}(m)$	$dk \leftarrow \mathsf{TKEM.Dec}_{sk}(\psi, \chi)$
$\psi \leftarrow \mathsf{TKEM.Enc}(\omega, \chi)$	$m \leftarrow \mathsf{DEM.Dec}_{dk}(\chi)$
Output $c = (\psi, \chi)$	Output m

When the length of DEM key varies depending on the length of message, like one-time pad, the syntax of Tag-KEM will be modified so that TKEM.Enc and TKEM.Dec can take necessary information.

Theorem 1. *If Tag-KEM is CCA secure and DEM is one-time secure then the Hybrid PKE scheme in Section 3 is CCA secure. In particular,* $\epsilon_{\mathrm{pke}} < 2\epsilon_{\mathrm{tkem}} + \epsilon_{\mathrm{dem}}$.

Proof. First we modify PKE.Enc in Step-3 of GAME.PKE so that DEM.Enc takes random key $dk^{\times}$ instead of the legitimate one generated by TKEM.Key. Call this game GAME.PKE'. Let X and X' be events that $\tilde{b} = b$ happens in GAME.PKE and GAME.PKE', respectively. Then, we claim that $|\Pr[X] - \Pr[X']| \leq 2\,\epsilon_{\mathrm{tkem}}$, which is shown by constructing A_T that attacks the underlying Tag-KEM scheme by using A_E. First A_T is given public-key pk and passes it to A_E. Given m_0 and m_1 from A_E, A_T requests dk_δ to the encryption oracle of GAME.TKEM. A_T then selects $b \leftarrow \{0,1\}$ and computes $\chi = \mathsf{DEM.Enc}_{dk_\delta}(m_b)$. By sending TKEM.Enc χ as a tag, A_T receives ψ and sends ciphertext (ψ, χ) to A_E. Every decryption query from A_E is forwarded to decryption oracle TKEM.Dec. If $\perp$ is returned, it is forwarded to A_E. Otherwise, A_K decrypts χ by using the key given from oracle TKEM.Dec and pass the resulting message to A_E. When A_E outputs $\tilde{b} = b$, A_K outputs $\tilde{\delta} = 1$ meaning that dk_δ is the real key. Otherwise, if A_E outputs $\tilde{b} \neq b$, A_K outputs $\tilde{\delta} = 0$ meaning that dk_δ is random. Now observe that the view of A_E is identical to that in GAME.PKE when $\delta = 1$, and that in GAME.PKE' when $\delta = 0$. Accordingly, $\Pr[\tilde{b} = b|\delta = 1] = \Pr[X]$ and $\Pr[\tilde{b} = b|\delta = 0] = \Pr[X']$. Therefore,

$$\begin{aligned}
\Pr[\tilde{\delta} = \delta] - \frac{1}{2} &= \frac{1}{2}(\Pr[\tilde{\delta} = 1|\delta = 1] - \Pr[\tilde{\delta} = 1|\delta = 0]) \\
&= \frac{1}{2}(\Pr[\tilde{b} = b|\delta = 1] - \Pr[\tilde{b} = b|\delta = 0]) \\
&= \frac{1}{2}(\Pr[X] - \Pr[X'])
\end{aligned}$$

Since $\left|\Pr[\tilde{\delta} = \delta] - \frac{1}{2}\right| \leq \epsilon_{\mathrm{tkem}}$, we have $|\Pr[X] - \Pr[X']| \leq 2\epsilon_{\mathrm{tkem}}$.

Next, we show that A_E playing GAME.PKE' essentially conducts a passive attack to DEM, i.e., $\left|\Pr[X'] - \frac{1}{2}\right| \leq \epsilon_{\mathrm{dem}}$. It is shown by constructing A_D that plays GAME.DEM by using A_E. A_D first generates (pk, sk) by using PKE.Gen and gives pk to A_E. When m_0 and m_1 are given from A_E, A_D forwards them to encryption oracle of GAME.DEM and receives cipehrtext χ. It then computes ψ by following TKEM.Key and TKEM.Enc by using χ as a tag, and sends $c = (\psi, \chi)$ to A_E. Note that the key chosen by the encryption oracle of GAME.DEM and the one embedded in ψ are independent and randomly chosen. All decryption queries are correctly processed by using sk. When A_E outputs $\tilde{b}$, A_D outputs $\tilde{\xi} = \tilde{b}$. It is now easy to see that, in this construction, GAME.PKE' is perfectly simulated and whenever A_E wins, so does A_D. Hence $\left|\Pr[X'] - \frac{1}{2}\right| \leq \epsilon_{\mathrm{dem}}$. The major factors of the running time of A_D is that of A_E and that for simulating the decryption oracle which grows linearly in the number of decryption queries.

In summary, we have:

$$\begin{aligned}
|(\Pr[X] - \frac{1}{2}) - (\Pr[X'] - \frac{1}{2})| &\leq 2\epsilon_{\mathrm{tkem}} \\
\epsilon_{\mathrm{pke}} - \epsilon_{\mathrm{dem}} &\leq 2\epsilon_{\mathrm{tkem}} \\
\epsilon_{\mathrm{pke}} &\leq 2\epsilon_{\mathrm{tkem}} + \epsilon_{\mathrm{dem}}
\end{aligned}$$

where ϵ_{tkem} and ϵ_{dem} are assumed negligible. □

4 Construction of Tag-KEM

This section develops some methods for obtaining Tag-KEM from PKE or KEM. (Note that KEM is generally obtained from PKE. Hence starting from KEM is more generic.) Since some methods are available to convert weak PKE to CCA-secure one in various setting, we assume CCA-secure PKE is available.

4.1 Based on PKE with Long Plaintext

When CCA-secure PKE is available, the first idea would be to encrypt the tag as a part of the plaintext together with the DEM key to encapsulate. It indeed works well if there is enough space in a plaintext. Lengthy tags would be compressed by using a hash function. We show that a target collision-free hash function (see Appendix B) is sufficient for this purpose. Formally, we construct Tag-KEM from PKE as follows. $\mathsf{TKEM.Gen}$ is essentially the same as $\mathsf{PKE.Gen}$; It outputs (pk, sk). It also selects hash function H. (For notational simplicity, we assume that H is included in pk and sk.) $\mathsf{TKEM.Key}$ chooses random dk from $\mathcal{K}_D$. It also outputs state information $\omega = pk||dk$. The encryption and decryption functions are as follows.

Function: $\mathsf{TKEM.Enc}(\omega, \tau)$

$(pk, dk) \leftarrow \omega$
$\tau' = H(\tau)$
$\psi = \mathsf{PKE.Enc}_{pk}(dk||\tau')$
Output ψ.

Function: $\mathsf{TKEM.Dec}_{sk}(\psi, \tau)$

$dk||\tau' \leftarrow \mathsf{PKE.Dec}(sk, \psi)$
If $\tau' = H(\tau)$, return dk.
Return $\perp$, otherwise.

Let ϵ_{tch} be the success probability of finding a collision of H. (Formal security definitions and related notations are given in Appendix B.)

Theorem 2. *If PKE is CCA-secure and H is target collision-free, the above Tag-KEM is CCA-secure. Especially, $\epsilon_{\mathrm{tkem}} \leq \epsilon_{\mathrm{pke}} + \epsilon_{\mathrm{tch}}$.*

Proof is given in Appendix C. The RSA-based KEM introduced in [23] can be seen as an instance of this method in the random oracle model.

4.2 Based on CCA-Secure KEM and MAC

In this section we present a CCA-secure Tag-KEM based on a CCA-secure KEM and a secure message authentication code (MAC) scheme (see Appendices A.1 and A.2 for formal definitions of these tools).

The idea is to encrypt a random key K using the KEM, and derive from K two keys dk, mk. The first, dk is the actual encrypted key, while mk is used to MAC the tag. The resulting MAC is appended to the ciphertext. A decryptor not only checks that the KEM decryption is correct, but also checks that the MAC on the tag, using the decrypted key mk is correct. A formal description follows.

Construction of Tag-KEM: Let $\Pi_L = (\mathsf{KEM.Gen}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$ be a KEM. Let $\mathsf{MAC} = (\mathsf{MAC.Sign}, \mathsf{MAC.Ver})$ be a MAC. Let $\mathsf{KDF}_2 : \mathcal{K}_K \to \mathcal{K}_D \times \mathcal{K}_M$ be a key derivation function where $\mathcal{K}_D$ is the key-space of DEM and $\mathcal{K}_M$ is the key-space of MAC. By using these components, we construct a Tag-KEM as follows. $\mathsf{TKEM.Gen}$ is the same as $\mathsf{PKE.Gen}$; It outputs (pk, sk).[1] $\mathsf{TKEM.Key}$ is that, given pk, it computes $(K, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}()$ and $(dk, mk) \leftarrow \mathsf{KDF}_2(K)$. Then it outputs dk and state information $\omega = (mk, \phi)$. The encryption and decryption functions are as follows.

Function: $\mathsf{TKEM.Enc}(\omega, \tau)$

$(mk, \phi) \leftarrow \omega$
$\sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\tau)$
Output $\psi = (\phi, \sigma)$

Function: $\mathsf{TKEM.Dec}_{sk}(\psi, \tau)$

$(\phi, \sigma) \leftarrow \psi$
$K \leftarrow \mathsf{KEM.Dec}_{sk}(\phi)$
$(dk, mk) \leftarrow \mathsf{KDF}_2(K)$
If $K = \perp$ or $\mathsf{MAC.Ver}_{mk}(\sigma, \tau) \neq 1$, output $\perp$.
Otherwise, output dk.

Clearly the CCA security of the KEM scheme will prevent an adversary from gaining any advantage by manipulating the KEM ciphertext. On the other hand the security of the MAC will prevent an adversary from gaining any advantage by manipulating the MAC.

Applying Theorem 1 to the above Tag-KEM yields the same hybrid encryption scheme as in Shoup's KEM/DEM framework. But by looking at that scheme in a different light, we are able to proceed a step further in refining the assumptions and the efficiency, as shown in the next section.

4.3 Based on weak KEM and MAC

In the previous scheme, there is some redundancy at play. If a KEM is combined with a MAC as shown in Section 4.2, the MAC will be used to preserve the integrity of ciphertexts. Accordingly, one may no longer need KEM's functionality of verifying ciphertexts. Following this intuition, in this Section we formally describe a new security notion of KEM that can be strictly weaker than CCA but sufficient to yield CCA-secure Tag-KEM when combined with MAC. We note, however, the result of this section might be regarded as theoretical. In practice, proving that a KEM conforms to the new notion could only be slightly easier than proving the security of resulting scheme as Tag-KEM. And one can expect better reduction cost by directly proving the security of Tag-KEM by exploiting specific properties.

Predicate-dependent CCA Security: Let Π_L be a KEM as in Section 4.2. Let $\mathcal{P} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be a poly-time computable predicate. Let $\mathcal{VD}$ be an extended decryption oracle that takes $(\phi, \eta) \in \{0,1\}^* \times \{0,1\}^*$ and outputs $\mathsf{KEM.Dec}_{sk}(\phi)$ if $\mathcal{P}(\mathsf{KEM.Dec}_{sk}(\phi), \eta) = 1$. Otherwise it outputs $\perp$. Let A_L be an adversary attacking Π_L. We define $\mathsf{GAME.LKEM}$

[1] If KDF_2 requires a key, it is generated in $\mathsf{TKEM.Gen}$ and included to pk and sk. See Appendix A.3 for details of KDF.

by modifying GAME.KEM so that decryption oracle receives ciphertexts accompanied by an arbitrary string and the decryption oracle $\mathsf{KEM.Dec}_{sk}(\phi)$ is replaced with $\mathcal{VD}$.

