CCA Security and Trapdoor Functions via Key-Dependent-Message Security

Fuyuki Kitagawa[1], Takahiro Matsuda[2], and Keisuke Tanaka[3]

[1] NTT Secure Platform Laboratories, Tokyo, Japan, `fuyuki.kitagawa.yh@hco.ntt.co.jp`
[2] National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan, `t-matsuda@aist.go.jp`
[3] Tokyo Institute of Technology, Tokyo, Japan, `keisuke@is.titech.ac.jp`

Abstract

We study the relationship among public-key encryption (PKE) satisfying indistinguishability against chosen plaintext attacks (IND-CPA security), that against chosen ciphertext attacks (IND-CCA security), and trapdoor functions (TDF). Specifically, we aim at finding a unified approach and some additional requirement to realize IND-CCA secure PKE and TDF based on IND-CPA secure PKE, and show the following two main results.

As the first main result, we show how to achieve IND-CCA security via a weak form of key-dependent-message (KDM) security. More specifically, we construct an IND-CCA secure PKE scheme based on an IND-CPA secure PKE scheme and a secret-key encryption (SKE) scheme satisfying one-time KDM security with respect to projection functions (projection-KDM security). Projection functions are very simple functions with respect to which KDM security has been widely studied. Since the existence of projection-KDM secure PKE implies that of the above two building blocks, as a corollary of this result, we see that the existence of IND-CCA secure PKE is implied by that of projection-KDM secure PKE.

As the second main result, we extend the above construction of IND-CCA secure PKE into that of TDF by additionally requiring a mild requirement for each building block. Our TDF satisfies adaptive one-wayness. We can instantiate our TDF based on a wide variety of computational assumptions. Especially, we obtain the first TDF (with adaptive one-wayness) based on the sub-exponential hardness of the constant-noise learning-parity-with-noise (LPN) problem.

Keywords: chosen ciphertext security, trapdoor functions, key-dependent-message security

Contents

1 Introduction

1.1 Background

Public-key encryption (PKE) is one of the most fundamental cryptographic primitives. The most basic security requirement for PKE is indistinguishability against chosen plaintext attacks (IND-CPA security) [GM82]. However, in many practical applications, PKE schemes should satisfy the stronger notion of indistinguishability against chosen ciphertext attacks (IND-CCA security) [RS92, DDN91] in order to take active adversaries into consideration [Ble98].

Since IND-CCA security is stronger than IND-CPA security, the existence of IND-CCA secure PKE implies that of IND-CPA secure one. However, the implication of the opposite direction is not known. While a partial negative result was shown by Gertner, Malkin, and Myers [GMM07], the question whether an IND-CCA secure PKE scheme can be constructed from an IND-CPA secure one has still been standing as a major open question in cryptography.

In addition to IND-CCA secure PKE, a family of trapdoor functions (TDF) is also a fundamental primitive whose relationship with IND-CPA secure PKE has been widely studied. It was shown that an IND-CPA secure PKE can be constructed from TDF [Yao82, BHSV98]. For the opposite direction, Gertner, Malkin, and Reingold [GMR01] showed a negative result stating that TDF cannot be built from PKE in a black-box way.

In fact, in the random oracle model [BR93], we can construct both IND-CCA secure PKE and TDF based solely on IND-CPA secure PKE using a simple and unified derandomization technique [BHSV98, FO99]. However, in the standard model, we cannot use such a simple derandomization technique successfully. Especially, in order to construct IND-CCA secure PKE and TDF in the standard model by circumventing the impossibility results [GMM07, GMR01], we need non-black-box techniques or some additional requirements for the building block PKE scheme.

Hajiabadi and Kapron [HK15] tackled the above question, and as a main result, they built a TDF based on a PKE scheme satisfying circular security [CL01] and a randomness re-usability property called reproducibility [BBS03]. Since their TDF satisfies one-wayness under correlated products, based on the same assumption, they also obtained a construction of IND-CCA secure PKE by relying on the result by Rosen and Segev [RS09]. Their TDF construction is elegant and can also be extended to deterministic encryption [BBO07]. However, due to the somewhat strong additional requirement of randomness re-usability, its instantiations are limited to specific number theoretic assumptions.

In this work, we further study the above question. Especially, we aim at finding a unified approach and some additional requirement to realize IND-CCA secure PKE and TDF based on IND-CPA secure PKE.

1.2 Our Results

We show a unified approach to build IND-CCA secure PKE and TDF based on IND-CPA secure PKE by additionally using secret-key encryption (SKE) satisfying a weak form of *key-dependent-message (KDM) security* [BRS03]. Roughly speaking, an encryption scheme is said to be KDM secure if it can securely encrypt a message of the form $f(\mathsf{sk})$, where sk is the secret key and f is a function. The details of our results are as follows.

IND-CCA Security via Key-Dependent-Message Security. As the first main result, we construct an IND-CCA secure PKE scheme based on an IND-CPA secure PKE scheme and an SKE scheme satisfying KDM security. The building block SKE scheme is required to be one-time KDM secure with respect to projection functions (projection-KDM secure). Projection functions are very simple functions such that each output bit depends on at most a single bit of an input. An SKE scheme satisfying one-time projection-KDM security can be built from a wide variety of computational assumptions [BHHO08, ACPS09, BG10, BLSV18, DGHM18]. We obtain this result based on a construction technique used by Koppula and Waters [KW18] who showed how to construct IND-CCA secure attribute-based encryption (ABE) from IND-CPA secure one using a pseudorandom generator (PRG) with a special security property called hinting PRG. We extend the techniques of Koppula and Waters in several aspects. See Section 2 for the details.

The existence of PKE satisfying projection-KDM security against chosen plaintext attacks implies that of the above two building blocks. Therefore, as a corollary of this result, we see that the existence of IND-CCA secure PKE is implied by that of PKE with projection-KDM security (against CPA!).

Given our result and the result by Koppula and Waters, it is natural to ask what is the relationship between hinting PRG and one-time KDM secure SKE. To clarify this, we show that a one-time projection-KDM secure SKE scheme can be built from a hinting PRG. This means that one-time projection-KDM secure SKE is not a stronger assumption than hinting PRG.

Previously, Matsuda and Hanaoka [MH15] constructed an IND-CCA secure PKE scheme from a PKE scheme satisfying the sender non-committing property and an SKE scheme satisfying one-time KDM security with respect to circuits of a-priori bounded size. We improve their result in the sense that our construction requires weaker security properties for both of the underlying PKE and SKE schemes compared to theirs.

On Black-Box Usage of Building Blocks. Our construction of an IND-CCA secure PKE scheme is *fully-black-box* [RTV04] and *non-shielding* [GMM07]. A construction of a PKE scheme is said to be shielding if the decryption algorithm of the scheme does *not* call the encryption algorithm of the building block schemes, and otherwise it is called non-shielding. We show that our construction being a non-shielding construction is essential by showing that a fully-black-box and shielding construction of an IND-CCA secure PKE scheme based on our assumptions is *impossible* by extending the impossibility result shown by Gertner et al. [GMM07]. More specifically, we show that there is no fully-black-box and shielding construction of an IND-CCA secure PKE scheme based on a projection-KDM secure PKE scheme that trivially implies both of our building blocks.

Extension to TDF. As the second main result, we extend the above construction of an IND-CCA secure PKE scheme into that of a TDF by additionally requiring a mild requirement for each building block. Our construction of a TDF satisfies adaptive one-wayness [KMO10]. Adaptive one-wayness ensures that an adversary cannot invert a function in the family even under the existence of the inversion oracle, and thus it is a much stronger security property compared to ordinary one-wayness.

The additional requirements for the building blocks are as follows.

- First, we require that the underlying IND-CPA secure PKE scheme have the *pseudorandom ciphertext property*. Namely, a ciphertext of the underlying IND-CPA secure PKE scheme needs to be indistinguishable from a uniformly random element sampled from the ciphertext space of the scheme.

- Second, we require that the underlying projection-KDM secure SKE scheme be *randomness-recoverable*. Namely, random coins used to encrypt a message needs to be recovered together with the message in the decryption process.

Both of the above two requirements are mild in the following sense.

For the first requirement, a number of IND-CPA secure PKE schemes based on concrete computational assumptions naturally have this property. In fact, as far as we know, an IND-CPA secure PKE scheme satisfying the pseudorandom ciphertext property can be constructed from any concrete computational assumption implying IND-CPA secure PKE.

For the second requirement, the randomness-recovering property is easy to achieve in the secret-key setting while this property is so hard to achieve in the public-key setting that it immediately yields a TDF. Projection-KDM secure PKE schemes based on projective hash functions [BHHO08, BG10, Wee16] can easily be transformed into SKE variants satisfying the randomness-recovering property. Also, projection-KDM secure SKE schemes based on the learning-parity-with-noise (LPN) and learning-with-errors (LWE) assumptions proposed by Applebaum, Cash, Peikert, and Sahai [ACPS09] already satisfy this property. Moreover, even the recent constructions of KDM secure PKE schemes based on the computational Diffie-Hellman (CDH) and factoring assumptions [BLSV18, DGHM18] can be transformed into one-time projection-KDM secure SKE with the randomness-recovering property.

As noted above, the additional requirements needed to realize a TDF are mild. As a result, we can instantiate our TDF based on a wide variety of computational assumptions. Especially, by combining the previous results [YZ16, ACPS09], we obtain the first TDF (with adaptive one-wayness) based on the sub-exponential hardness of the constant-noise LPN problem. Moreover, we also obtain the first TDF satisfying adaptive one-wayness based on the low-noise LPN assumption. Previously to our work, a TDF satisfying ordinary one-wayness based on the low-noise LPN assumption was shown by Kiltz, Masny, and Pietrzak [KMP14].