[GAME.LKEM]

Step 1. $(pk, sk) \leftarrow \mathsf{KEM.Gen}(1^\lambda)$
Step 2. $\upsilon \leftarrow A_L^{\mathcal{VD}(\cdot)}(pk)$
Step 3. $(K_1, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}()$, $K_0 \leftarrow \mathcal{K}_K$, $b \leftarrow \{0,1\}$.
Step 4. $\tilde{b} \leftarrow A_L^{\mathcal{VD}(\cdot)}(\upsilon, \phi, K_b)$

In Step 4, $\mathcal{VD}$ returns $\perp$ for any query such as $(\phi, *)$. We define $\epsilon_{\mathrm{lkem},A_L} = \left|\Pr[\tilde{b} = b] - \frac{1}{2}\right|$ and $\epsilon_{\mathrm{lkem}} = \max_{A_L}(\epsilon_{\mathrm{lkem},A_L})$ where maximum is taken over all machines. We say that a KEM is LCCA secure with respect to predicate $\mathcal{P}$ if ϵ_{lkem} is negligible in λ.

The above definition is too generic since a useful case is, as we shall see later, where MAC.Ver can be used as $\mathcal{P}$. Nevertheless, the generic treatment may be helpful to specify what property is really needed. Also in some cases, it makes the security analysis slightly simpler like shown in our analysis of Kurosawa-Desmedt scheme.

When does LCCA become weaker than CCA?: The strength of LCCA security depends on the property of $\mathcal{P}$. If $\mathcal{P}$ outputs 1 for any input, LCCA is clearly equivalent to CCA. If it outputs 0 for any input, LCCA is equivalent to an passive attack, i.e., attack without decryption oracle, for which we cannot prove the security of Tag-KEM in Section 4.2. We show that there exists a meaningful predicate for which LCCA becomes strictly weaker than CCA but for which we can prove the security of the resulting Tag-KEM.

For a valid public-key, say pk, generated by KEM.Gen, let VC be the set of all legitimately created ciphertexts, i.e., $VC = \{\phi \mid (K, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}()\}$. We say that ciphertext ϕ is valid if $\phi \in VC$. Otherwise, we say ϕ is invalid.

Let $\mathcal{P}$ be a predicate such that given pk, it is easy to find $(K, \phi, \eta) \in \mathcal{K}_K \times VC \times \{0,1\}^*$ such that $K = \mathsf{KEM.Dec}_{sk}(\phi)$ and $\mathcal{P}(K, \eta) = 1$. Consider the probability that A_L in GAME.LKEM makes a decryption query such that $\phi \notin VC$ and $\mathcal{P}(\mathsf{KEM.Dec}_{sk}(\phi_j), \eta) = 1$. We say KEM is *ciphertext sound* (CS) with respect to $\mathcal{P}$ if, for any ppt A_L, the probability is upper bound by a negligible function, $\epsilon_{\mathrm{sound}}$, in λ. Intuitively, $\mathcal{P}$ can be used to reject invalid ϕ.

We now claim that there exists a KEM that is LCCA and CS with respect to $\mathcal{P}$ but insecure in the sense of full CCA. A simple example would be a KEM that reveals the secret key when it receives $\phi \notin VC$. Accordingly, LCCA with respect to such a predicate for which CS holds is strictly weaker than CCA.

Proof of the Tag-KEM in Section 4.2: We prove the security of the construction when the underlying KEM is LCCA secure and ciphertext sound with respect to $\mathcal{P}^{\mathrm{mac}}$ defined as $\mathcal{P}^{\mathrm{mac}}(K, (\sigma, \tau)) = \mathsf{MAC.Ver}_{mk}(\sigma, \tau)$ where mk is $(dk, mk) \leftarrow \mathsf{KDF}_2(K)$. Note that, in the Tag-KEM construction shown in Section 4.2, $\mathcal{P}$ is not used in TKEM.Dec. Hence the underlying KEM Π_L itself might be insecure against CCA as mentioned above. However,

since $\mathcal{P}$ is assumed to be $\mathcal{P}^{\mathrm{mac}}$ and it is indeed provided from outside, LCCA security will be achieved. Namely, the MAC has two different role in the construction; one is to authenticate the tag and the other is to work as a predicate as a part of underlying KEM. As we could have predicted, this is very close to the combination of CCA KEM and MAC (but not exactly the same). Nevertheless, we have to formally prove the security to see the MAC plays the different roles without inconsistency.

Theorem 3. *If Π_L is LCCA secure with respect to $\mathcal{P}^{\mathrm{mac}}$ then the Tag-KEM defined in Section 4.2 is CCA secure. In particular, $\epsilon_{\mathrm{tkem}} \leq 4\epsilon_{\mathrm{lkem}} + q_D\,\epsilon_{\mathrm{mac}} + 5\epsilon_{\mathrm{kdf}}$ where q_D is the maximum number of decryption queries.*

Proof is in Appendix D. Note that $\epsilon_{\mathrm{sound}}$ is not included in the theorem because it is blended in ϵ_{lkem}. Indeed, CS is irrelevant to the theorem because such a notion is needed only to show that LCCA security can be weaker than CCA security. Still it is an important notion to characterize the strength of LCCA. We argue this issue again in Section 5.2.

We finally remark that one can also construct CCA-secure Tag-KEM from RCCA-secure KEM which is strictly weaker than CCA-secure ones. See Section 5.4 for further discussion.

4.4 Based on KEM with Hash function

We show another approach that might be available when your PKE does not have enough plaintext length as needed in Section 4.1 and/or increasing ciphertext length as in Section 4.2 is not acceptable.

If a KEM uses a hash function, probably for verifying ciphertexts, the KEM may be converted to a Tag-KEM simply by including the tag into the hash function input. This approach is correct if the hash function is involved in the scheme in a 'meaningful' way and provides 'sufficient' security. Although generic construction that follows formal version of these intuitive terms can be shown, it does not seem quite useful due to its complexity. Showing that a KEM fits to the generic framework may not be simpler than directly proving that the resulting Tag-KEM scheme is secure. Indeed, in all cases we have in mind, the security proof is essentially similar to the security proof of the original KEM (or PKE). Therefore, we only show two concrete constructions of Tag-KEM based on well known encryption schemes; OAEP+ [22] and Cramer-Shoup encryption [9]. In the following, the description of the original schemes are obtained just by dropping the tag τ.

From OAEP+. Let f be a one-way trapdoor permutation. OAEP+ encrypts dk with tag τ into ciphertext ψ in the following way:

$$r' = H'(r||dk||\tau),\ s = (G(r) \oplus dk)||r',\ w = H(s) \oplus r,\ \psi = f(s||w)$$

where r and r' are random and G, H, H' are random oracles [3].

Security is argued in the same way as the original one except the case that, for challenge ciphertext (ψ, τ) the adversary finds another valid ciphertext (ψ, τ'). Since ψ uniquely identifies r, r' and K, (ψ, τ') is valid only if $H'(r||dk||\tau) = H'(r||dk||\tau')$ holds. When H' outputs

a k_1-bit string, such an event happens with probability at most $q_{H'}\, 2^{-k_1}$ where $q_{H'}$ is the maximum number of queries to H'. Based on this observation, we define game GAME.0' where decryption oracle returns $\perp$ for all queries that differs only in the tag part with the challenge ciphertext. The rest of the security proof is done in the same way as in the original paper [22] except for obvious modifications. Accordingly, only $q_{H'}\, 2^{-k_1}$ is an extra reduction cost to that of OAEP+.

From Cramer-Shoup Encryption. A Tag-KEM scheme based on Cramer-Shoup encryption over a multiplicative group, say G_q, of prime order q is the following. A private-key is $(x_1, x_2, y_1, y_2, z_1, z_2) \in Z_q$ and the public-key is $g_1, g_2 \leftarrow G_q^2$, and $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, $h = g_1^{z_1} g_2^{z_2}$. The encryption function yields $dk = h^r$ where r is random, and ciphertext (u_1, u_2, v) such that

$$u_1 = g_1^r,\ u_2 = g_2^r,\ \alpha = H(u_1||u_2||\tau),\ v = c^r d^{\alpha r}$$

where H is a hash function. Decryption first checks if $v \stackrel{?}{=} u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$ and then recovers $dk = u_1^{z_1} u_2^{z_2}$. Applying Theorem 1 results in the hybrid PKE briefly mentioned in [9], which is not appeared in their full paper version [11].

In [11], the security of Cramer-Shoup encryption requires H to be Target Collision-Free since, without τ, all inputs to the hash function, i.e., u_1 and u_2, are chosen randomly by the encryption oracle. In this case, however, τ is chosen by the adversary after seeing the description of H. Hence we required a different condition on H.

We say that H is *Random Prefix Collision-Free* if any adversary wins the following game with at most negligible probability. The adversary is first given H and outputs τ and then given random x and finally outputs x' and τ' such that $H(x||\tau) = H(x'||\tau')$. A formal definition for this property is in Appendix B.

It holds that (Collision-Free) $\Rightarrow$ (Random Prefix Collision-Free) $\Rightarrow$ (Target Collision-Free). Hence it is reasonable to use cryptographic hash functions like SHA-1 which can be assumed collision-free. Nevertheless, we stress that random prefix collision-freeness may not necessarily be equivalent to collision-free because, for example, it is not clear how to perform a birthday attack in the above game (if the randomness of x affects to the output). Theoretically, we do not know constructions of random prefix collision-free hash functions from target collision-free or universal one-way hash functions, thus we resort to strong collision-freeness. The only drawback is that this requires a longer output (about twice as much because the birthday paradox applies here), but that does not affect our construction. This issue about the hash function did not arise in the original exposition of the hybrid version of Cramer-Shoup encryption in [9] since the hash function was considered as collision free rather than target collision free.

4.5 Based on ID-based PKE

An ID-based encryption scheme is selective-ID secure when it is secure against chosen ciphertext and chosen ID attacks provided that the target ID is committed at the beginning

and the ID must not be included in any decryption query. It is shown in [7] that selective-ID ID-based encryption schemes (sIBE in short) can be strengthened to a full CCA secure ones by using one-time signature. Then, according to the framework of Shoup, an ID-based hybrid encryption scheme can be obtained by combining it with a CCA secure DEM. We show that the conversion from sIBE to full IBE also yields a Tag-KEM. Accordingly, the DEM part can be simplified to be a one-time secure DEM. The resulting scheme yields shorter ciphertexts than before.

Let $(\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Ver})$ be a one-time signature scheme where $\mathsf{SIG.Gen}$ is a key generation algorithm, $\mathsf{SIG.Sign}$ is a signature generation algorithm, and $\mathsf{SIG.Ver}$ is a signature verification algorithm. Let $\mathsf{sIBE.Enc}(pk, \mathrm{ID}, m)$ be the encryption function of an sIBE. Then, we construct a Tag-KEM scheme as follows: It encrypts (pk, dk) and τ into ciphertext $\psi = (vk, \phi, \sigma)$ where

$$(vk, sk) \leftarrow \mathsf{SIG.Gen}(1^\lambda),\ \phi \leftarrow \mathsf{sIBE.Enc}(pk, vk, dk),\ \sigma = \mathsf{SIG.Sign}(sk, \phi||\tau).$$

Including τ into the message to be signed provides integrity to the tag without affecting the security of the original scheme. Indeed, the security proof is almost the same as in [7] with obvious modification. The reduction cost does not change, either. One can extend the above Tag-KEM to ID-based Tag-KEM in the same way starting from a 2nd-level ID Encryption function that takes two ID's. (A given ID is assigned to the first ID and vk is assigned to the second ID.) For efficient implementations of sIBE based on standard cryptographic assumptions, we refer to [25].

In [5], Boneh and Katz improved the efficiency of [7] by replacing the one-time signature with commitment scheme (using hash function) and MAC. Part of their scheme can also be seen as a Tag-KEM.

5 Applications

In this section, we show how our framework yields new hybrid encryption schemes, captures some known schemes, and even finds ways to improve them.

5.1 Threshold Hybrid PKE

Designing a threshold hybrid PKE is not a trivial task. Even though threshold PKE is available, it is not clear how it can be extended to hybrid threshold PKE. By following Shoup's KEM/DEM framework, one will suffer from sharing KDF and $\mathsf{MAC.Ver}$, which are often implemented by number-theoretically unstructured primitives. Although these tasks are feasible using generic techniques from multi-party computation, we are focusing on efficient and practical solutions.