1.3 Concurrent and Subsequent Works

Very recently, in a concurrent work, Lombardi, Quach, Rothblum, Wichs, and Wu [LQR+19a] showed how to construct a reusable designated-verifier non-interactive zero-knowledge (DV-NIZK) argument system based on the combination of an IND-CPA secure PKE scheme and a hinting PRG. In one of the steps in their construction, they employed the construction methodology of Koppula and Waters [KW18], and a hinting PRG is used in the step.

Based on our technique in this paper, Lombardi et al. (in their latest update [LQR+19b]) and Kitagawa and Matsuda [KM19] independently and concurrently observe that a hinting PRG used in Lombardi et al.'s reusable DV-NIZK argument system can also be replaced with a one-time $\mathcal{P}$-KDM secure SKE in exactly the same way as we do in our work. That is, these works show that a reusable DV-NIZK argument system can be constructed from an IND-CPA secure PKE scheme and a one-time $\mathcal{P}$-KDM secure SKE scheme. This leads to the first reusable DV-NIZK argument system based on the LPN assumption.

Furthermore, Kitagawa and Matsuda [KM19] show that using the reusable DV-NIZK argument system above and our result on IND-CCA secure PKE, we can transform a KDM-CPA secure PKE scheme into a KDM-CCA secure one without requiring any additional assumption. This leads to the first KDM-CCA secure PKE schemes based on the CDH and LPN assumptions.

1.4 Paper Organization

In Section 2, we show an overview of our techniques. In Section 3, we review definitions of cryptographic primitives. In Section 4, we show our proposed IND-CCA secure KEM. In Section 5, we prove the impossibility of fully-black-box shielding constructions. Finally, in Section 6, we present our proposed TDF.

2 Technical Overview

We give an overview of our techniques.

2.1 Achieving IND-CCA Security via Randomness-Recovering

One of classical mechanisms for achieving IND-CCA security is adopting a validity checking by re-encryption in the decryption process. In this technique, we make an encryption scheme randomness-recoverable, that is, a randomness used to generate a ciphertext is recovered during the decryption process. Then, when decrypting the ciphertext, we can check that the ciphertext was well-formed by re-encrypting the decrypted message using the recovered randomness.

Such a mechanism can be easily implemented in the random oracle model. Fujisaki and Okamoto [FO99] showed that by designing the encryption algorithm as $\mathsf{Enc}(\mathsf{pk}, \mathsf{r}\|\mathsf{m}; \mathsf{H}(\mathsf{r}\|\mathsf{m}))$, we can construct an IND-CCA secure PKE scheme based on the above strategy, where $\mathsf{Enc}(\mathsf{pk}, \cdot; \cdot)$ is the encryption algorithm of an IND-CPA secure PKE scheme and H is a hash function modeled as a random oracle. On the other hand, in the standard model, realizing a randomness-recoverable encryption scheme is difficult. Almost all existing such schemes are based on a TDF with advanced security properties [PW08, RS09, KMO10]. The main theme of this work is how we implement the mechanism in the standard model when starting from an IND-CPA secure PKE scheme.

A naive idea for our goal would be to design the encryption algorithm as $\mathsf{Enc}(\mathsf{pk}, \mathsf{r}\|\mathsf{m}; \mathsf{r})$, where $\mathsf{Enc}(\mathsf{pk}, \cdot; \cdot)$ again denotes the encryption algorithm of an IND-CPA secure PKE scheme. Unfortunately, it seems difficult to prove the security of this construction based on its IND-CPA security, since in order to rely on IND-CPA security, we need to ensure that a message to be encrypted is completely independent of the encryption randomness r.

A natural idea to remove the dependency is to use a variant of the hybrid encryption paradigm. Namely, we design the encryption algorithm as $(\mathsf{Enc}(\mathsf{pk}, \mathsf{s}; \mathsf{r}), \mathsf{E}(\mathsf{s}, \mathsf{r}\|\mathsf{m}))$, where $\mathsf{E}(\mathsf{s}, \cdot)$ is the encryption algorithm of an SKE scheme. At first glance, the dependency is removed, but the construction is in fact at a "dead-lock" and it also seems difficult to prove its security. We can solve the dead-lock by using the *signaling technique*[1] recently introduced by Koppula and Waters [KW18] who showed how to construct IND-CCA secure ABE from IND-CPA secure one using a PRG with a special security property called hinting PRG.

2.2 Partial Randomness-Recovering Using the Signaling Technique

We now use $2n$ public keys $(\mathsf{pk}_i^v)_{i\in[n],v\in\{0,1\}}$ of the IND-CPA secure PKE scheme to encapsulate a secret key $\mathsf{s} = (\mathsf{s}_1, \ldots, \mathsf{s}_n) \in \{0,1\}^n$ of the SKE scheme, where $[n] := \{1, \ldots, n\}$. Below, let $(\mathsf{sk}_i^v)_{i\in[n],v\in\{0,1\}}$ be secret keys corresponding to $(\mathsf{pk}_i^v)_{i\in[n],v\in\{0,1\}}$. Roughly, we "encode" each bit s_i of s as $(\mathsf{ct}_i^0, \mathsf{ct}_i^1)$, where

$$\mathsf{ct}_i^{\mathsf{s}_i} = \mathsf{Enc}(\mathsf{pk}_i^{\mathsf{s}_i}, 1; \mathsf{r}_i^{\mathsf{s}_i}) \quad \text{and} \quad \mathsf{ct}_i^{1-\mathsf{s}_i} = \mathsf{Enc}(\mathsf{pk}_i^{1-\mathsf{s}_i}, 0; \mathsf{r}_i^{1-\mathsf{s}_i}).$$

Namely, we encapsulate s by using $2n$ ciphertexts $(\mathsf{ct}_i^0, \mathsf{ct}_i^1)_{i\in[n]}$. During the decapsulation, we decrypt ct_i^0 by using sk_i^0 and set $\mathsf{s}_i := 0$ if the decryption result is 1 and $\mathsf{s}_i := 1$ otherwise.

Of course, if we encrypt all of the random coins $(\mathsf{r}_i^v)_{i\in[n],v\in\{0,1\}}$ used to encapsulate s by the SKE scheme to make the resulting scheme randomness-recoverable, it leads to a dead-lock as before. However, by using the signaling technique used by Koppula and Waters, we can perform the validity check by re-encrypting n out of $2n$ ciphertexts of the IND-CPA secure PKE scheme in the decryption process, and solve the dead-lock as follows.

[1]Garg, Gay, and Hajiabadi [GGH19] also used a similar technique called *mirroring*.

We say that "an encoding $(\mathsf{ct}_i^0, \mathsf{ct}_i^1)$ signals α" when ct_i^α encrypts 1. By using an (ordinary) PRG and adding a "tag" T_i to each encoding $(\mathsf{ct}_i^0, \mathsf{ct}_i^1)$ as $(\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)$, we can build a mechanism ensuring that it is statistically impossible to generate an encoding $(\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)$ that signals both 0 and 1 at the same time. In order to implement this mechanism, we also add some random strings to the public key that are used to generate tags, but we ignore them for simplicity in this overview. In this case, we can perform the validity check of the key encapsulation part $(\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}$ by checking whether $(\mathsf{ct}_i^{\mathsf{s}_i})_{i\in[n]}$ are well-formed encryptions of 1 by re-encryption. This is intuitively because if we confirm that these n ciphertexts are encryptions of 1, we can also be sure that the remaining n ciphertexts $(\mathsf{ct}_i^{1-\mathsf{s}_i})_{i\in[n]}$ are not encrypting 1 due to the added mechanism based on the PRG and tags $(\mathsf{T}_i)_{i\in[n]}$, and thus we can finish the pseudo-validity-check of all $2n$ ciphertexts of the key encapsulation part. Thus, in this construction, in addition to a message to be encrypted, the SKE scheme needs to encrypt only n random coins $(\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]}$ used to generate $(\mathsf{ct}_i^{\mathsf{s}_i})_{i\in[n]}$.

2.3 Outline of the Proof: Necessity of KDM Secure SKE

We explain how to prove the IND-CCA security of the above construction. A ciphertext of the scheme is of the form

$$\Big(\ (\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]},\ \ \mathsf{E}(\mathsf{s}, (\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]} \| \mathsf{m})\ \Big).$$

The general picture of the security proof is the same as that for the ordinary hybrid encryption scheme, and thus we first eliminate the information of s from the key encapsulation part $(\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}$ and then complete the entire proof by using the security of SKE.

We first explain how to eliminate the information of s from the key encapsulation part. In the security proof, thanks to the validity check by re-encryption in the decryption process, we can simulate the decryption oracle correctly by using $(\mathsf{sk}_i^{\mathsf{s}_i})_{i\in[n]}$ instead of $(\mathsf{sk}_i^0)_{i\in[n]}$. In this case, we can change the distribution of $(\mathsf{ct}_i^{1-\mathsf{s}_i})_{i\in[n]}$ in the challenge ciphertext by using the IND-CPA security of the PKE scheme since $(\mathsf{r}_i^{1-\mathsf{s}_i})_{i\in[n]}$ used to generate $(\mathsf{ct}_i^{1-\mathsf{s}_i})_{i\in[n]}$ are not encrypted by the SKE scheme and the decryption oracle can be simulated without $(\mathsf{sk}_i^{1-\mathsf{s}_i})_{i\in[n]}$. We can eliminate the information of s from the key encapsulation part $(\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}$ by changing $(\mathsf{ct}_i^{1-\mathsf{s}_i})_{i\in[n]}$ encrypting 0 into ciphertexts encrypting 1. This means that after this change, every encoding $(\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)$ contained in the challenge ciphertext signals 0 and 1 at the same time. While an adversary cannot generate such an encoding that signals 0 and 1 at the same time as noted above, the reduction algorithm can do it by programming random strings contained in the public key that are used to generate tags $(\mathsf{T}_i)_{i\in[n]}$.

Since we eliminate the information of s from the key encapsulation part above, it seems that we can complete the entire security proof by using the security of the SKE scheme. However, in order to do so, we need an SKE scheme that satisfies *KDM security*. This is because the underlying SKE scheme needs to encrypt $(\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]}$, which is a message depending on the key s. Concretely, $(\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]}$ can be seen as $f(\mathsf{s})$ for the function f that has $(\mathsf{r}_i^v)_{i\in[n],v\in\{0,1\}}$ hard-wired, and given $\mathsf{s} \in \{0,1\}^n$ outputs $(\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]}$. Such a function is described as a very simple form of functions called projection functions, for which KDM security has been widely studied [BHHO08, ACPS09, BG10, App11, BLSV18, DGHM18]. In our construction, we need an SKE scheme satisfying only *one-time* KDM security with respect to projection functions, since our construction is basically a hybrid encryption scheme. This is the reason KDM secure SKE is needed for our construction of IND-CCA secure PKE.

The Construction by Koppula and Waters [KW18]. The construction we explained so far is in fact almost the same as the PKE variant of the construction proposed by Koppula and

Waters, except that a one-time KDM secure SKE scheme is used instead of a *hinting PRG*. Here, we briefly explain the notion of hinting PRG and how it is used in their construction.

A hinting PRG is a PRG that, given an n-bit string x, outputs an $(n+1)\cdot\ell$-bit string $y_0\|y_1\|\cdots\|y_n$, where y_i is an ℓ-bit string for every $i\in[n]$. Then, its security property requires that $Y := y_0\|(y_{i,0}\|y_{i,1})_{i\in[n]} \in \{0,1\}^{(2n+1)\cdot\ell}$ be indistinguishable from a uniformly random string in $\{0,1\}^{(2n+1)\cdot\ell}$, where $y_{i,x_i} = y_i$ and $y_{i,1-x_i}$ is a uniformly random string in $\{0,1\}^\ell$ for every $i\in[n]$. We see that the locations where $y_1,\cdots y_n$ are placed in Y depend on the seed x, and thus Y itself can be seen as a "hint" of the seed x. Therefore, we can say that the security property of a hinting PRG requires that its output be pseudorandom even if such a hint of the seed is revealed.

Koppula and Waters used a hinting PRG HPRG in their construction as follows. When encrypting a message m, their scheme first generates a seed $x=(x_1,\cdots,x_n)\in\{0,1\}^n$ of HPRG and computes $y_0\|y_1\|\cdots\|y_n \leftarrow \mathsf{HPRG}(x)$. Then, it generates an encapsulation of x by generating an encoding $(\mathsf{ct}_i^0,\mathsf{ct}_i^1,\mathsf{T}_i)$ of x_i in which y_i is used as the encryption randomness for $\mathsf{ct}_i^{x_i}$ for every $i\in[n]$. Note that $\mathsf{ct}_i^{1-x_i}$ is generated by using truly random coins. Moreover, it generates the data encapsulation part as $\mathsf{m}\oplus y_0$. The resulting ciphertext is of the form

$$\Big(\ (\mathsf{ct}_i^0,\mathsf{ct}_i^1,\mathsf{T}_i)_{i\in[n]},\ \mathsf{m}\oplus y_0\ \Big).$$

When decrypting the ciphertext, we can first recover x and thus $y_0\|y_1\|\cdots\|y_n \leftarrow \mathsf{HPRG}(x)$ from the encapsulation part. Since $(y_1,\cdots,y_n)$ are random coins used to generate $(\mathsf{ct}_i^{x_i})_{i\in[n]}$, we can also perform the pseudo-validity-check of all $2n$ ciphertexts of the key encapsulation part as we explained above. The security proof of their construction also goes through in a similar fashion to the proof of our construction, except that the security property of HPRG is utilized instead of KDM security.

2.4 Extension to TDF

We explain how we extend the above construction of IND-CCA secure PKE based on IND-CPA secure PKE and one-time KDM secure SKE, into a TDF. More concretely, we explain how we make the above construction completely randomness-recoverable.

In the above construction, there are two types of encryption randomness that are not recovered in the decryption process. The first one is $(\mathsf{r}_i^{1-\mathsf{s}_i})_{i\in[n]}$ for the underlying IND-CPA secure PKE scheme. The other one is the encryption randomness for the underlying SKE scheme. We require an additional requirement for each building block to make it possible to recover these two types of encryption randomness.

First, to deal with $(\mathsf{r}_i^{1-\mathsf{s}_i})_{i\in[n]}$ for the IND-CPA secure PKE scheme, we require the underlying IND-CPA secure PKE scheme have the *pseudorandom ciphertext property*. Namely, we require that a ciphertext of the underlying IND-CPA secure PKE scheme be indistinguishable from a uniformly random element sampled from the ciphertext space of the scheme. In the above construction, recall that we encode each bit s_i of s as $(\mathsf{ct}_i^0,\mathsf{ct}_i^1,\mathsf{T}_i)$, where $\mathsf{ct}_i^{\mathsf{s}_i} = \mathsf{Enc}(\mathsf{pk}_i^{\mathsf{s}_i},1;\mathsf{r}_i^{\mathsf{s}_i})$ and $\mathsf{ct}_i^{1-\mathsf{s}_i} = \mathsf{Enc}(\mathsf{pk}_i^{1-\mathsf{s}_i},0;\mathsf{r}_i^{1-\mathsf{s}_i})$. We now modify the way s_i is encoded so that $\mathsf{ct}_i^{1-\mathsf{s}_i}$ is an element sampled from the ciphertext space uniformly at random. We can still decode s_i correctly with overwhelming probability thanks to the signaling technique even if we add this modification.[2] Then, we see that the issue of recovering $(\mathsf{r}_i^{1-\mathsf{s}_i})_{i\in[n]}$ is solved by designing the TDF such that $(\mathsf{ct}_i^{1-\mathsf{s}_i})_{i\in[n]}$ are directly sampled from the ciphertext space as part of an input to the TDF.

Second, to deal with the random coins for the SKE scheme, we simply require that the SKE scheme be *randomness-recoverable*. Namely, we require that random coins used to encrypt a

[2]While we cannot achieve perfect correctness by this modification, we can still achieve almost-all-keys correctness [DNR04]. For its formal definition, see Section 3.

message be recovered with the message in the decryption process. The randomness-recovering property is easy to achieve in the secret-key setting, and it is also the case even if we require the SKE scheme to be KDM secure. In fact, we can easily construct a one-time projection-KDM secure SKE scheme that is randomness-recoverable by modifying existing projection-KDM secure PKE schemes [BHHO08, BG10, Wee16, BLSV18, DGHM18]. Moreover, the projection-KDM secure SKE schemes based on the LPN and LWE assumptions proposed by Applebaum et al. [ACPS09] already satisfy this property.

With the help of these two additional requirements, we can modify our IND-CCA secure PKE scheme into a TDF. Since our TDF is an extension of IND-CCA secure PKE, it naturally satisfies adaptive one-wayness [KMO10].

2.5 Optimizations and Simplifications

Finally, we explain several optimizations and simplifications that are applied in the actual constructions.

The first optimization is on the number of key pairs of the underlying IND-CPA secure PKE scheme. In the above overview, $2n$ key pairs of the underlying IND-CPA secure PKE scheme are used to construct the key encapsulation part $(\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}$. In our actual constructions, we use only two key pairs of the underlying IND-CPA secure PKE scheme. More concretely, in our actual constructions, every encoding $(\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)$ is generated by using the same pair of public keys $(\mathsf{pk}^0, \mathsf{pk}^1)$. In fact, if we allow a public parameter shared by all users of the resulting schemes, even one of these public keys, pk^1, can be put into the public parameter, and a public key of the resulting IND-CCA secure scheme and an evaluation key of the resulting TDF consist only of a *single* public key pk^0 of the underlying IND-CPA secure scheme. This optimization is possible by devising at which step of the hybrid games we switch the secret keys of the underlying IND-CPA secure PKE scheme used to simulate the decryption oracle.

The second optimization is on how to make each tag T_i contained in each encoding $(\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)$. In the original signaling technique, a one-time signature scheme is additionally used in order to generate tags. We show that we can replace a one-time signature scheme with a target collision resistant hash function. Such a technique was previously used by Matsuda and Hanaoka [MH15]. Although both of these primitives can be realized using only a one-way function as an assumption [Rom90], this improvement is critical when constructing a TDF since if we attempt to use a one-time signature scheme for constructing a TDF, we would need to recover the random coins used to generate a key pair of the one-time signature scheme during the inversion process. We can avoid this issue by the use of a target collision resistant hash function instead. This modification is made possible due to the use of a deferred analysis technique in the security proof.