Since our framework allows the DEM part to be CPA, it immediately yields a threshold hybrid PKE once a shared Tag-KEM is available. Decrypting the DEM part is a local task. By defining CCA security for threshold PKE and DEM as well as [24, 14], we can translate and prove Theorem 1 in the threshold setting. Accordingly, one can concentrate

on constructing threshold Tag-KEM. A threshold KEM or PKE can be converted into a threshold Tag-KEM by following the construction in Section 4.4 or 4.1 without considerable overheads.

Threshold Cramer-Shoup encryption, secure against static adversaries, is shown in [1, 6], and the conversion technique in Section 4.4 (or result of section 4.1 with larger security parameter) can be used to obtain a threshold Cramer-Shoup Tag-KEM. Accordingly, by following the threshold version of Theorem 1, one can have a secure threshold hybrid encryption scheme in the standard model. Adaptive security can be achieved as well based on the adaptively secure threshold Cramer-Shoup encryption of [2].

5.2 Revisiting the Kurosawa-Desmedt Scheme

In [17], Kurosawa and Desmedt introduced a hybrid encryption scheme based on Cramer-Shoup encryption. The private-key $sk = (x_1, x_2, y_1, y_2)$ and public-key $pk = (g_1, g_2, c, d)$ are a part of that for Cramer-Shoup encryption as shown in Section 4.4. Encryption of message $m \in \{0,1\}^*$ is :

$$u_1 = g_1^r,\ u_2 = g_2^r,\ \alpha = H(u_1||u_2),\ v = c^r d^{\alpha r},\ (dk, mk) \leftarrow \mathsf{KDF}_2(v),$$

$$\chi = G(dk) \oplus m,\ \sigma = \mathsf{MAC.Sign}_{mk}(\chi),$$

where r is random, H is a target collision-free hash function, G is a pseudo-random bit generator, and $\mathsf{MAC.Sign}$ is a MAC generation function. The ciphertext is (u_1, u_2, χ, σ). In this scheme, (u_1, u_2) is considered as the KEM part and (χ, σ) is considered as the CCA-secure DEM part. Though the combination results in a CCA-secure hybrid PKE, it is not known whether the KEM part is CCA or not and what security is really needed for such a KEM if it is not CCA.

Our framework reveals another approach to the analysis of the scheme. That is, we consider (u_1, u_2, σ) as Tag-KEM part and χ as one-time secure DEM part. The Tag-KEM part is further decomposed to KEM part, (u_1, u_2) and MAC, (σ), according to the framework of Section 4.3. We can then see that (u_1, u_2) constitutes a KEM that is LCCA secure and ciphertext sound with regard to a predicate $\mathcal{P}^{\mathrm{mac}}(K = v, \eta = (\chi, \sigma))$. See Appendix E for proof. Accordingly, the Kurosawa-Desmedt scheme can be thoroughly explained by our framework and their design approach is validated.

We note that it is not necessary to show that the KD scheme is ciphertext sound. However, showing that property implies that the scheme belongs to a weaker class of KEM. This is important because it is not known if the scheme belongs to the class of CCA KEM.

5.3 Refined Fujisaki-Okamoto Conversion and More

Fujisaki-Okamoto Conversion: We revisit the Fujisaki-Okamoto conversion [15] that provides secure construction of hybrid encryption in the random oracle model. By fitting their scheme into our framework, we can see that one of their assumptions can be eliminated and a refined version is obtained without loss of efficiency.

Let $\mathsf{PKE.Enc}_{pk}(\cdot;\cdot)$ be public-key encryption function where the last argument denotes a random coin used in the function. Fujisaki-Okamoto conversion combines PKE and DEM by using two random oracles, H and G, as follows:

$$\psi \leftarrow \mathsf{PKE.Enc}_{pk}(K; H(K||m)), \chi \leftarrow \mathsf{DEM.Enc}_{G(K)}(m).$$

A ciphertext is (ψ, χ). The resulting hybrid PKE is CCA-secure if PKE is one-way and DEM is one-time secure and $\mathsf{DEM.Enc}$ is a bijection between ciphertexts and messages for every fixed key.

Now one can observe that $\mathsf{PKE.Enc}_{pk}(K; H(K||\tau))$ works as a Tag-KEM encryption function that encapsulates DEM key $G(K)$. Then, according to our framework, we have slightly modified hybrid encryption:

$$\psi \leftarrow \mathsf{PKE.Enc}_{pk}(K; H(K||\chi)), \chi \leftarrow \mathsf{DEM.Enc}_{G(K)}(m)$$

which does not require $\mathsf{DEM.Enc}$ to be a bijection. Details are given in Appendix F.

Bellare-Rogaway Scheme: The scheme shown by Bellare and Rogaway in [3] is a special case of Fujisaki-Okamoto construction. The encryption function consists of one-way permutation f and random oracles H and G;

$$\psi = f(r), \sigma = H(r||m), \chi = G(r) \oplus m$$

This scheme is dedicated for one-time pad for DEM part. According to our framework, we can generalize to any one-time secure DEM by modifying the scheme as

$$\psi = f(r), \sigma = H(r||\chi), \chi = \mathsf{DEM.Enc}_{G(r)}(m).$$

REACT: REACT-RSA [19] is very similar to the above Bellare-Rogaway scheme;

$$\psi = f(r), \sigma = H(r||m||\psi||\chi), \chi = \mathsf{DEM.Enc}_{G(r)}(m),$$

where f is the RSA encryption function. In this case, our framework shows that m can be removed from the inputs to H. Including m to H would results in slightly better reduction in the security proof. But removing it yields more benefit when m is long enough. Even ψ can be removed if decryption function verifies that ψ is in the correct domain. In the case of RSA, domain checking is done just by comparison with the modulus. Hence by modifying to $\sigma = H(r||\chi)$, efficiency increases. Indeed, the resulting scheme is the same scheme as above.

The common factor lying underneath the above-mentioned examples is the Tag-KEM scheme whose ciphertext is $\psi = (f(r), H(r||\tau))$ where H is a random oracle.

Some ISO standard candidates: Finally, as mentioned in Section 4.1, KEM schemes based on RSA and HIME described in [23] allow to label each ciphertext. This label can be used as a tag in our framework. Hence the DEM no longer need to provide CCA security when combined with those KEM according to our framework.

5.4 Revisiting RCCA-secure PKE

This section revisits RCCA-secure PKE in [8] and show that their construction of CCA-secure hybrid PKE from RCCA-secure PKE can be improved by following our Tag-KEM/DEM framework.

The notion of RCCA-secure PKE is introduced in [8]. RCCA is a variant of CCA where the decryption oracle returns a special nonce 'test' when it receives a ciphertext that yields one of the questioned message, m_0 and m_1. Accordingly, even if the adversary can tweak the challenge ciphertext without affecting the embedded plaintext (such a feature is called benign-malleability [23]), sending it to the decryption oracle will give no advantage to the adversary in determining which of the questioned messages is hidden there. 'R' stands for 'replayable' in this sense. RCCA-security is a strict relaxation of CCA-security and proven useful for several cryptographic tasks, though, currently, there is no known instance of RCCA-secure PKE that is more efficient than known CCA-secure ones.

In [8], it is shown that combining RCCA-secure PKE and CCA-secure symmetric encryption can yield CCA-secure hybrid PKE. Suppose that a CCA-secure symmetric encryption is made by combining passively secure DEM and one-time MAC. Then, their construction is summarized as follows. Given message m, output ciphertext (ϕ, χ, σ) such that;

$$\phi \leftarrow \mathsf{PKE.Enc}_{pk}(dk||mk),\ \chi \leftarrow \mathsf{DEM.Enc}_{dk}(m||\phi),\ \sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\chi)$$

where dk and mk, are chosen randomly from appropriate domains. It is stressed that ϕ is encrypted by DEM and this double-encryption structure is essential in their security proof. Due to this special structure, the construction does not fit to our framework. Below, we show a slightly more efficient variant that avoids double encryption and fits to our framework.

$$\phi \leftarrow \mathsf{PKE.Enc}_{pk}(dk||mk),\ \chi \leftarrow \mathsf{DEM.Enc}_{dk}(m),\ \sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\chi||\phi)$$

Intuitively, applying MAC to ϕ offsets the benign-malleability of ϕ. The modified scheme yields shorter ciphertext and needs less computation.

From the above, we derive a Tag-KEM scheme which is summarized as follows.

$$(K, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}(), \quad (dk, mk) \leftarrow KDF_2(K), \quad \sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\tau||\phi)$$

It can be seen as a variant of the construction shown in Section 4.2; MAC is applied to $\tau||\phi$ rather than to τ. In Appendix G, we give definition of RCCA-security for KEM, which is an analogue notion of that for PKE, and prove that the above Tag-KEM is CCA-secure if KEM is RCCA-secure. Hence, according to Theorem 1, the modified hybrid PKE is CCA-secure. This uncovers the superfluousness of the double-encryption in the original construction and obtains a more efficient scheme.

Acknowledgments

The authors would like to thank Hugo Krawczyk, Shai Halevi, Yevgeniy Dodis, Victor Shoup and Eiichiro Fujisaki for valuable discussion.

References

1. M. Abe. Robust distributed multiplication without interaction. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 130–147. Springer-Verlag, 1999.
2. M. Abe and S. Fehr. Adaptively secure feldman VSS and applications to universally-composable threshold cryptography. Cryptology ePrint Archive, Report 2004/119, 2004.
3. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communication Security*, pages 62–73. Association for Computing Machinery, 1993.
4. D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1998.
5. D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. Technical Report 2004/261, IACR ePrint archive, 2004.
6. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 90–106. Springer-Verlag, 1999.
7. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer-Verlag, 2004.
8. R. Canetti, H. Krawczyk, and J. Nielsen. Relaxing chosen-ciphertext security. Technical Report 2003/174, IACR ePrint archive, 2003.
9. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998.
10. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. Knudsen, editor, *Advances in Cryptology—EUROCRYPTO '02*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer-Verlag, 2002.
11. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
12. Y. Dodis, R. Gennaro, J. Haastad, H. Krawczyk, and T. Rabin. Randomness extraction and key derivation using the CBC, Cascade and HMAC modes. In Matt Franklin, editor, *Advances in Cryptology — CRYPTO '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 494–510. Springer-Verlag, 2004.
13. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of the* 23*rd annual ACM Symposium on the Theory of Computing*, pages 542–552, New York City, 1991.
14. P. Fouque and D. Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In C. Boyd, editor, *Advances in Cryptology – Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 351–368. Springer-Verlag, 2001.
15. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer-Verlag, 1999.
16. R. Gennaro and V. Shoup. A note on an encryption scheme of Kurosawa and Desmedt. Technical Report 2004/194, IACR ePrint archive, 2004.

17. K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In Matt Franklin, editor, *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer-Verlag, 2004.
18. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the* 22*nd annual ACM Symposium on the Theory of Computing*, pages 427–437, 1990.
19. T. Okamoto and D. Pointcheval. REACT: Rapid enhanced-security asymmetric cryptosystem transform. In *RSA '2001*, Lecture Notes in Computer Science. Springer-Verlag, 2001.
20. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology — CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992.
21. V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 275–288. Springer-Verlag, 2000.
22. V. Shoup. OAEP reconsidered. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *LNCS*, pages 239–259. Springer-Verlag, 2001.
23. V. Shoup. ISO 18033-2: An emerging standard for public-key encryption (committee draft). Available at `http://shoup.net/iso/`, June 3 2004.
24. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1998.
25. B. Waters. Efficient identity-based encryption without random oracles. Technical Report 2004/180, IACR ePrint archive, 2004.

Appendices

A More Building Blocks

A.1 Key Encapsulation Mechanism

This section describes syntax and security definitions for KEM which is the same as the one shown by Shoup.

$(pk, sk) \leftarrow \mathsf{KEM.Gen}(1^\lambda)$ A probabilistic algorithm that generates public and private keys (pk, sk). The public-key defines the key space $\mathcal{K}_K$.

$(K, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}()$ A probabilistic algorithm that generates key $K \in \mathcal{K}_K$ and its encryption ϕ.

$K \leftarrow \mathsf{KEM.Dec}_{sk}(\phi)$ An algorithm that decrypts ϕ to recover K. As well as PKE, an obvious soundness condition applies. It may output a special symbol $\perp \notin \mathcal{K}_K$.

Since we use KEM only as a component to construct Tag-KEM in this paper, we consider $\mathsf{KEM.Enc}$ that outputs $K \in \mathcal{K}_K$ for some specific domain $\mathcal{K}_K$ rather than the ones adjusted to DEM key-space.

Let $\mathcal{O}$ denote decryption oracle, $\mathsf{KEM.Dec}_{sk}(\cdot)$. Let A be a polynomial-time oracle machine that plays the following game.

[GAME.KEM]

Step 1. $(pk, sk) \leftarrow \mathsf{KEM.Gen}(1^\lambda)$
Step 2. $\upsilon \leftarrow A^{\mathcal{O}}(pk)$
Step 3. $(K_1, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}()$, $K_0 \leftarrow \mathcal{K}_K$, $b \leftarrow \{0,1\}$.
Step 4. $\tilde{b} \leftarrow A^{\mathcal{O}}(\upsilon, \phi, K_b)$.

In Step 4, A is restricted not to ask ϕ to $\mathsf{KEM.Dec}$.

We define $\epsilon_{\mathrm{kem},A} = \left|\Pr[\tilde{b} = b] - \frac{1}{2}\right|$ and $\epsilon_{\mathrm{kem}} = \max_A(\epsilon_{\mathrm{kem},A})$ where maximum is taken over all machines. We say that a KEM is secure against adaptive chosen ciphertext attacks (CCA secure) if ϵ_{kem} is negligible in λ.

A.2 Message Authentication Code

MAC is a pair of algorithms $(\mathsf{MAC.Sign}, \mathsf{MAC.Ver})$ and a key-space $\mathcal{K}_M$ defined by security parameter λ. Typically, $\mathcal{K}_M = \{0,1\}^\lambda$. $\mathsf{MAC.Sign}$ takes a mac-key $mk \in \mathcal{K}_M$ and a message $\tau \in \{0,1\}^*$ and outputs a string σ. We say (σ, τ) is valid with regard to mk if $\sigma = \mathsf{MAC.Sign}_{mk}(\tau)$. $\mathsf{MAC.Ver}$ takes a triple (mk, σ, τ) and outputs 1 if (σ, τ) is valid with respect to mk, or outputs 0, otherwise.

We require MAC to be secure against one-time chosen message attack; an adversary chooses arbitrary message τ and given its MAC σ created with MAC key mk randomly chosen from $\mathcal{K}_M$, and outputs (σ', τ') which is different from (σ, τ) as a pair. The adversary wins if the resulting pair is correct with respect to the same mk. Let $\mathsf{GAME.MAC}$ denote this attack game. We say that MAC is secure against one-time chosen message attack if any ppt adversary can win $\mathsf{GAME.MAC}$ at most with negligible probability, say ϵ_{mac}.

A.3 Key Derivation Function

Our construction uses a function that maps a key chosen by KEM into a pair of keys used for DEM and MAC. We require its output distribution is indistinguishable from uniform.

Since the input domain depends on a specific choice of KEM, DEM, and MAC, KDF is defined with regard to the key-space of these schemes. Let $\mathsf{KDF}_2 : \mathcal{K}_K \to \mathcal{K}_D \times \mathcal{K}_M$ and $\{\mathsf{KDF}_2\}_\lambda$ be a family of functions indexed by the key-spaces associated to the same security parameter λ. (Extra keys may also be used as index if needed.) We require that distribution of KDF_2 is indistinguishable from uniform over $\mathcal{K}_D \times \mathcal{K}_M$. Let

$$D_1 = \{(dk, mk) \,|\, K \leftarrow \mathcal{K}_K, (dk, mk) \leftarrow \mathsf{KDF}_2(K)\}, \text{ and}$$
$$D_0 = \{(dk, mk) \,|\, (dk, mk) \leftarrow \mathcal{K}_D \times \mathcal{K}_M\}$$

We say that KDF_2 is secure if, for polynomial time machine A_{KDF},

$$|\Pr[b \leftarrow \{0,1\}, (dk, mk) \leftarrow D_b, \tilde{b} \leftarrow A_{\mathsf{KDF}}(pk, \mathsf{KDF}_2, (dk, mk)) \,;\, \tilde{b} = b] - \frac{1}{2}| \leq \epsilon_{\mathrm{kdf}}$$

where ϵ_{kdf} is a negligible function in λ. The probability is taken over the choice of KDF_2 which includes coins of $\mathsf{KEM.Gen}$ that determines $\mathcal{K}_K$ and the choice of (dk, mk), b, and the coins of A_{KDF}.

For construction issue, we refer Section 8 of [11]. It can be constructed without additional assumption but losing efficiency. Or, for efficiency, we simply assume that a cryptographic hash function like SHA-1 suffices. Some more constructions based on other cryptographic primitives are found in [12].

In our construction of Tag-KEM, we require dk and mk derived from independent application of KDF_2 to two different inputs to be indistinguishable from the legitimately generated ones. Let

$$U_1 = D_1, \text{and}$$
$$U_0 = \{(dk, mk) | K \leftarrow \mathcal{K}_K, (dk, *) \leftarrow \mathsf{KDF}_2(K), K' \leftarrow \mathcal{K}_K, (*, mk) \leftarrow \mathsf{KDF}_2(K')\}.$$

Lemma 1. *If KDF_2 is secure, for all polynomial machine A,*

$$\left| \Pr[d \leftarrow \{0,1\}, (dk, mk) \leftarrow U_d, \tilde{d} \leftarrow A(dk, mk)\,;\, \tilde{d} = d] - \frac{1}{2} \right| \leq 2\epsilon_{\mathrm{kdf}}.$$

Proof. Consider a hybrid distribution

$$W = \{(dk, mk) | K \leftarrow \mathcal{K}_K, (dk, *) \leftarrow \mathsf{KDF}_2(K), mk \leftarrow \mathcal{K}_M\}.$$

By hybrid argument, the advantage of distinguishing D_1 and W is upper bound by ϵ_{kdf}. Let ϵ'_{kdf} be advantage of distinguishing U_1 and U_0. Again by hybrid argument, the advantage of distinguishing U_0 and W is at least $\epsilon'_{\mathrm{kdf}} - \epsilon_{\mathrm{kdf}}$. Then, given a machine that distinguishes U_0 and W with that probability, one can easily construct a machine that distinguishes D_0 and D_1 with the same probability. Hence $\epsilon'_{\mathrm{kdf}} - \epsilon_{\mathrm{kdf}} \leq \epsilon_{\mathrm{kdf}}$. This completes the proof.

By simple computation, we have:

Corollary 1. *If KDF_2 is secure, for all polynomial machine A,*

$$|\Pr[(dk, mk) \leftarrow U_0, 1 \leftarrow A(dk, mk)] - \Pr[(dk, mk) \leftarrow U_1, 1 \leftarrow A(dk, mk)]| \leq 4\epsilon_{\mathrm{kdf}}.$$

B Target Collision-Free and Random Prefix Collision-Free

Target Collision-Free (TCH): Target Collision-Free is a special case of universal one-way; An adversary is given (H, x) and then attempts to find x' such that $H(x) = H(x')$. Let $\mathcal{X}_\lambda = \{X\}$ be a collection of domains and $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$. Let $\mathcal{H}_\lambda = \{H : X \to \{0,1\}^\lambda \mid X \in \mathcal{X}_\lambda\}$ and $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda \in \mathbb{N}}$. Note that X is identified by the description of H. Let A_{TCH} be a machine playing the following game.

[GAME.TCH]

Step 1. $H \leftarrow \mathcal{H}_\lambda$, $x \leftarrow X$.
Step 2. $x' \leftarrow A_{TCH}(H, x)$ such that $x' \in X$.

A_{TCH} wins if $H(x') = H(x)$. We define $\epsilon_{\mathrm{tch}, A_{TCH}} = \Pr[H(x') = H(x)]$ in GAME.TCH and $\epsilon_{\mathrm{tch}} = \max_{A_{TCH}}(\epsilon_{\mathrm{tch}, A_{TCH}})$ where maximum is taken over all machines. We say that $\mathcal{H}$ is

target collision-free with regard to $\mathcal{X}$ if ϵ_{tch} is negligible in λ. For simplicity, we also say that $\mathcal{H}_\lambda$ (or even H) is target collision-free with regard to $\mathcal{X}_\lambda$ (or X, respectively).

Random Prefix Collision-Free (RPH): Random prefix collision-free is a notion in between target collision-free and collision-free; An adversary is first given H and finds x and then given random r and outputs r' and x' such that $H(r,x) = H(r',x')$. Let $\mathcal{X}_\lambda = \{X\}$ be a collection of domains and $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$. We define $\mathcal{R}_\lambda$ and $\mathcal{R}$ in the same way. Then, let $\mathcal{H}_\lambda = \{H : X \times R \to \{0,1\}^\lambda \mid X \in \mathcal{X}_\lambda, R \in \mathcal{R}_\lambda\}$ and $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$. Let A_{RPH} be a machine playing the following game named GAME.RPH.

[GAME.RPH]

Step 1. $H \leftarrow \mathcal{H}_\lambda$
Step 2. $(\upsilon, x) \leftarrow A_{RPH}(H)$
Step 3. $r \leftarrow R$
Step 4. $(r', x') \leftarrow A_{RPH}(\upsilon, r)$ such that $r' \in R$ and $x' \in X$.

A_{TCH} wins if $H(r',x') = H(r,x)$ and $(r',x') \neq (r,x)$. We define $\epsilon_{\mathrm{rph},A_{TCH}}$, ϵ_{tch} for GAME.RPH as well and say that $\mathcal{H}$ is random prefix collision-free with regard to $\mathcal{X}$ and $\mathcal{R}$ if ϵ_{rph} is negligible in λ.

C Proof of Theorem 2

Let A_T be an adversary against the Tag-KEM presented in Section 4.1. By using A_T, we construct A_E that plays GAME.PKE to attack the underlying PKE as follows.

1. Given pk, A_E sends it to A_T.
2. For the encryption query from A_T, A_E does the following.
   (a) Choose $(dk_0, dk_1) \leftarrow \mathcal{K}_K \times \mathcal{K}_K$, $\delta \leftarrow \{0,1\}$ and give dk_δ to A_T.
   (b) Given τ from A_T, send $(dk_0||\mathrm{TH}(\tau), dk_1||\mathrm{TH}(\tau))$ to the encryption oracle of PKE.
   (c) Given ψ from the encryption oracle, forward it to A_T.
3. For every decryption query (τ_i, ψ_i) from A_T, A_E does the following. If $\psi_i = \psi$, return $\perp$ to A_T. Otherwise, send ψ_i to the decryption oracle of PKE. Given $dk_i||\tau_i'$ from the decryption oracle, if $\tau_i \neq \tau_i$, return $\perp$ to A_T. Otherwise, return dk_i.
4. When A_T outputs $\tilde{\delta} = \delta$, A_E outputs 1. Otherwise, output 0.

Let Col denote an event that $\mathrm{TH}(\tau) = \mathrm{TH}(\tau')$ happens. Since the target ciphertext ψ is uniquely decrypted to $dk_\delta||\tau$, any (τ', ψ) other than (τ, ψ) cannot be a valid ciphertext of Tag-KEM unless Col takes place. Hence $\perp$ is a correct answer to any decryption query with $\psi_i = \psi$. All other decryption queries from A_T are correctly answered since ψ_i is correctly decrypted by the decryption oracle of PKE. It is also easy to see that encryption oracle is simulated as given.