Third, we make a simplification by using key encapsulation mechanism (KEM) instead of PKE. In this overview, we have explained how to construct an IND-CCA secure PKE scheme and a TDF based on IND-CPA secure PKE by additionally using KDM secure SKE. In our actual proposals, we construct an IND-CCA secure KEM (and a TDF) based on IND-CPA secure KEM and KDM secure SKE. As explained above, in the original signaling technique, we use an (ordinary) PRG. More precisely, in the original signaling technique, $\mathsf{ct}_i^{\mathsf{s}_i}$ in each encoding is generated as $\mathsf{ct}_i^{\mathsf{s}_i} = \mathsf{Enc}(\mathsf{pk}_i^{\mathsf{s}_i}, 1\|\mathsf{u}_i; \mathsf{r}_i^{\mathsf{s}_i})$, where u_i is a seed of PRG. In our actual construction, in order to hide the use of a PRG from the description and simplify the construction, we use a KEM whose session-key space is sufficiently larger than its randomness space. We can generically transform an IND-CPA secure PKE scheme into a KEM with such a property. We show that the signaling technique can be implemented by using such a KEM.

For the construction of TDF, we also add an optimization that is made possible by the pseudorandom ciphertext property of the underlying IND-CPA secure PKE scheme. By this optimization, an image of a function consists of n ciphertexts of the IND-CPA secure PKE scheme corresponding to $(\mathsf{ct}_i^{s_i})_{i\in[n]}$, n tags $(\mathsf{T}_i)_{i\in[n]}$, and a ciphertext of the SKE scheme.

We finally remark that all of the above optimizations and simplifications can be brought back to the construction of an IND-CCA secure ABE scheme based on an IND-CPA secure one and a hinting PRG by Koppula and Waters [KW18].

3 Preliminaries

In this section, we review the basic notation and the definitions of main cryptographic primitives. The definitions of primitives that are not reviewed here are given in Appendix A.

Basic Notation. $\mathbb{N}$ denotes the set of natural numbers, and for $n \in \mathbb{N}$, we define $[n] := \{1, \dots, n\}$. For a discrete finite set S, $|S|$ denotes its size, and $x \xleftarrow{\mathsf{r}} S$ denotes choosing an element x uniformly at random from S. For strings x and y, $x\|y$ denotes their concatenation. For a (probabilistic) algorithm or a function A, $y \leftarrow \mathsf{A}(x)$ denotes assigning to y the output of A on an input x, and if we need to specify a randomness r used in A, we denote $y \leftarrow \mathsf{A}(x; r)$ (in which case the computation of A is understood as deterministic on input x and r). $\mathsf{Sup}(\mathsf{A})$ denotes the support of A (i.e. the set of elements that could be output by A with non-zero probability). For any values x, y, $(x \overset{?}{=} y)$ is defined to be 1 if $x = y$ and 0 otherwise. λ denotes a security parameter. (P)PT stands for *(probabilistic) polynomial time*. A function $f(\lambda)$ is said to be *negligible* if $f(\lambda)$ tends to 0 faster than $\frac{1}{\lambda^c}$ for every constant $c > 0$. We write $f(\lambda) = \mathsf{negl}(\lambda)$ to denote that $f(\lambda)$ is a negligible function. $\mathsf{poly}(\cdot)$ denotes an unspecified positive polynomial.

3.1 Key Encapsulation Mechanism

Here, we review the definitions for a KEM. For the definition of correctness, we formalize "almost-all-keys" correctness, which is naturally adapted from the definition for PKE formalized by Dwork, Naor, and Reingold [DNR04].

Definition 1 (Key Encapsulation Mechanism) *A key encapsulation mechanism (KEM)* KEM *consists of the three PPT algorithms* $(\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$*:*

- KKG *is the key generation algorithm that takes* 1^λ *as input, and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*. We assume that the security parameter* λ *determines the ciphertext space* $\mathcal{C}$*, the session-key space* $\mathcal{K}$*, and the randomness space* $\mathcal{R}$ *of* Encap.

- Encap *is the encapsulation algorithm that takes a public key* pk *as input, and outputs a ciphertext/session-key pair* $(\mathsf{ct}, \mathsf{k})$*.*

- Decap *is the (deterministic) decapsulation algorithm that takes a secret key* sk *and a ciphertext* ct *as input, and outputs a session-key* k *or the invalid symbol* $\bot \notin \mathcal{K}$*.*

Let $\epsilon : \mathbb{N} \to [0, 1]$*. We say that a KEM* $\mathsf{KEM} = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ *is* ϵ-almost-all-keys correct *if we have*

$$\mathsf{Err}_{\mathsf{KEM}}(\lambda) := \Pr_{(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KKG}(1^\lambda)}\left[\exists \mathsf{r} \in \mathcal{R} \text{ s.t. } \begin{array}{l} \mathsf{Encap}(\mathsf{pk}; \mathsf{r}) = (\mathsf{ct}, \mathsf{k}) \\ \wedge\ \mathsf{Decap}(\mathsf{sk}, \mathsf{ct}) \neq \mathsf{k} \end{array}\right] = \epsilon(\lambda).$$

(A public key pk *under which incorrect decapsulation could occur is called* erroneous*.) Furthermore, we just say that* KEM *is* correct *(resp.* almost-all-keys correct*) if* $\mathsf{Err}_{\mathsf{KEM}}(\lambda)$ *is zero (resp.* $\mathsf{negl}(\lambda)$*).*

```
$\mathsf{Expt}^{\mathsf{cca}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$ :
  $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KKG}(1^\lambda)$
  $(\mathsf{ct}^*,\mathsf{k}^*_1) \leftarrow \mathsf{Encap}(\mathsf{pk})$
  $\mathsf{k}^*_0 \leftarrow \mathcal{K}$
  $b \xleftarrow{\mathsf{r}} \{0,1\}$
  $b' \leftarrow \mathcal{A}^{\mathsf{Decap}(\mathsf{sk},\cdot)}(\mathsf{pk},\mathsf{ct}^*,\mathsf{k}^*_b)$
  Return $(b' \overset{?}{=} b)$.
```

```
$\mathsf{Expt}^{\mathsf{mcpa}}_{\mathsf{KEM},\ell,\mathcal{A}}(\lambda)$ :
  $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KKG}(1^\lambda)$
  $\forall i \in [\ell]$ :
    $(\mathsf{ct}^*_i,\mathsf{k}^*_{i,1}) \leftarrow \mathsf{Encap}(\mathsf{pk})$
    $\mathsf{k}^*_{i,0} \xleftarrow{\mathsf{r}} \mathcal{K}$
  $b \xleftarrow{\mathsf{r}} \{0,1\}$
  $b' \leftarrow \mathcal{A}(\mathsf{pk},(\mathsf{ct}^*_i,\mathsf{k}^*_{i,b})_{i\in[\ell]})$
  Return $(b' \overset{?}{=} b)$.
```

```
$\mathsf{Expt}^{\mathsf{mprct}}_{\mathsf{KEM},\ell,\mathcal{A}}(\lambda)$ :
  $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KKG}(1^\lambda)$
  $\forall i \in [\ell]$ :
    $(\mathsf{ct}^*_{i,1},\mathsf{k}^*_{i,1}) \leftarrow \mathsf{Encap}(\mathsf{pk})$
    $(\mathsf{ct}^*_{i,0},\mathsf{k}^*_{i,0}) \xleftarrow{\mathsf{r}} \mathcal{C} \times \mathcal{K}$
  $b \xleftarrow{\mathsf{r}} \{0,1\}$
  $b' \leftarrow \mathcal{A}(\mathsf{pk},(\mathsf{ct}^*_{i,b},\mathsf{k}^*_{i,b})_{i\in[\ell]})$
  Return $(b' \overset{?}{=} b)$.
```

Figure 1: Security experiments for a KEM: IND-CCA experiment (left), (Multi-challenge) IND-CPA experiment (center), and the (multi-challenge) pseudorandom ciphertext property experiment (right).

Now we review the security definitions for a KEM used in this paper, which are *IND-CCA security*, *IND-CPA security*, and the *pseudorandom ciphertext* property. For convenience, we will define the multi-challenge versions for the latter two notions, which are polynomially equivalent to the single-challenge versions via a standard hybrid argument.

Definition 2 (Security Notions for a KEM) *Let* $\mathsf{KEM} = (\mathsf{KKG},\mathsf{Encap},\mathsf{Decap})$ *be a KEM whose ciphertext and session-key spaces are* $\mathcal{C}$ *and* $\mathcal{K}$*, respectively. We say that* KEM *satisfies*

- IND-CCA security *if for all PPT adversaries* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{cca}}_{\mathsf{KEM},\mathcal{A}}(\lambda) := 2\cdot|\Pr[\mathsf{Expt}^{\mathsf{cca}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$*, where* $\mathsf{Expt}^{\mathsf{cca}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$ *is defined as in Figure 1 (left), and in the experiment,* $\mathcal{A}$ *is not allowed to submit* ct^* *to the decapsulation oracle* $\mathsf{Decap}(\mathsf{sk},\cdot)$*.*

- IND-CPA security *if for all PPT adversaries* $\mathcal{A}$ *and all polynomials* $\ell = \ell(\lambda)$*, we have* $\mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},\ell,\mathcal{A}}(\lambda) := 2\cdot|\Pr[\mathsf{Expt}^{\mathsf{mcpa}}_{\mathsf{KEM},\ell,\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$*, where* $\mathsf{Expt}^{\mathsf{mcpa}}_{\mathsf{KEM},\ell,\mathcal{A}}(\lambda)$ *is defined as in Figure 1 (center).*