Now if $\tilde{\delta} = \delta$, it means dk_δ is embedded in ψ_i. On the other hand, if $\tilde{\delta} \neq \delta$, dk_δ is independent of ψ, i.e., $dk_{1-\delta}$ is embedded in ψ. Thus, A_T, A_E wins in GAME.PKE with the

same probability as A_T wins in GAME.TKEM in the case of $\neg$Col. Thus we have;

$$\begin{aligned}\Pr[A_T \text{ wins}|\neg\mathsf{Col}] &= \Pr[A_E \text{ wins}]\\ \Pr[A_T \text{ wins}] &\leq \Pr[A_E \text{ wins}] + Pr[\mathsf{Col}]\\ \epsilon_{\text{tkem}} &\leq \epsilon_{\text{pke}} + \epsilon_{\text{tch}},\end{aligned}$$

where ϵ_{tch} is the probability of breaking TH as defined in Appendix B.

D Proof of Theorem 3

The following is the CCA attack against the Tag-KEM in Section 4.3. Let $\mathcal{O}$ denote decryption oracle, $\mathsf{TKEM.Dec}_{sk}(\cdot,\cdot)$. Let GAME.0 denote this CCA game.

[GAME.0]

Step 1. $(pk, sk) \leftarrow \mathsf{KEM.Gen}(1^\lambda)$
Step 2. $\upsilon_1 \leftarrow {A_T}^{\mathcal{O}}(pk)$
Step 3. $(K_1, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}()$, $\delta \leftarrow \{0,1\}$. If $\delta = 1$, set dk by $(dk, *) \leftarrow \mathsf{KDF}_2(K_1)$. Otherwise, $dk \leftarrow \mathcal{K}_D$.
Step 4. $(\tau, \upsilon_2) \leftarrow {A_T}^{\mathcal{O}}(\upsilon_1, dk)$
Step 5. $(*, mk) \leftarrow \mathsf{KDF}_2(K_1)$, $\sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\tau)$.
Step 6. $\tilde{\delta} \leftarrow {A_T}^{\mathcal{O}}(\upsilon_2, (\phi, \sigma))$

In Step 6, A_T is restricted not to send $((\phi, \sigma), \tau)$ to the decryption oracle. It is important to see that the MAC is always created with the correct mac-key embedded in ϕ regardless of δ.

The outline of our proof is the same as that of [11]; Defining a series of games, GAME.1, $\cdots$, GAME.3 by modifying GAME.0 and examining fluctuation of probability $\Pr[X_0]$, $\cdots$, $\Pr[X_3]$ where X_i denotes the event that $\tilde{\delta} = \delta$ in GAME.i. Our final goal is to upper limit $\Pr[X_0]$.

The following is the description of each game. Every claim is proven formally after all outline is shown.

GAME.1: We modify the encryption oracle in such a way that when $\delta = 0$, dk is chosen by $K_0 \leftarrow \mathcal{K}_K$, $(dk, *) \leftarrow \mathsf{KDF}_2(K_0)$. It is straightforward to show that

$$|\Pr[X_0] - \Pr[X_1]| = \frac{1}{2}|\Pr[X_0 \,|\, \delta = 0] - \Pr[X_1 \,|\, \delta = 0]| \leq \epsilon_{\text{kdf}}. \tag{1}$$

GAME.2: We modify GAME.1 in such a way that the decryption oracle returns $\perp$ to all queries containing $\phi_j = \phi$. We claim that

$$|\Pr[X_1] - \Pr[X_2]| \leq 2\epsilon_{\text{lkem}} + q_D\epsilon_{\text{mac}} + 2\epsilon_{\text{kdf}}. \tag{2}$$

GAME.3: We modify GAME.2 in such a way that the encryption oracle computes the MAC σ by using a key derived from K_δ instead of the legitimate key K_1. Namely, we replace $\mathsf{KDF}_2(K_1)$ in Step 5 with $\mathsf{KDF}_2(K_\delta)$.

We claim that

$$|\Pr[X_2] - \Pr[X_3]| \leq \epsilon_{\mathrm{lkem}} + 2\epsilon_{\mathrm{kdf}}, \tag{3}$$

and

$$|\Pr[X_3] - \frac{1}{2}| \leq \epsilon_{\mathrm{lkem}}. \tag{4}$$

Conclusion: From (1), (2), (3), and (4), we have

$$|\Pr[X_0] - \frac{1}{2}| \leq 4\epsilon_{\mathrm{lkem}} + q_D\,\epsilon_{\mathrm{mac}} + 5\epsilon_{\mathrm{kdf}}$$

as stated in the theorem.

Proof of (2): Let F_2 be an event that the adversary creates at least one query that is rejected in $\mathsf{GAME.2}$ but accepted in $\mathsf{GAME.1}$. Unless event F_2 happens, the view of the adversary is identical in both games. Hence we have

$$|\Pr[X_1] - \Pr[X_2]| \leq \Pr[F_2]. \tag{5}$$

We first consider F_2 in the case of $\delta = 1$. Consider the simulation conducted by machine A_L that launches LCCA attack to KEM by using A_T as follows.

Step 1. Forward given pk to A_T.
Step 2. Simulate the encryption oracle as follows.
- Given request from A_T, make a challenge request to the encryption oracle of $\mathsf{GAME.LKEM}$ and receive (ϕ, K_b). Then, compute $(dk, mk) \leftarrow \mathsf{KDF}_2(K_b)$ and return dk.
- Given τ from A_T, compute $\sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\tau)$ and return (ψ, σ).

Step 3. For every decryption query $(\phi_j, \sigma_j, \tau_j)$, simulate the decryption oracle as follows. If $\phi_j \neq \phi$, send ϕ_j and $\eta_j = (\sigma_j, \tau_j)$ to the decryption oracle of $\mathsf{GAME.LKEM}$ and receive K_j. (If $\perp$ is returned, forward it to A_T.) Then simulate the decryption oracle of $\mathsf{GAME.2}$ as prescribed by using K_j. If $\phi_j = \phi$, compute $\mathsf{MAC.Ver}_{mk}(\sigma_j, \tau_j)$. If it is 1, output $\tilde{b} = 1$ and halt. Otherwise, return $\perp$ to A_T and continue simulation.
Step 4. When A_T stops, output $\tilde{b} = 0$.

Remember that K_b is either random ($b = 0$) or correct ($b = 1$) with regard to ϕ. The intuition is that when $b = 1$, the decryption oracle checks every MAC with the correct key. Hence it captures event F_2 and $\tilde{b} = 1$ occurs. On the other hand, when $b = 0$, every MAC σ_j is verified with a random key associated only to the MAC σ attached to the challenge ciphertext. Therefore, unless MAC is forged, $\tilde{b} = 1$ does not happen. If, the probability that event $\tilde{b} = 1$ occurs is meaningfully different, it contradicts to the LCCA security. Now, we consider two cases, $b = 1$ and $b = 0$, in detail.

1. When $b = 1$ in GAME.LKEM, both dk and σ are made from correct key embedded in ψ. This simulates the encryption oracle of GAME.2 at $\delta = 1$. The decryption oracle is simulated as given until event F_2 happens. A_T can correctly capture the event when it happens since mk is correct. Therefore, $\Pr[F_2 \,|\, \delta = 1] = \Pr[\tilde{b} = 1 \,|\, b = 1]$.
2. When $b = 0$, we claim $\Pr[\tilde{b} = 1 \,|\, b = 0] \leq q_D\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}$. Hereafter, we only consider the case $b = 0$ where dk and mk are independent of ψ. Let F_2' be the event that $\tilde{b} = 1$ happens in the environment of $b = 0$. We consider a series of auxiliary games as follows. First modify the encryption oracle in GAME.2 in such a way that dk and mk are chosen just randomly, i.e., $(dk, mk) \leftarrow \mathcal{K}_D \times \mathcal{K}_M$. Let F_2'' be the event of $\tilde{b} = 1$ in this game. It is easy to see that $|\Pr[F_2'] - \Pr[F_2'']| \leq 2\epsilon_{\mathrm{kdf}}$. Since $\delta = 0$, the input to KDF_2 is random and independent from all other views, the first claim is proven by showing a straightforward reduction from A_T to A_{KDF}. We also claim that $\Pr[F_2''] \leq q_D\epsilon_{\mathrm{mac}}$. It is also easy to prove since mk is randomly and independently chosen from $\mathcal{K}_M$ in the modified game. Factor of q_D is due to the fact that the reduction algorithm has to correctly grab the first occurrence of F_2'' without knowing its MAC key.

Since $|\Pr[\tilde{b} = 1 \,|\, b = 1] - \Pr[\tilde{b} = 1 \,|\, b = 0]| \leq 2\epsilon_{\mathrm{lkem}}$, we have

$$\Pr[F_2 \,|\, \delta = 1] \leq 2\epsilon_{\mathrm{lkem}} + q_D\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}. \tag{6}$$

We next consider F_2 in the case of $\delta = 0$ where dk is always made from random K_0. Consider the simulation conducted by machine A_L that launches LCCA attack to KEM by using A_T almost in the same way as above except that the encryption oracle computes dk by $(dk, *) \leftarrow \mathsf{KDF}_2(K_0)$ where K_0 is chosen randomly from $\mathcal{K}_K$. This means that dk distributes independently from all other variables. Accordingly, when $b = 1$, the view of A_T in the simulation is identical to that in GAME.2 at $\delta = 0$. Hence we have $\Pr[F_2 \,|\, \delta = 0] = \Pr[\tilde{b} = 1 \,|\, b = 1]$. Furthermore, one can show that $\Pr[\tilde{b} = 1 \,|\, b = 0] \leq q_D\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}$ in exactly the same way as above. Accordingly, we have

$$\Pr[F_2 \,|\, \delta = 0] \leq 2\epsilon_{\mathrm{lkem}} + q_D\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}. \tag{7}$$

From (6) and (7), we have the following as claimed.

$$\begin{aligned}\Pr[F_2] &= \frac{1}{2}\left(\Pr[F_2 \,|\, \delta = 0] + \Pr[F_2 \,|\, \delta = 1]\right) \\ &\leq 2\epsilon_{\mathrm{lkem}} + q_D\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}\end{aligned}$$

Proof of (3): First of all, observe that, when $\delta = 1$ in GAME.2 and GAME.3, the views of A_T are identical since there is no difference between the games. Therefore,

$$\begin{aligned}|\Pr[X_2] - \Pr[X_3]| &= \frac{1}{2}|\Pr[X_2 \,|\, \delta = 1] - \Pr[X_3 \,|\, \delta = 1] + \Pr[X_2 \,|\, \delta = 0] - \Pr[X_3 \,|\, \delta = 0]| \\ &= \frac{1}{2}|\Pr[X_2 \,|\, \delta = 0] - \Pr[X_3 \,|\, \delta = 0]|\end{aligned}$$

Accordingly, we only need to consider the case where $\delta = 0$, i.e., dk is always made from random K_0.

We construct machine A_L, that launches LCCA attack to KEM by using A_T. A_L works as follows.

Step 1. Forward given pk to A_T.
Step 2. Simulate the encryption oracle as follows.
- Given request from A_T, compute $K'_0 \leftarrow \mathcal{K}_K$, $(dk, *) \leftarrow \mathsf{KDF}_2(K'_0)$ and return dk. Then, make a challenge request to the encryption oracle of GAME.LKEM and receive (ϕ, K_b).
- Given τ from A_T, generate σ by using K_b as $(*, mk) \leftarrow \mathsf{KDF}_2(K_b)$, $\sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\tau)$. Then return (ϕ, σ).

Step 3. For every decryption query $(\phi_j, \sigma_j, \tau_j)$, simulate the decryption oracle as follows. If $\phi_j = \phi$, return $\perp$. Otherwise, send $(\phi_j, (\sigma_j, \tau_j))$ to the decryption oracle of GAME.LKEM and receive K_j. Then simulate the decryption oracle as prescribed by using K_j.
Step 4. When A_T outputs $\tilde{\delta}$, output it as $\tilde{b}$.