- *the* pseudorandom ciphertext property *if for all PPT adversaries* $\mathcal{A}$ *and all polynomials* $\ell = \ell(\lambda)$*, we have* $\mathsf{Adv}^{\mathsf{mprct}}_{\mathsf{KEM},\ell,\mathcal{A}}(\lambda) := 2\cdot|\Pr[\mathsf{Expt}^{\mathsf{mprct}}_{\mathsf{KEM},\ell,\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$*, where* $\mathsf{Expt}^{\mathsf{mprct}}_{\mathsf{KEM},\ell,\mathcal{A}}(\lambda)$ *is defined as in Figure 1 (right).*

3.2 Secret-Key Encryption

Definition 3 (Secret-Key Encryption) *A secret-key encryption (SKE) scheme* SKE *consists of the three PPT algorithms* $(\mathsf{K},\mathsf{E},\mathsf{D})$*:*

- K *is the key generation algorithm that takes* 1^λ *as input, and outputs a secret key* sk*. We assume that the security parameter* λ *determines the secret key space* $\mathcal{K}$ *and the message space* $\mathcal{M}$*.*

- E *is the encryption algorithm that takes a secret key* sk *and a plaintext* m *as input, and outputs a ciphertext* ct*.*

- D *is the (deterministic) decryption algorithm that takes a secret key* sk *and a ciphertext* ct *as input, and outputs a plaintext* m *or the invalid symbol* $\perp \notin \mathcal{M}$*.*

An SKE scheme $\mathsf{SKE} = (\mathsf{K},\mathsf{E},\mathsf{D})$ *is said to be* correct *if for all* $\mathsf{sk} \in \mathcal{K}$ *and all* $\mathsf{m} \in \mathcal{M}$*, it holds that* $\mathsf{D}(\mathsf{sk},\mathsf{E}(\mathsf{sk},\mathsf{m})) = \mathsf{m}$*.*

In our proposed constructions of a TDF, we will use an SKE scheme that satisfies the "randomness-recovering decryption" property, which requires that for an honestly generate ciphertext, the randomness used to generate it can be recovered in the decryption process. We formally define the property as follows.

```
$\mathsf{Expt}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{F},\mathcal{A}}(\lambda)$ :
    $\mathsf{sk} \leftarrow \mathsf{K}(1^\lambda)$
    $b \xleftarrow{\mathsf{r}} \{0,1\}$
    $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{kdm}}(\cdot,\cdot)}(1^\lambda)$
    Return $(b' \overset{?}{=} b)$.
```

```
$\mathcal{O}_{\mathsf{kdm}}((f_0, f_1) \in \mathcal{F}^2)$ :
    $\mathsf{ct} \leftarrow \mathsf{E}(\mathsf{sk}, f_b(\mathsf{sk}))$
    Return $\mathsf{ct}$.
```

```
$\mathsf{Expt}^{\mathsf{aow}}_{\mathsf{TDF},\mathcal{A}}(\lambda)$ :
    $(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda)$
    $\mathsf{x}^* \leftarrow \mathsf{Samp}(1^\lambda)$
    $\mathsf{y}^* \leftarrow \mathsf{Eval}(\mathsf{ek}, \mathsf{x}^*)$
    $\mathsf{x}' \leftarrow \mathcal{A}^{\mathsf{Inv}(\mathsf{td},\cdot)}(\mathsf{ek}, \mathsf{y}^*)$
    Return $(\mathsf{x}' \overset{?}{=} \mathsf{x})$.
```

Figure 2: The KDM security experiment for an SKE (left) scheme, the KDM-encryption oracle used in the KDM security experiment (center), and the adaptive one-wayness experiment for a TDF (right).

Definition 4 (Randomness-Recovering Decryption) *Let* $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *be an SKE scheme whose secret key space is* $\mathcal{K}$*, whose plaintext space is* $\mathcal{M}$*, and the randomness space of whose encryption algorithm* E *is* $\mathcal{R}$*. We say that* SKE *satisfies the* randomness-recovering decryption *property, if there exists a deterministic PT algorithm* RD *(called the* randomness-recovering decryption *algorithm) such that for all* $\mathsf{sk} \in \mathcal{K}$*, all* $\mathsf{m} \in \mathcal{M}$*, and all* $\mathsf{r} \in \mathcal{R}$*, we have* $\mathsf{RD}(\mathsf{sk}, \mathsf{E}(\mathsf{sk}, \mathsf{m}; \mathsf{r})) = (\mathsf{m}, \mathsf{r})$*.*

Here, we recall KDM security of an SKE scheme. For simplicity, we only give the definition for the single key setting, which is sufficient for our purpose.

Definition 5 (KDM Security) *Let* $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *be an SKE scheme with a secret key space* $\mathcal{K}$ *and a plaintext space* $\mathcal{M}$*. For a family of functions* $\mathcal{F}$ *with domain* $\mathcal{K}$ *and range* $\mathcal{M}$ *and an adversary* $\mathcal{A}$*, consider the experiment* $\mathsf{Expt}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{F},\mathcal{A}}(\lambda)$ *defined as in Figure 2 (left), where the KDM-encryption oracle* $\mathcal{O}_{\mathsf{kdm}}$ *is described in Figure 2 (center).*

We say that SKE *is* $\mathcal{F}$-KDM secure *if for all PPT adversaries* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{F},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{F},\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$*.*

Furthermore, we say that SKE *is* one-time $\mathcal{F}$-KDM secure *if we have* $\mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{F},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ *for all PPT adversaries* $\mathcal{A}$ *that make a single KDM-encryption query.*

Note that in our definition, the KDM-encryption oracle $\mathcal{O}_{\mathsf{kdm}}$ is slightly different from the standard definition where an adversary is required to distinguish encryptions of $f(\mathsf{sk})$ from encryptions of some fixed message. However, the two definitions are equivalent if the function class $\mathcal{F}$ contains constant functions, and this is the case for the function families used in this paper (see below).

Function Families for KDM Security. We will deal with the following function families for KDM security of an SKE scheme with key space $\mathcal{K}$ and plaintext space $\mathcal{M}$:

- $\mathcal{P}$ *(Projection functions):* A function is said to be a projection function if each of its output bits depends on at most a single bit of its input. We denote by $\mathcal{P}$ the family of projection functions with domain $\mathcal{K}$ and range $\mathcal{M}$.
- $\mathcal{B}_{\mathsf{size}}$ *(Circuits of a-priori bounded size* size*):* We denote by $\mathcal{B}_{\mathsf{size}}$, where $\mathsf{size} = \mathsf{size}(\lambda)$ is a polynomial, the function family with domain $\mathcal{K}$ and range $\mathcal{M}$ such that each member in $\mathcal{B}_{\mathsf{size}}$ can be described by a circuit of size size.

3.3 Trapdoor Function

Here, we review the definitions for a TDF. As in the KEM case, for correctness, we will define almost-all-keys correctness.

Definition 6 (Trapdoor Function) *A trapdoor function (TDF)* TDF *consists of the four PPT algorithms* $(\mathsf{Setup}, \mathsf{Samp}, \mathsf{Eval}, \mathsf{Inv})$*:*

- Setup *is the setup algorithm that takes* 1^λ *as input, and outputs an evaluation key/trapdoor pair* $(\mathsf{ek}, \mathsf{td})$*. We assume that the security parameter* λ *determines the domain* $\mathcal{X}$*.*
- Samp *is the domain sampling algorithm that takes* 1^λ *as input, and outputs a domain element* $x \in \mathcal{X}$*.*
- Eval *is the evaluation algorithm that takes an evaluation key* ek *and a domain element* x *as input, and outputs some element* y*.*
- Inv *is the (deterministic) inversion algorithm that takes a trapdoor* td *and an element* y *as input, and outputs some element* x *which could be the invalid symbol* $\bot \notin \mathcal{X}$*.*

Let $\epsilon : \mathbb{N} \to [0,1]$*. We say that a TDF* $\mathsf{TDF} = (\mathsf{Setup}, \mathsf{Samp}, \mathsf{Eval}, \mathsf{Inv})$ *is* ϵ-almost-all-keys correct *if we have*

$$\mathsf{Err}_{\mathsf{TDF}}(\lambda) := \Pr_{(\mathsf{ek},\mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda)}\left[\ \exists x \in \mathcal{X} \text{ s.t. } \mathsf{Inv}(\mathsf{td}, \mathsf{Eval}(\mathsf{ek}, x)) \neq x\ \right] = \epsilon(\lambda).$$

Furthermore, we just say that TDF *is* correct *(resp.* almost-all-keys correct*) if* $\mathsf{Err}_{\mathsf{TDF}}(\lambda)$ *is zero (resp.* $\mathsf{negl}(\lambda)$*).*

Definition 7 (Adaptive One-wayness/(Ordinary) One-wayness) *Let* $\mathsf{TDF} = (\mathsf{Setup}, \mathsf{Samp}, \mathsf{Eval}, \mathsf{Inv})$ *be a TDF with domain* $\mathcal{X}$*. We say that* TDF *is* adaptively one-way *if for all PPT adversaries* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{aow}}_{\mathsf{TDF},\mathcal{A}}(\lambda) := \Pr[\mathsf{Expt}^{\mathsf{aow}}_{\mathsf{TDF},\mathcal{A}}(\lambda) = 1] = \mathsf{negl}(\lambda)$*, where* $\mathsf{Expt}^{\mathsf{aow}}_{\mathsf{TDF},\mathcal{A}}(\lambda)$ *is defined as in Figure 2 (right), and in the experiment,* $\mathcal{A}$ *is not allowed to submit* y^* *to the inversion oracle* $\mathsf{Inv}(\mathsf{td}, \cdot)$*.*

Furthermore, we say that TDF *is* one-way *if* $\mathsf{Adv}^{\mathsf{aow}}_{\mathsf{TDF},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ *for all adversaries that never use the inversion oracle* $\mathsf{Inv}(\mathsf{td}, \cdot)$*.*

4 Chosen Ciphertext Security via KDM Security

In this section, we show our proposed construction of an IND-CCA secure KEM.

Specifically, in Section 4.1, we present the formal description of our proposed KEM, state theorems regarding its correctness/security, and discuss its consequences and extensions. Then, in Sections 4.2 and 4.3, we prove the correctness and IND-CCA security of our proposed construction, respectively.

4.1 Our Construction

Let $\ell = \ell(\lambda)$ be a polynomial, which will denote the session-key length of the constructed KEM. Our construction uses the building blocks KEM, SKE, and Hash with the following properties:

- $\mathsf{KEM} = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ is a KEM such that (1) its session-key space is $\{0,1\}^{4\lambda}$, (2) the randomness space of Encap is $\{0,1\}^\lambda$, and (3) the image size of $\mathsf{Decap}(\mathsf{sk}, \cdot)$ for any sk output by $\mathsf{KKG}(1^\lambda)$ (other than $\bot$) is at most 2^λ.[3]

[3]These three requirements are without loss of generality for an IND-CPA secure KEM: The properties (1) and (3) can be achieved by stretching a session-key of a KEM with session-key space $\{0,1\}^\lambda$ by using a PRG $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{4\lambda}$, and the randomness space of Encap can also be freely adjusted by using a PRG whose range is the randomness space of Encap.

```
$\mathsf{KKG_{cca}}(1^\lambda)$ :
    $\forall v \in \{0,1\} : (\mathsf{pk}^v, \mathsf{sk}^v) \leftarrow \mathsf{KKG}(1^\lambda)$
    $\mathsf{A}_1, \ldots, \mathsf{A}_n, \mathsf{B} \xleftarrow{\mathsf{r}} \{0,1\}^{4\lambda}$
    $\mathsf{hk} \leftarrow \mathsf{HKG}(1^\lambda)$
    $\mathsf{PK} \leftarrow (\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i)_{i\in[n]}, \mathsf{B}, \mathsf{hk})$
    $\mathsf{SK} \leftarrow (\mathsf{sk}^0, \mathsf{PK})$
    Return $(\mathsf{PK}, \mathsf{SK})$.
```

```
$\mathsf{Encap_{cca}}(\mathsf{PK})$ :
    $(\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i)_{i\in[n]}, \mathsf{B}, \mathsf{hk}) \leftarrow \mathsf{PK}$
    $\mathsf{s} = (\mathsf{s}_1, \ldots, \mathsf{s}_n) \leftarrow \mathsf{K}(1^\lambda)$
    $\mathsf{r}_1^0, \ldots, \mathsf{r}_n^0, \mathsf{r}_1^1, \ldots, \mathsf{r}_n^1 \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$
    $\mathsf{k} \xleftarrow{\mathsf{r}} \{0,1\}^\ell$
    $\mathsf{ct_{SKE}} \leftarrow \mathsf{E}(\mathsf{s}, (\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]} \| \mathsf{k})$
    $\forall (i,v) \in [n] \times \{0,1\}$ :
        $(\mathsf{ct}_i^v, \mathsf{k}_i^v) \leftarrow \mathsf{Encap}(\mathsf{pk}^v; \mathsf{r}_i^v)$
    $\mathsf{h} \leftarrow \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i^0, \mathsf{ct}_i^1)_{i\in[n]} \| \mathsf{ct_{SKE}})$
    $\forall i \in [n]$ :
        $\mathsf{T}_i \leftarrow \mathsf{k}_i^{\mathsf{s}_i} + \mathsf{s}_i \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h})$   $^{(\dagger)}$
            $= \begin{cases} \mathsf{k}_i^0 & \text{if } \mathsf{s}_i = 0 \\ \mathsf{k}_i^1 + \mathsf{A}_i + \mathsf{B} \cdot \mathsf{h} & \text{if } \mathsf{s}_i = 1 \end{cases}$
    $\mathsf{CT} \leftarrow ((\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct_{SKE}})$
    Return $(\mathsf{CT}, \mathsf{k})$.
```

```
$\mathsf{Decap_{cca}}(\mathsf{SK}, \mathsf{CT})$ :
    $(\mathsf{sk}^0, \mathsf{PK}) \leftarrow \mathsf{SK}$
    $(\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i)_{i\in[n]}, \mathsf{B}, \mathsf{hk}) \leftarrow \mathsf{PK}$
    $((\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct_{SKE}}) \leftarrow \mathsf{CT}$
    $\mathsf{h} \leftarrow \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i^0, \mathsf{ct}_i^1)_{i\in[n]} \| \mathsf{ct_{SKE}})$
    $\forall i \in [n]$ :
        $\mathsf{s}_i \leftarrow 1 - (\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i^0) \overset{?}{=} \mathsf{T}_i)$   $^{(\star)}$
            $= \begin{cases} 0 & \text{if } \mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i^0) = \mathsf{T}_i \\ 1 & \text{otherwise} \end{cases}$
    $\mathsf{s} \leftarrow (\mathsf{s}_1, \ldots, \mathsf{s}_n) \in \{0,1\}^n$
    $\mathsf{m} \leftarrow \mathsf{D}(\mathsf{s}, \mathsf{ct_{SKE}})$
    Parse $\mathsf{m}$ as $((\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]}, \mathsf{k}) \in (\{0,1\}^\lambda)^n \times \{0,1\}^\ell$.
    If $\forall i \in [n]$ :
        $\mathsf{Encap}(\mathsf{pk}^{\mathsf{s}_i}; \mathsf{r}_i^{\mathsf{s}_i}) = (\mathsf{ct}_i^{\mathsf{s}_i}, \mathsf{T}_i - \mathsf{s}_i \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}))$
            then return $\mathsf{k}$ else return $\perp$.   $^{(\dagger)}$
```

Figure 3: The proposed KEM $\mathsf{KEM_{cca}}$. $^{(\dagger)}$ $\mathsf{h} \in \{0,1\}^\lambda$ is treated as an element of $\{0,1\}^{4\lambda}$ by some canonical injective encoding (say, putting the prefix $0^{3\lambda}$), and the arithmetic is done over $\mathrm{GF}(2^{4\lambda})$ where we identify $\{0,1\}^{4\lambda}$ with $\mathrm{GF}(2^{4\lambda})$. $^{(\star)}$ We call this step the *find step*.

- $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ is an SKE scheme whose secret key space is $\{0,1\}^n$ for some polynomial $n = n(\lambda)$ and whose plaintext space is $\{0,1\}^{n\cdot\lambda+\ell}$, and we denote the randomness space of E by $\mathcal{R}_{\mathsf{SKE}}$.

- $\mathsf{Hash} = (\mathsf{HKG}, \mathsf{H})$ is a keyed hash function such that the range of H is $\{0,1\}^\lambda$, which we are going to assume to be target collision resistant.

  (The formal definition of a target collision resistant hash function is recalled in Appendix A.1.)

Using these building blocks, the proposed KEM $\mathsf{KEM_{cca}} = (\mathsf{KKG_{cca}}, \mathsf{Encap_{cca}}, \mathsf{Decap_{cca}})$ is constructed as in Figure 3. Its session-key space is $\{0,1\}^\ell$, and the randomness space $\mathcal{R}$ of $\mathsf{Encap_{cca}}$ is $\mathcal{R} = \{0,1\}^n \times (\{0,1\}^\lambda)^{2n} \times \{0,1\}^\ell \times \mathcal{R}_{\mathsf{SKE}}$.

For the correctness and security of $\mathsf{KEM_{cca}}$, the following theorems hold.

Theorem 1 *Let $\epsilon = \epsilon(\lambda) \in [0,1]$. If* KEM *is ϵ-almost-all-keys correct and* SKE *is correct, then* $\mathsf{KEM_{cca}}$ *is $(\epsilon + n \cdot 2^{-\lambda})$-almost-all-keys correct.*

Theorem 2 *Assume that* KEM *is almost-all-keys correct and IND-CPA secure,* SKE *is one-time $\mathcal{P}$-KDM secure, and* Hash *is target collision resistant. Then,* $\mathsf{KEM_{cca}}$ *is IND-CCA secure.*

The proofs of Theorems 1 and 2 are given in Sections 4.2 and 4.3, respectively.

Implications to Black-Box Constructions/Reductions. It is straightforward to see that our construction uses the underlying primitives in a black-box manner. As will be clear from our security proof, our reduction algorithms also treat the underlying primitives and an adversary in a black-box manner. In fact, our construction/reduction is fully black-box in the sense of [RTV04]. Since there exists a black-box construction of a target collision resistant hash function from a one-way function, which can be trivially constructed from an IND-CPA secure PKE scheme/KEM in a black-box manner, and since an IND-CCA/CPA PKE scheme and KEM imply each other (in a black-box manner), we obtain the following result as a corollary of our theorems.

Corollary 1 *There exists a fully black-box construction of an IND-CCA secure PKE scheme/KEM from an IND-CPA secure PKE scheme/KEM and a one-time $\mathcal{P}$-KDM secure SKE scheme that can encrypt plaintexts of length $\Omega(n \cdot \lambda)$, where $n = n(\lambda)$ is the secret key length of the SKE scheme.*

Furthermore, since a $\mathcal{P}$-KDM secure *PKE* scheme trivially implies both an IND-CPA secure PKE scheme/KEM and a one-time $\mathcal{P}$-KDM secure SKE scheme, we obtain another corollary.