It is important to see that dk is independent of ϕ regardless of b. Also remember that K_b is either random ($b = 0$) or correct ($b = 1$) with respect to ϕ. Now, by inspection, one can see that the following holds.

1. When $b = 1$ in GAME.LKEM, σ is made from correct key embedded in ϕ. The decryption oracle is simulated as given. Hence the view of A_T in the simulation by A_K is identical to that in GAME.2 at $\delta = 0$, i.e., $\Pr[X_2 \,|\, \delta = 0] = \Pr[\tilde{b} = 0 \,|\, b = 1]$.
2. Similarly, when $b = 0$, σ is made from random key independent of that embedded in ϕ as well as the case in GAME.3 at $\delta = 0$. The decryption oracle is simulated just as given. However, the view of A_T in the simulation is slightly different from that in GAME.3 at $\delta = 0$ because dk and mk are made from independent sources, K'_0 and K_0, of KDF_2 while they are made from a single input to KDF_2 in GAME.3 at $\delta = 0$. However, from Corollary 1, we claim that $|\Pr[X_3 \,|\, \delta = 0] - \Pr[\tilde{b} = 0 \,|\, b = 0]| \leq 4\,\epsilon_{\mathrm{kdf}}$. (Since K'_0 and K_0 are randomly chosen and used only as inputs to KDF_2, the claim is proven by a straightforward reduction.)

In summary, we have:

$$\begin{aligned}
|\Pr[X_2] - \Pr[X_3]| &= \frac{1}{2}|\Pr[X_2 \,|\, \delta = 0] - \Pr[X_3 \,|\, \delta = 0]| \\
&\leq \frac{1}{2}|\Pr[\tilde{b} = 0 \,|\, b = 1] - \Pr[\tilde{b} = 0 \,|\, b = 0] - 4\,\epsilon_{\mathrm{kdf}}| \\
&\leq \epsilon_{\mathrm{lkem}} + 2\epsilon_{\mathrm{kdf}}
\end{aligned}$$

Proof of (4): Proof is done by constructing A_L playing GAME.LKEM by using A_T in GAME.3. Basically, what A_L does is to simulate the encryption oracle by creating σ, and to simulate the decryption oracle by sending ϕ_j and (σ_j, τ_j) to the decryption oracle of GAME.LKEM. These simulations are easy because necessary keys are provided by corresponding oracles of GAME.LKEM. A_L finally outputs $\tilde{\delta}$ as A_T does. Simulation is perfect since, in GAME.3, A_L will not send ϕ_j that is identical to ϕ to the decryption oracle but simply reject it. Hence the advantage of A_L in GAME.LKEM is the same as that of A_T in GAME.3.

E Kurosawa-Desmedt KEM

We define the Kurosawa-Desmedt KEM as follows. Key generation function $\mathsf{KEM.Gen}$ is as illustrated in Section 5.2. It outputs $pk = (g_1, g_2, c, d)$ and $sk = (x_1, x_2, y_1, y_2)$. On input pk, $\mathsf{KEM.Enc}$ outputs random source key K and ciphertext $\phi = (u_1, u_2)$ such that

$$r \leftarrow \mathbb{Z}_q, u_1 = g_1^r, u_2 = g_2^r, \alpha = H(u_1||u_2), K = c^r d^{\alpha r}.$$

Given sk and ϕ, decryption function $\mathsf{KEM.Dec}$ outputs K such that

$$\alpha = H(u_1||u_2),\ K = {u_1}^{x_1+\alpha y_1} {u_2}^{x_2+\alpha y_2}.$$

We say $\phi = (u_1, u_2)$ is valid if there exists $r \in \mathbb{Z}_q$ such that $u_1 = g_1^r, u_2 = g^r$. Otherwise, it is invalid.

Lemma 2. *KD scheme is LCCA-secure and CS with respect to $\mathcal{P}^{\mathrm{mac}}$ under the DDH assumption.*

We start by describing $\mathsf{GAME.LKEM}$ against KD-KEM.

1. Run $\mathsf{KEM.Gen}$ to generate $pk = (g_1, g_2, c, d)$ and $sk = (x_1, x_2, y_1, y_2)$. Give pk to A_L.
2. A_L make query $((u_{1j}, u_{2j}), \eta_j)$ to extended oracle $\mathcal{VD}$ arbitrarily. Let $K_j = \mathsf{KEM.Dec}_{sk}(u_{1j}, u_{2j})$. Oracle $\mathcal{VD}$ returns $\perp$ if (K_j, η_j) does not satisfy $\mathcal{P}^{\mathrm{mac}}$. Otherwise, it returns K_j (which might be $\perp$ anyway).
3. On receiving the challenge request from A_L, choose $r^* \leftarrow Z_q$ and computes

   $$u_1^* = g_1^{r^*},\ u_2^* = g_2^{r^*}, \alpha^* = H(u_1^*, u_2^*), v^* = c^{r^*} d^{r^* \alpha^*}.$$

   Let $\phi^* = (u_1^*, u_2^*)$ and $K_1 = v^*$. Choose $K_0 \in G_q$ randomly. Then choose $\delta \leftarrow \{0, 1\}$ and sends (ϕ^*, K_δ) to A_L.
4. A_L continues to query to the extended decryption oracle and eventually outputs $\tilde{\delta} \in \{0, 1\}$.

Since the proof is quite similar to that of [17, 16], we only show a sketch of it. From a viewpoint of A_L, there are four unknown variables x_1, x_2, y_1, y_2, and two (linear) equations on them given by (the discrete log of) c and d. Hence there is a freedom of $4 - 2 = 2$ dimensions. We consider this probability space on (x_1, x_2, y_1, y_2) of the 2 dimensions. Let $\mathsf{GAME.0}$ be the original game as shown above. We will define a sequence of games $\mathsf{GAME.1}, \cdots$. Let X_i be the event that $\tilde{\delta} = \delta$ in $\mathsf{GAME.}i$.

$\mathsf{GAME.1}$ is the same as $\mathsf{GAME.0}$, except that v^* is computed as $v^* = (u_1^*)^{x_1+y_1\alpha^*}(u_2^*)^{x_2+y_2\alpha^*}$. It is clear that $\Pr[X_1] = \Pr[X_0]$ because the value of v^* does not change.

$\mathsf{GAME.2}$ is the same as $\mathsf{GAME.1}$, except that the challenger chooses $u_1^*, u_2^* \in G_q$ at random. By DDH assumption, $\Pr[X_2] - \Pr[X_1]$ is negligible.

GAME.3 is the same as GAME.2, except that the extended decryption oracle returns $\perp$ for a query $((u_{1j}, u_{2j}), \eta_j)$ if (u_{1j}, u_{2j}) is invalid.

Suppose that A_L queries with invalid (u_{1j}, u_{2j}) at step 2 or step 5 of GAME.2. Let $\alpha' = H(u_{1j}, u_{2j})$ and $v_j = {u_{1j}}^{x_1+y_1\alpha'} {u_{2j}}^{x_2+y_2\alpha'}$.
(Step 2) As shown in [11], v_j is uniformly distributed over G_q. At this point, we can show that $\mathcal{P}^{\mathrm{mac}}$ will be satisfied with random key only with negligible probability when MAC forgery is infeasible. One can easily show that A_L that outputs η_j that satisfies $\mathcal{P}^{\mathrm{mac}}$ with random key can be turned into an adversary to one-time secure MAC. Hence the query is rejected with overwhelming probability.
(Step 5) If $(u_{1j}, u_{2j}) = (u_1^*, u_2^*)$, then the query is immediately rejected by the extended decryption oracle. Suppose that $(u_{1j}, u_{2j}) \neq (u_1^*, u_2^*)$. Note that (u_1^*, u_2^*) is invalid with overwhelming probability. Then as shown in [11], v^* and v_j are pair-wise independently distributed over G_q. Therefore, as well as in the above case, the query is rejected with overwhelming probability. Consequently, $\Pr[X_3] - \Pr[X_2]$ is negligible.

GAME.4 is the same as GAME.3, except that v^* is chosen at random from G_q. In GAME.3, (u_1^*, u_2^*) is invalid with overwhelming probability and v^* is uniformly distributed over G_q. Hence $\Pr[X_4] - \Pr[X_3]$ is negligible. Also in game GAME.4, it is clear that $\Pr[X_4] = 1/2$ because the view of A_L is independent of δ. This completes the proof on LCCA security.

Finally, the above argument also proves that the scheme is ciphertext sound with respect to $\mathcal{P}^{\mathrm{mac}}$ since every query including invalid (u_{1j}, u_{2j}) is rejected by the predicate with overwhelming probability.

F Details of Refined Fujisaki-Okamoto Conversion

The scheme. Let $\Pi_E = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public-key encryption. By $\mathcal{R}$, we denote a space of random coins used in PKE.Enc. Let $H : \{0,1\}^\lambda \times \{0,1\}^* \to \mathcal{R}$ and $G : \{0,1\}^\lambda \to \mathcal{K}_D$ be random oracles. TKEM.Gen is the same as PKE.Gen; Given 1^λ, it outputs key pair (pk, sk). TKEM.Key is that, given pk, it outputs $dk \leftarrow G(K)$ where $K \leftarrow \{0,1\}^\lambda$ and state information (pk, K). TKEM.Enc and TKEM.Dec are as follows.

Function: $\mathsf{TKEM.Enc}((pk, K), \tau)$

$r \leftarrow H(K||\tau)$
$\psi \leftarrow \mathsf{PKE.Enc}_{pk}(K; r)$
Output ψ.

Function: $\mathsf{TKEM.Dec}_{sk}(\psi, \tau)$

$K \leftarrow \mathsf{PKE.Dec}_{sk}(\psi)$
$r \leftarrow H(K||\tau)$
If $\psi \leftarrow \mathsf{PKE.Enc}_{pk}(K; r)$, output $G(K)$.
Output $\perp$, otherwise.

Suppose that the PKE is one-way against chosen plaintext attack and γ-uniform. [2] Then the following holds.

[2] Roughly, a PKE is one-way against chosen plaintext attack if it is infeasible to compute K from $\mathsf{PKE.Enc}_{pk}(K)$. Also, a PKE is γ-uniform if, for any K and ψ, randomly chosen r causes $\psi = \mathsf{PKE.Enc}_{pk}(K; r)$ with probability less than γ. See [15] for details.

Theorem 4. *The above Tag-KEM is CCA secure. Especially,* $\epsilon_{\mathrm{tkem}} \leq q_D\,\gamma + q_G\,\epsilon_{\mathrm{ow}}$.

Proof. The following is the CCA attack against the Tag-KEM in Section 5.3. By $\mathcal{O}$, we denote decryption oracle $\mathsf{TKEM.Dec}_{sk}(\cdot,\cdot)$. Let GAME.0 denote this CCA game.

Step 1. $(pk, sk) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$
Step 2. $\upsilon_1 \leftarrow {A_T}^{\mathcal{O},H,G}(pk)$
Step 3. $K_1 \leftarrow \{0,1\}^\lambda$, $dk_1 = G(K_1)$, $dk_0 \leftarrow \mathcal{K}_D$, $\delta \leftarrow \{0,1\}$.
Step 4. $(\tau, \upsilon_2) \leftarrow {A_T}^{\mathcal{O},H,G}(\upsilon_1, dk_\delta)$
Step 5. $\psi \leftarrow \mathsf{PKE.Enc}_{pk}(K_1; H(K_1||\tau))$
Step 6. $\tilde{\delta} \leftarrow {A_T}^{\mathcal{O},H,G}(\upsilon_2, \psi)$

We define games, GAME.1 and GAME.2, by modifying GAME.0 and examining fluctuation of probability $\Pr[X_0], \cdots, \Pr[X_2]$ where X_i denotes the event that $\tilde{\delta} = \delta$ in GAME.i.