Corollary 2 *There exists a fully black-box construction of an IND-CCA secure PKE scheme/KEM from a $\mathcal{P}$-KDM secure PKE scheme.*

In contrast to Corollary 2, in Section 5, we will show that there exists no *shielding* black-box construction [GMM07] of an IND-CCA1 secure PKE scheme from a $\mathcal{P}$-KDM secure PKE scheme.

In [MH15], Matsuda and Hanaoka showed a construction of an IND-CCA secure PKE scheme/KEM from a PKE scheme satisfying the security notion called the sender non-committing property and a one-time $\mathcal{B}_{\mathsf{size}}$-KDM secure SKE scheme (where size is related to the running time of the sender non-committing encryption scheme). Although their construction uses the underlying primitives as black-boxes, their security reduction (to the $\mathcal{B}_{\mathsf{size}}$-KDM security of the underlying SKE scheme) is non-black-box in the sense that the reduction needs to use the description of one of the algorithms in the sender non-committing encryption scheme as a KDM-encryption query. Compared to the result by Matsuda and Hanaoka, our results are superior in terms of both the strength of the assumptions on the building blocks (IND-CPA security is weaker than the sender non-committing property, and $\mathcal{P}$-KDM security is weaker than $\mathcal{B}_{\mathsf{size}}$-KDM security), and the "black-boxness" of the reductions.

Hinting PRG vs. KDM Secure SKE. As mentioned earlier, the result of Koppula and Waters [KW18], when specialized to PKE, implies that if there exists an IND-CPA secure PKE scheme and a hinting PRG, one can realize an IND-CCA secure PKE scheme. Given our result in this section and the result of [KW18], it is natural to ask whether there exists an implication/separation between a (one-time) KDM secure SKE scheme and a hinting PRG. We give a partial affirmative answer to this question. Specifically, we show the following theorem.

Theorem 3 *If there exists a hinting PRG, then for any polynomials $m = m(\lambda)$ and $\mathsf{size} = \mathsf{size}(\lambda) \geq m$, there exists a one-time $\mathcal{B}_{\mathsf{size}}$-KDM secure SKE scheme whose plaintext space is $\{0,1\}^m$. Furthermore, for any polynomial $m = m(\lambda)$, there exists a fully black-box construction of a one-time $\mathcal{P}$-KDM secure SKE scheme with plaintext space $\{0,1\}^m$ from a hinting PRG.*

The formal proof of this theorem is given in Appendix B. This result shows that the existence of a KDM-secure SKE scheme is not stronger (as an assumption) than that of a hinting PRG. At this moment, it is not clear if the implication of the opposite direction can be established.

Additional Remarks.

- If we adopt the syntax of a KEM in which there is a public parameter shared by all users, then we can push pk^1, $(\mathsf{A}_i)_{i\in[n]}$, B, and hk in PK to a public parameter, so that a key pair of each user consists only of a single key pair $(\mathsf{pk}^0, \mathsf{sk}^0)$ of the underlying IND-CPA secure KEM.
- Although our proposed construction satisfies only almost-all-keys correctness, a minor variant of our construction can achieve perfect correctness, by using a PKE scheme and a PRG, instead of a KEM, as done in the Koppula-Waters construction [KW18].

4.2 Proof of Correctness (Proof of Theorem 1)

Let $\mathsf{PK} = (\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i)_{i\in[n]}, \mathsf{B}, \mathsf{hk})$ be a public key. Using pk^0, pk^1, and B in PK, we define the function $f : \{0,1\}^{3\lambda} \to \{0,1\}^{4\lambda}$ by

$$f(\mathsf{r}, \mathsf{r}', \mathsf{h}) : \Big[\ (\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Encap}(\mathsf{pk}^0; \mathsf{r});\ (\mathsf{ct}', \mathsf{k}') \leftarrow \mathsf{Encap}(\mathsf{pk}^1; \mathsf{r}');\ \text{Return } \mathsf{k} - \mathsf{k}' - \mathsf{B} \cdot \mathsf{h}\ \Big].$$

We say that a public key PK is *bad* if (1) pk^0 is erroneous, or (2) some of $(\mathsf{A}_i)_{i\in[n]}$ belongs to the image of f. Note that the image size of f is at most $2^{3\lambda}$. Since each A_i is chosen uniformly at random from $\{0,1\}^{4\lambda}$, when $\mathsf{KKG_{cca}}(1^\lambda)$ is executed, the probability that a bad PK is output is at most $\epsilon + n \cdot \frac{2^{3\lambda}}{2^{4\lambda}} = \epsilon + n \cdot 2^{-\lambda}$.

Now, consider the case that $(\mathsf{PK}, \mathsf{SK})$ is output by $\mathsf{KKG_{cca}}$ and PK is not bad. Let $\mathsf{R} = (\mathsf{s} = (\mathsf{s}_1, \dots, \mathsf{s}_n), (\mathsf{r}_i^0, \mathsf{r}_i^1)_{i\in[n]}, \mathsf{k}, \mathsf{r_{SKE}}) \in \{0,1\}^n \times (\{0,1\}^\lambda)^{2n} \times \{0,1\}^\ell \times \mathcal{R}_{\mathsf{SKE}}$ be a randomness for $\mathsf{Encap_{cca}}$, and let $(\mathsf{CT} = ((\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct_{SKE}}), \mathsf{k}) = \mathsf{Encap_{cca}}(\mathsf{PK}; \mathsf{R})$. Moreover, for each $i \in [n]$, let $\mathsf{s}_i' := 1 - (\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i^0) \overset{?}{=} \mathsf{T}_i)$.

Note that if $\mathsf{s}_i' = \mathsf{s}_i$ holds for all $i \in [n]$, then the decryption result of $\mathsf{ct_{SKE}}$ using $\mathsf{s}' = (\mathsf{s}_1', \dots, \mathsf{s}_n')$ as a secret key is exactly $(\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]} \| \mathsf{k}$ due to the correctness of SKE. Thus, the validity check done in the last step of $\mathsf{Decap_{cca}}$ never fails, and $\mathsf{Decap_{cca}}(\mathsf{SK}, \mathsf{CT})$ will output k.

Hence, it remains to show that $\mathsf{s}_i' = \mathsf{s}_i$ holds for all $i \in [n]$.

- For the positions i with $\mathsf{s}_i = 0$, we have $(\mathsf{ct}_i^0, \mathsf{k}_i^0 = \mathsf{T}_i) = \mathsf{Encap}(\mathsf{pk}^0; \mathsf{r}_i^0)$. Thus, the property that pk^0 is not erroneous implies $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i^0) = \mathsf{T}_i$, and we have $\mathsf{s}_i' = 0$.
- For the positions i with $\mathsf{s}_i = 1$, we have $(\mathsf{ct}_i^0, \mathsf{k}_i^0) = \mathsf{Encap}(\mathsf{pk}^0; \mathsf{r}_i^0)$ and $(\mathsf{ct}_i^1, \mathsf{k}_i^1 = \mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}) = \mathsf{Encap}(\mathsf{pk}^1; \mathsf{r}_i^1)$, where $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i^0, \mathsf{ct}_i^1)_{i\in[n]} \| \mathsf{ct_{SKE}})$. Since A_i is not in the image of f, we have

  $$\mathsf{A}_i \neq f(\mathsf{r}_i^0, \mathsf{r}_i^1, \mathsf{h}) = \mathsf{k}_i^0 - \mathsf{k}_i^1 - \mathsf{B} \cdot \mathsf{h} = \mathsf{k}_i^0 - (\mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}) - \mathsf{B} \cdot \mathsf{h} \quad \Longleftrightarrow \quad \mathsf{k}_i^0 \neq \mathsf{T}_i.$$

  Furthermore, since pk^0 is not erroneous, we have $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i^0) = \mathsf{k}_i^0$. These together imply that we must have $\mathsf{s}_i' = 1$.

The above shows that $\mathsf{s}_i' = \mathsf{s}_i$ holds for all $i \in [n]$.

Putting everything together, except for a probability at most $\epsilon + n \cdot 2^{-\lambda}$ over $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{KKG_{cca}}(1^\lambda)$, there exists no randomness R satisfying $\mathsf{Encap_{cca}}(\mathsf{PK}; \mathsf{R}) = (\mathsf{CT}, \mathsf{k})$ and $\mathsf{Decap_{cca}}(\mathsf{SK}, \mathsf{CT}) \neq \mathsf{k}$ simultaneously. $\square$ (**Theorem 1**)

4.3 Proof of IND-CCA Security (Proof of Theorem 2)

Let $\epsilon : \mathbb{N} \to [0,1]$ be such that KEM is ϵ-almost-all-keys correct. Let $\mathcal{A}$ be any PPT adversary that attacks the IND-CCA security of $\mathsf{KEM}_{\mathsf{cca}}$ and makes $q_{\mathsf{dec}} = q_{\mathsf{dec}}(\lambda) > 0$ decapsulation queries. We will show that for this $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_{\mathsf{tcr}}$, $\{\mathcal{B}^{j}_{\mathsf{cpa}}\}_{j\in[4]}$, $\mathcal{B}'_{\mathsf{cpa}}$, and $\mathcal{B}_{\mathsf{kdm}}$ (which makes a single KDM-encryption query) satisfying

$$\mathsf{Adv}^{\mathsf{cca}}_{\mathsf{KEM}_{\mathsf{cca}},\mathcal{A}}(\lambda) \le 2\cdot \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{Hash},\mathcal{B}_{\mathsf{tcr}}}(\lambda) + 2\cdot \sum_{j\in[4]} \mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}^{j}_{\mathsf{cpa}}}(\lambda) + 2q_{\mathsf{dec}}\cdot \mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}'_{\mathsf{cpa}}}(\lambda)$$
$$+ 2\cdot \mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{P},\mathcal{B}_{\mathsf{kdm}}}(\lambda) + 8\epsilon + n\cdot 2^{-\lambda+3} + n(q_{\mathsf{dec}}+1)\cdot 2^{-4\lambda+1}. \quad (1)$$

This is negligible by our assumption, and thus will prove the theorem.