We treat each random oracle as a table that appends an entry every time it is drawn with a fresh query. Given fresh K_j, oracle G outputs random dk_j and append (dk_j, K_j) to its table. Given fresh $K_j||\tau_j$, oracle H outputs random r_j. Since $K_j||\tau_j$ and r_j uniquely determines corresponding ciphertext, say ψ_j, we assume that H stores $(K_j||\tau_j, r_j, \psi_j)$ to the table. When exact reduction cost is concerned, computation time for the ciphertexts, which is linear in the number of H oracle queries, should be included in the running time of the simulator.

GAME.1: For every decryption query (ψ_j, τ_j), if table of H does not have an entry that contains both ψ_j and τ_j, return $\perp$.

Since (ψ_j, τ_j) is not in the table of H, r_j is chosen randomly in GAME.0. With random r_j, it passes the verification test in the decryption with probability at most γ. If at most q_D queries are made, the probability that there is a query that is accepted in GAME.0 but not in GAME.1 is at most $q_D\gamma$. The view of the adversary is unchanged unless such a query is made. Accordingly,

$$|\Pr[X_0] - \Pr[X_1]| \leq q_D\,\gamma.$$

GAME.2: If oracle G receives K_1, abort the game.

The game is also aborted if K_1 is already asked to G when it is chosen at Step 3. Let AskG denote the event that G receives K_1. By using A_T that causes AskG, we construct an adversary, A_{ow}, that derives K from given ψ and pk. A_{ow} first flips coin $j^\star \leftarrow \{1, \ldots, q_G\}$ and give pk to A_T. A_{ow} then simulates the oracles in GAME.2 as follows.

- H and G are simulated just as given. Given $j^\star$-th query $K_{j^\star}$ to G, output it and halt.
- For every decryption query (ψ_j, τ_j), if there is an entry, $(K_i||\tau_i, r_i, \psi_i)$, such that $\psi_j = \psi_i$ and $\tau_j = \tau_i$, return $G(K_i)$. Otherwise, return $\perp$.
- Given request to the encryption oracle, send dk chosen randomly from $\mathcal{K}_D$. Then, given τ, return ψ.

If $j^\star$-th query to G is made before the encryption oracle is invoked, the simulation is perfect. Even if it happens after the encryption oracle is invoked, randomly chosen dk perfectly simulates the output of the encryption oracle regardless of the choice of δ. Accordingly, $A_{\rm ow}$ perfectly simulates GAME.2 up to the moment $j^\star$-th query to G is made. And once event AskG happens at $j^\star$-th query, the output of $A_{\rm ow}$ is $\mathsf{PKE.Dec}(sk, \psi)$.

Running time of $A_{\rm ow}$ is almost the same as that of A_T plus computing time of encryption function q_H times. Now we have

$$|\Pr[X_1] - \Pr[X_2]| \leq q_G\, \epsilon_{\rm ow}.$$

Since A_T never asks K_1 to G in GAME.2, δ is independent from the view of A_T due to the true randomness of G. Hence $\Pr[X_2] = \frac{1}{2}$.

In conclusion, we have $\epsilon_{\rm tkem} = |\Pr[X_0] - \frac{1}{2}| \leq q_D\, \gamma + q_G\, \epsilon_{\rm ow}$ as stated.

G Tag-KEM from RCCA-secure KEM

RCCA security for KEM is derived in the same way as for PKE. That is, we modify GAME.KEM in Appendix A.1 in such a way that decryption oracle $\mathcal{O}$ returns 'test' when the result of decryption is in $\{K_1, K_0\}$. [3] Call this modified game GAME.RKEM. The scheme is RCCA-secure if any ppt adversary wins GAME.RKEM only with negligible advantage, say $\epsilon_{\rm rkem}$, as defined in the same way for CCA security. It is clear that the standard use of RCCA secure PKE is sufficient to construct RCCA secure KEM.

The construction of Tag-KEM is the same as that in Section 4.2 except that τ given to MAC.Sign and MAC.Ver is replaced by $\tau||\phi$. The resulting Tag-KEM is secure in the following sense.

Theorem 5. *If KEM is RCCA-secure, MAC is one-time secure, and DEM is secure, the above Tag-KEM is CCA-secure. Especially,* $\epsilon_{\rm tkem} \leq 2\epsilon_{\rm rkem} + (q_D + 3)\epsilon_{\rm kdf} + \frac{q_D}{2}\epsilon_{\rm mac}$

The attack game to the scheme is the same as GAME.0 shown in Appendix D with obvious modification that replaces $\mathsf{MAC.Sign}_{mk}(\tau)$ in Step 5 with $\mathsf{MAC.Sign}_{mk}(\tau||\phi)$. Now we give a sketch of a proof of security which is similar to that in Appendix D starting from GAME.0.

The idea of the proof is to continuously modify the game so that the relation among (dk, mk, ϕ) in the encryption oracle changes. Throughout the modifications, we only consider the case of $\delta = 0$. (Hence no modification in the case of $\delta = 1$.) By $[X], [Y]$, we denote that X and Y are independent. Similarly, $[X, Y]$ denotes that X and Y are related each other in some way.

- In GAME.0, dk is chosen randomly from $\mathcal{K}_D$, and mk and ϕ are related via K_1. Hence $([dk], [mk, \phi])$.

[3] Notice that when $\delta = 1$, K_0 is independent from the transcript between the adversary and the encryption oracle. Nevertheless, the decryption oracle returns 'test' if the decryption coincidentally yields K_0. Though it is totally a definitional matter and can be fixed if necessary, we prefer current definition that gives obvious reduction to RCCA PKE.

– In GAME.1, dk is generated by applying KDF to a random input. The relation is unchanged, i.e., $([dk], [mk, \phi])$.
– In GAME.2, mk is generated independently. Hence $([dk], [mk], [\phi])$.
– In GAME.3, dk and mk are coupled by generating them from a single application of KDF. Hence $([dk, mk], [\phi])$.

Now we consider the case of $\delta = 1$ in GAME.3. Since dk, mk and ϕ are all relative to K_1 embedded in ϕ, we have $([dk, mk, \phi])$. Then, GAME.3 is easily reduced to GAME.RKEM. Remember that in GAME.RKEM, challenge (K_b, ϕ) is either $([K_b], [\phi])$ at $b = 0$ or $([K_b, \phi])$ at $b = 1$. Therefor, by generating dk and mk from K_b, we can create $([dk, mk], [\phi])$ at $b = 0$ and $([dk, mk, \phi])$ at $b = 1$. Therefore, distinguishing δ in GAME.3 implies to distinguishing b in GAME.RKEM. In the following, we show the details where the decryption oracle is also changing in accordance with the modifications of the encryption oracle.

GAME.1: The encryption oracle is modified in such a way that when $\delta = 0$, dk is generated by $(dk, *) \leftarrow \mathsf{KDF}_2(K_r)$ where K_r is chosen randomly from $\mathcal{K}_K$.

Since GAME.1 is exactly the same as that in the proof of Theorem 3 as shown in Appendix D, we have

$$|\Pr[X_0] - \Pr[X_1]| = \frac{1}{2}|\Pr[X_0 \,|\, \delta = 0] - \Pr[X_1 \,|\, \delta = 0]| \leq \epsilon_{\mathrm{kdf}}. \tag{8}$$

GAME.2: We modify the encryption oracle in such a way that, when $\delta = 0$, MAC key mk in Step 5 is generated from independently chosen random K_0. That is, $(*, mk) \leftarrow \mathsf{KDF}_2(K_1)$ in Step 5 is replaced with $K_0 \leftarrow \mathcal{K}_K$, $(dk', mk) \leftarrow \mathsf{KDF}_2(K_0)$. Then mk is used to create σ. (Challenge DEM key dk returned to the adversary is unchanged. dk' generated here will be used in the decryption oracle as shown below.) We also modify the decryption oracle in such a way that, when the encryption oracle selects $\delta = 0$, every decryption query $(\phi_j, \sigma_j, \tau_j)$ made after the challenge step is handled as follows.

– $K_j \leftarrow \mathsf{KEM.Dec}_{sk}(\phi_j)$
– If $K_j \notin \{K_1, K_0\}$, then proceed as done in GAME.1. Otherwise, verify the MAC by using mk and return dk' if the MAC is correct. Otherwise, return $\perp$.

Note that the modification affects only if $\delta = 0$. We claim that

$$|\Pr[X_1] - \Pr[X_2]| \leq \epsilon_{\mathrm{rkem}}. \tag{9}$$

GAME.3: We modify the encryption oracle in such a way that, when $\delta = 0$, it generates dk and mk together by computing $(dk, mk) \leftarrow \mathsf{KDF}_2(K_0)$ rather than compute them independently. Since dk' is no longer defined, the decryption oracle returns dk when $K_j \in \{K_1, K_0\}$ and the MAC is correct. (Or, the description of the decryption oracle is left unchanged by defining $dk' = dk$.)

We claim

$$|\Pr[X_2] - \Pr[X_3]| \leq \epsilon_{\mathrm{kdf}} + \frac{q_D}{2}(\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}) \tag{10}$$

and

$$|\Pr[X_3] - \frac{1}{2}| \leq \epsilon_{\mathrm{rkem}} \tag{11}$$

Conclusion: From (8), (9), (10), and (11), we have

$$\begin{aligned} |\Pr[X_0] - \frac{1}{2}| &\leq \epsilon_{\mathrm{kdf}} + \epsilon_{\mathrm{rkem}} + 2\epsilon_{\mathrm{kdf}} + \frac{q_D}{2}(\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}) + \epsilon_{\mathrm{rkem}} \\ &\leq 2\epsilon_{\mathrm{rkem}} + (q_D + 3)\epsilon_{\mathrm{kdf}} + \frac{q_D}{2}\epsilon_{\mathrm{mac}} \end{aligned}$$

as stated in the theorem.

Proof of (9). In the case of $\delta = 1$, GAME.1 and GAME.2 are identical. Hence we only consider the case of $\delta = 0$.

We show a reduction from adversary A_T playing in GAME.1 (and GAME.2) at $\delta = 0$ to adversary A_K that attacks the underlying RCCA-secure KEM. Specification of A_K follows.

Step 1. Forward given pk to A_T.
Step 2. Simulate the encryption oracle as follows.
- Given request from A_T, compute $K_r \leftarrow \mathcal{K}_K$, $(dk, *) \leftarrow \mathsf{KDF}_2(K_r)$ and return dk. Then, make a challenge request to the encryption oracle of GAME.RKEM and receive (ϕ, K_b). Then compute $(dk', mk) \leftarrow \mathsf{KDF}_2(K_b)$.
- Given τ from A_T, compute $\sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\tau||\phi)$ and return (ϕ, σ).

Step 3. Given decryption query $(\phi_j, \sigma_j, \tau_j)$ from A_T, send ϕ_j to the decryption oracle of GAME.RKEM. If K_j is returned, proceed as prescribed; Set $(dk_j, mk_j) \leftarrow \mathsf{KDF}_2(K_j)$ and return dk_j if $\mathsf{MAC.Ver}_{mk_j}(\sigma_j, \tau_j||\phi_j) = 1$. Otherwise return $\perp$. On the other hand, if 'test' is returned, verify the MAC by using mk as $\mathsf{MAC.Ver}_{mk}(\sigma_j, \tau_j||\phi_j) \stackrel{?}{=} 1$. If it holds, return dk'. Otherwise, return $\perp$.
Step 4. When A_T outputs $\tilde{\delta}$, output $\tilde{b} = \tilde{\delta}$.

First consider the case of $b = 1$ in the above simulation. Regarding the simulation of the encryption oracle, observe that dk is made from independent K_r and mk and ϕ are related via K_b in the correct manner as in GAME.1. Hence the view of A_T with respect to the encryption oracle is identical to that in GAME.1 at $\delta = 0$. The decryption oracle also simulates the view of GAME.1 at $\delta = 0$ perfectly unless 'test' is returned from the decryption oracle of GAME.RKEM. Observe that receiving 'test' means that K_b is embedded in both ϕ_j and ϕ (remember that we are in the case of $b = 1$). Hence mk and dk' derived from K_b are the same as those used in the real decryption oracle in GAME.1. Eventually, the simulation is perfect even if 'test' is returned. Therefore, we have;

$$\Pr[X_1 \,|\, \delta = 0] = \Pr[\tilde{b} = 0 \,|\, b = 1]. \tag{12}$$

Next consider the case of $b = 0$ which is much complicated than the previous case. We concern the difference from the simulation and $\mathsf{GAME.2}$ at $\delta = 0$ for this case. First observe that the encryption oracle is perfectly simulated since dk, mk, and ϕ distributes independently both in the simulation and $\mathsf{GAME.2}$ at $\delta = 0$. The decryption oracle is also simulated perfectly since receiving 'test' means $K_j \in \{K_0, K_1\}$ and the MAC is verified as prescribed by using mk defined in the encryption oracle. Accordingly, we have;

$$\Pr[X_2 \,|\, \delta = 0] = \Pr[\tilde{b} = 0 \,|\, b = 0]. \tag{13}$$

From (12) and (13), we have

$$\begin{aligned} |\Pr[X_1 \,|\, \delta = 0] - \Pr[X_2 \,|\, \delta = 0]| &= |\Pr[\tilde{b} = 0 \,|\, b = 1] - \Pr[\tilde{b} = 0 \,|\, b = 0]| \\ &\leq 2\epsilon_{\mathrm{rkem}}. \end{aligned} \tag{14}$$

Accordingly,

$$|\Pr[X_1] - \Pr[X_2]| = \frac{1}{2}|\Pr[X_1 \,|\, \delta = 0] - \Pr[X_2 \,|\, \delta = 0]| \leq \epsilon_{\mathrm{rkem}}$$

as claimed in (9).

Proof of (10). We show a reduction from A_T playing in $\mathsf{GAME.2}$ to A_{KDF} that attacks KDF in the sense of distinguishing distribution U_0 and U_1 described in Appendix A.3. Given challenge (dk, mk) chosen randomly from distribution U_b where $b \leftarrow \{0, 1\}$, adversary A_{KDF} works as follows.

Step 1. Generate $(pk, sk) \leftarrow \mathsf{KEM.Gen}(1^\lambda)$ and give pk to A_T.
Step 2. Encryption oracle is simulated as prescribed by using dk and mk. That is, first give dk to A_T. Then, on receiving τ, compute $(K_1, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}()$ and $\sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\tau||\phi)$. Then return ϕ and σ to A_T.
Step 3. Decryption oracle is simulated by using the real secret key sk; On receiving a query $(\phi_j, \sigma_j, \tau_j)$, decrypt ϕ_j into K_j. If $K_j = K_1$, then return dk if $\mathsf{MAC.Ver}_{mk}(\sigma_j, \tau_j||\phi_j) = 1$, or return $\perp$, otherwise. If $K_j \neq K_1$, compute $(dk_j, mk_j) \leftarrow \mathsf{KDF}_2(K_j)$ and return dk_j if $\mathsf{MAC.Ver}_{mk_j}(\sigma_j, \tau_j||\phi_j) = 1$, or return $\perp$, otherwise.
Step 4. When A_T outputs $\tilde{\delta}$, output it as $\tilde{b}$.

We first inspect the simulation in the case of $b = 0$ where dk and mk are independent each other. Our concern is the similarity of the simulation and $\mathsf{GAME.2}$ at $\delta = 0$. In the encryption oracle, ϕ is randomly generated. Hence dk, mk, and ϕ are all independent. This is the case in $\mathsf{GAME.2}$ at $\delta = 0$. With regard to the decryption oracle, let $E1$ denote an event that $K_j = K_1$ and the MAC is correct. Unless $E1$ happens, the output of the decryption oracle is correct; Especially, observe that dk_j is correctly related to K_j as well as those returned by the real decryption oracle in $\mathsf{GAME.2}$. On the other hand, if $E1$ happens, the

simulating decryption oracle returns dk that is independent of mk while the real decryption oracle in GAME.2 returns dk' generated together with mk. Accordingly, the simulation is done just as prescribed for GAME.2 at $\delta = 0$ unless $E1$ happens. Hence we have

$$|\Pr[X2 \,|\, \delta = 0] - \Pr[\tilde{b} = 0 \,|\, b = 0]| \leq \Pr[E1]. \tag{15}$$

Leaving $\Pr[E1]$ as is for a while, we next inspect the case of $b = 1$ where dk and mk came from the same application of KDF_2. We concern the similarity of the simulation and GAME.3 at $\delta = 0$. The encryption oracle is perfectly done as prescribed because ϕ is independently generated while dk and mk are correctly related as expected in GAME.3 at $\delta = 0$. Unlike the case of $b = 0$, the decryption oracle is perfect because it is expected to return the correctly related $dk(= dk')$ when $K_j = K_1$ and correct MAC is observed. Accordingly, we have

$$\Pr[X3 \,|\, \delta = 0] = \Pr[\tilde{b} = 0 \,|\, b = 1]. \tag{16}$$

From (15), (16), and Corollary 1 in Appendix A.3, we have

$$\begin{aligned} |\Pr[X2 \,|\, \delta = 0] - \Pr[X3 \,|\, \delta = 0]| &\leq \Pr[E1] + \Pr[\tilde{b} = 0 \,|\, b = 0] - \Pr[\tilde{b} = 0 \,|\, b = 1] \\ |\Pr[X2] - \Pr[X3]| &\leq 2\epsilon_{\mathrm{kdf}} + \frac{1}{2}\Pr[E1] \end{aligned} \tag{17}$$

We now claim that

$$\Pr[E1] \leq q_D(\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}). \tag{18}$$

To prove this claim, we first modify the attack game for one-time secure MAC (GAME.MAC in Appendix A.2) in such a way that the MAC key mk is generated by $(*, mk) \leftarrow \mathsf{KDF}_2(K)$ where K is chosen randomly from $\mathcal{K}_K$. Let GAME.KDF-MAC refer to this modified game. Let $\epsilon_{\text{kdf-mac}}$ denote the maximum success probability in GAME.KDF-MAC over all ppt machines. By a straightforward reduction, we can prove that if there exists an adversary that wins GAME.KDF-MAC with noticeably better probability than that in GAME.MAC, then there exists an adversary that successfully break the security of KDF. Namely, $|\epsilon_{\text{kdf-mac}} - \epsilon_{\mathrm{mac}}| \leq 2\epsilon_{\mathrm{kdf}}$. Hence we have

$$\epsilon_{\text{kdf-mac}} \leq \epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}. \tag{19}$$

Now we show a reduction from A_T that causes event $E1$ to an adversary, A_{KM}, that attacks MAC in GAME.KDF-MAC. It is important to see that MAC key mk is totally independent from any other variables such as dk and ϕ in GAME.2 at $\delta = 0$ where $E1$ is defined. Specification of A_{KM} follows.

Step 1. Choose $j^\star \leftarrow \{1, \ldots, q_D\}$.
Step 2. Generate (pk, sk) by following $\mathsf{KEM.Gen}$ and give pk to A_T.
Step 3. Simulate the encryption oracle as follows.
- Given request from A_T, compute $K_r \leftarrow \mathcal{K}_K$, $(dk, *) \leftarrow \mathsf{KDF}_2(K_r)$ and return dk. Also compute $(K_1, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}()$.
- Given τ from A_T, send $\tau||\phi$ to the MAC oracle of $\mathsf{GAME.KDF\text{-}MAC}$ and receive σ. Then return ϕ and σ A_T.

Step 4. For every decryption query $(\phi_j, \sigma_j, \tau_j)$ except $j = j^\star$, simulate the decryption oracle as follows. Compute $K_j \leftarrow \mathsf{KEM.Dec}_{sk}(\phi_j)$. If $K_j = K_1$, return $\perp$. Otherwise, compute $(dk_j, mk_j) \leftarrow \mathsf{KDF}_2(K_j)$ and verify $\mathsf{MAC.Ver}_{mk_j}(\sigma, \tau_j||\phi_j)$. If it is correct, return dk_j. Otherwise return $\perp$.
Step 5. On receiving $j^\star$-th decryption query, output σ_j and $\tau_j||\phi_j$ and halt.

First we examine the encryption oracle. It creates dk and ϕ from independent K_r and K_1, and hidden mk is chosen by the MAC oracle in $\mathsf{GAME.KDF\text{-}MAC}$ in the same way as specified in $\mathsf{GAME.2}$. Namely, it is generated from random and independent K_0 applying KDF_2. Therefore, the encryption oracle simulates the view of $\delta = 0$ in $\mathsf{GAME.2}$. Next we verify the decryption oracle. Assume that $E1$ first happens at $j^\star$-th query. Then, all queries for $j < j^\star$ that causes $K_j \in \{K_0, K_1\}$ should be rejected. This is achieved in the simulation because:

- $K_j = K_1$ is detectable and rejected immediately.
- $K_j = K_0$ is not detectable in the simulation because K_0 is chosen by the MAC oracle. However, verifying the MAC by using given mk does the job since the MAC is incorrect as we assume $E1$ does not happen yet.

Therefore, the decryption oracle is perfectly simulated until $E1$ happens. If $E1$ happens at $j^\star$-th query, A_{KM} wins in $\mathsf{GAME.KDF\text{-}MAC}$ because the query is assumed not to be exactly the same as the challenge. This concludes the proof of equation (18).

From (17) and (18), we have $|\Pr[X_2] - \Pr[X_3]| \leq 2\epsilon_{\mathrm{kdf}} + \frac{q_D}{2}(\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}})$ which concludes the proof of (10).

Proof of (11). We show a reduction from A_T in $\mathsf{GAME.3}$ to A_K that attacks the underlying RCCA-secure KEM. The reduction is rather straightforward except the case where 'test' is returned from the decryption oracle of $\mathsf{GAME.RKEM}$. Specification of A_K follows.

Step 1. Forward given pk to A_T.

Step 2. Simulate the encryption oracle as follows.
- Given request from A_T, make a challenge request to the encryption oracle of $\mathsf{GAME.RKEM}$ and receive (ϕ, K_b). Then compute $(dk, mk) \leftarrow \mathsf{KDF}_2(K_b)$ and return dk.
- Given τ from A_T, compute $\sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\tau||\phi)$ and return (ϕ, σ).

Step 3. Given decryption query $(\phi_j, \sigma_j, \tau_j)$ from A_T, send ϕ_j to the decryption oracle of $\mathsf{GAME.RKEM}$. If 'test' is returned, return dk if $\mathsf{MAC.Ver}_{mk}(\sigma_j, \tau_j||\phi_j) = 1$, otherwise, return $\perp$. If K_j is returned, set $(dk_j, mk_j) \leftarrow \mathsf{KDF}_2(K_j)$ and return dk_j if $\mathsf{MAC.Ver}_{mk_j}(\sigma_j, \tau_j||\phi_j) = 1$, otherwise, return $\perp$.

Step 4. When A_T outputs $\tilde{\delta}$, output $\tilde{b} = \tilde{\delta}$.

When $b = 1$, the encryption oracle is simulated so that (dk, mk) and ϕ are related correctly as in the case $\delta = 1$ in $\mathsf{GAME.3}$. On the other hand, when $b = 0$, the encryption oracle generates (dk, mk) and ϕ independently just as in the case $\delta = 0$ in $\mathsf{GAME.3}$. Similar observation holds for the decryption oracle. Accordingly, when $b = 1$ (and $b = 0$), the view of A_T is that of $\delta = 1$ (and $\delta = 0$, respectively) in $\mathsf{GAME.3}$. Hence

$$\begin{aligned}\frac{1}{2}\Pr[X_3 \,|\, \delta = 0] + \frac{1}{2}\Pr[X_3 \,|\, \delta = 1] &= \frac{1}{2}\Pr[\tilde{b} = b \,|\, b = 0] + \frac{1}{2}\Pr[\tilde{b} = b \,|\, b = 1] \\ \Pr[X_3] &= \Pr[\tilde{b} = b] \\ &\leq \frac{1}{2} + \epsilon_{\mathrm{rkem}}\end{aligned}$$

as claimed in (11).