Our proof is via a sequence of games argument using the following six games.

Game 1: This is the IND-CCA experiment $\mathsf{Expt}^{\mathsf{cca}}_{\mathsf{KEM}_{\mathsf{cca}},\mathcal{A}}(\lambda)$. However, for making it easier to describe the subsequent games, we change the ordering of the operations for how the key pair $(\mathsf{PK},\mathsf{SK})$ and the challenge ciphertext/session-key pair $(\mathsf{CT}^*,\mathsf{k}^*_b)$ are generated so that the distribution of $(\mathsf{PK},\mathsf{SK},\mathsf{CT}^*,\mathsf{k}^*_b)$ is identical to that in the original IND-CCA experiment.

Specifically, the description of the game is as follows:

- Generate $\mathsf{PK} = (\mathsf{pk}^0,\mathsf{pk}^1,(\mathsf{A}_i)_{i\in[n]},\mathsf{B},\mathsf{hk})$, $\mathsf{SK} = (\mathsf{sk}^0,\mathsf{PK})$, and $\mathsf{CT}^* = ((\mathsf{ct}^{*0}_i,\mathsf{ct}^{*1}_i,\mathsf{T}^*_i)_{i\in[n]},\mathsf{ct}^*_{\mathsf{SKE}})$ as follows:
  1. Compute $(\mathsf{pk}^v,\mathsf{sk}^v) \leftarrow \mathsf{KKG}(1^\lambda)$ for $v\in\{0,1\}$, and pick $\mathsf{B} \xleftarrow{\mathsf{r}} \{0,1\}^{4\lambda}$.
  2. Compute $\mathsf{s}^* = (\mathsf{s}^*_1,\dots,\mathsf{s}^*_n) \leftarrow \mathsf{K}(1^\lambda)$, and pick $\mathsf{r}^{*0}_1,\dots,\mathsf{r}^{*0}_n,\mathsf{r}^{*1}_1,\dots,\mathsf{r}^{*1}_n \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and $\mathsf{k}^*_1 \xleftarrow{\mathsf{r}} \{0,1\}^\ell$.
  3. Compute $\mathsf{ct}^*_{\mathsf{SKE}} \leftarrow \mathsf{E}(\mathsf{s}^*,(\mathsf{r}^{*(\mathsf{s}^*_i)}_i)_{i\in[n]}\|\mathsf{k}^*_1)$.
  4. Compute $(\mathsf{ct}^{*v}_i,\mathsf{k}^{*v}_i) \leftarrow \mathsf{Encap}(\mathsf{pk}^v;\mathsf{r}^{*v}_i)$ for every $(i,v)\in[n]\times\{0,1\}$.
  5. Compute $\mathsf{hk} \leftarrow \mathsf{HKG}(1^\lambda)$ and $\mathsf{h}^* \leftarrow \mathsf{H}(\mathsf{hk},(\mathsf{ct}^{*0}_i,\mathsf{ct}^{*1}_i)_{i\in[n]}\|\mathsf{ct}^*_{\mathsf{SKE}})$.
  6. Pick $\mathsf{A}_1,\dots,\mathsf{A}_n \xleftarrow{\mathsf{r}} \{0,1\}^{4\lambda}$.
  7. Compute $\mathsf{T}^*_i \leftarrow \mathsf{k}^{*(\mathsf{s}^*_i)}_i + \mathsf{s}^*_i\cdot(\mathsf{A}_i + \mathsf{B}\cdot\mathsf{h}^*)$ for every $i\in[n]$.
  8. Set $\mathsf{PK} \leftarrow (\mathsf{pk}^0,\mathsf{pk}^1,(\mathsf{A}_i)_{i\in[n]},\mathsf{B},\mathsf{hk})$, $\mathsf{SK} \leftarrow (\mathsf{sk}^0,\mathsf{PK})$, and $\mathsf{CT}^* \leftarrow ((\mathsf{ct}^{*0}_i,\mathsf{ct}^{*1}_i,\mathsf{T}^*_i)_{i\in[n]},\mathsf{ct}^*_{\mathsf{SKE}})$.
- Then, pick the random session-key $\mathsf{k}^*_0 \xleftarrow{\mathsf{r}} \{0,1\}^\ell$ and the challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$, and run $\mathcal{A}(\mathsf{PK},\mathsf{CT}^*,\mathsf{k}^*_b)$. From here on, $\mathcal{A}$ may start making decapsulation queries.
- Decapsulation queries $\mathsf{CT} = ((\mathsf{ct}^0_i,\mathsf{ct}^1_i,\mathsf{T}_i)_{i\in[n]},\mathsf{ct}_{\mathsf{SKE}})$ are answered as follows: First, compute $\mathsf{h} \leftarrow \mathsf{H}(\mathsf{hk},(\mathsf{ct}^0_i,\mathsf{ct}^1_i)_{i\in[n]}\|\mathsf{ct}_{\mathsf{SKE}})$. Next, compute $\mathsf{s}_i \leftarrow 1 - (\mathsf{Decap}(\mathsf{sk}^0,\mathsf{ct}^0_i) \overset{?}{=} \mathsf{T}_i)$ for every $i\in[n]$, and set $\mathsf{s} \leftarrow (\mathsf{s}_1,\dots,\mathsf{s}_n)$. Then, compute $\mathsf{m} \leftarrow \mathsf{D}(\mathsf{s},\mathsf{ct}_{\mathsf{SKE}})$ and parse m as $((\mathsf{r}^{\mathsf{s}_i}_i)_{i\in[n]},\mathsf{k}) \in (\{0,1\}^\lambda)^n \times \{0,1\}^\ell$. Finally, if $\mathsf{Encap}(\mathsf{pk}^{\mathsf{s}_i};\mathsf{r}^{\mathsf{s}_i}_i) = (\mathsf{ct}^{\mathsf{s}_i}_i,\mathsf{T}_i - \mathsf{s}_i\cdot(\mathsf{A}_i + \mathsf{B}\cdot\mathsf{h}))$ holds for all $i\in[n]$, then return k to $\mathcal{A}$. Otherwise, return $\perp$ to $\mathcal{A}$.
- At some point, $\mathcal{A}$ terminates with output $b'\in\{0,1\}$.

For convenience, in the following we will use the following sets:

$$\mathcal{S}_{\mathsf{zero}} := \left\{ j\in[n] \;\middle|\; \mathsf{s}^*_j = 0 \right\} \quad\text{and}\quad \mathcal{S}_{\mathsf{one}} := \left\{ j\in[n] \;\middle|\; \mathsf{s}^*_j = 1 \right\} = [n]\setminus\mathcal{S}_{\mathsf{zero}}.$$

Game 2: Same as Game 1, except for an additional rejection rule in the decapsulation oracle. Specifically, in this game, if $\mathcal{A}$'s decapsulation query $\mathsf{CT} = ((\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ satisfies $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i^0, \mathsf{ct}_i^1)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}) = \mathsf{h}^*$, then the decapsulation oracle immediately returns $\perp$ to $\mathcal{A}$.

Game 3: Same as Game 2, except for how A_i's for the positions $i \in \mathcal{S}_{\mathsf{zero}}$ are generated. Specifically, in this game, A_i for every position $i \in \mathcal{S}_{\mathsf{zero}}$ is generated by

$$\mathsf{A}_i \leftarrow \mathsf{k}_i^{*0} - \mathsf{k}_i^{*1} - \mathsf{B} \cdot \mathsf{h}^*. \tag{2}$$

(At this point, A_i's for the remaining positions $i \in \mathcal{S}_{\mathsf{one}}$ are unchanged.)

Game 4: Same as Game 3, except for the behavior of the decapsulation oracle. Specifically, for answering $\mathcal{A}$'s decapsulation queries $\mathsf{CT} = ((\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$, the oracle in this game first computes $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i^0, \mathsf{ct}_i^1)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}})$, and returns $\perp$ to $\mathcal{A}$ if $\mathsf{h} = \mathsf{h}^*$. (This rejection rule is the same as in Game 3.) Otherwise, the oracle uses the "alternative decapsulation algorithm" $\mathsf{AltDecap}$ and the "alternative secret key" SK' defined below for computing the decapsulation result k returned to $\mathcal{A}$.

$\mathsf{AltDecap}$ takes $\mathsf{SK}' := (\mathsf{sk}^1, \mathsf{PK})$ and CT as input, and proceeds identically to $\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK}, \mathsf{CT})$, except that the "find step" (i.e. the step for computing s_i's) is replaced with the following procedure:

$$\forall i \in [n]: \quad \mathsf{s}_i \leftarrow \left(\mathsf{Decap}(\mathsf{sk}^1, \mathsf{ct}_i^1) \overset{?}{=} \mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h} \right) = \begin{cases} 1 & \text{if } \mathsf{Decap}(\mathsf{sk}^1, \mathsf{ct}_i^1) = \mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h} \\ 0 & \text{otherwise} \end{cases}.$$

Note that due to this change, the decapsulation oracle answers $\mathcal{A}$'s queries without using sk^0.

Game 5: Same as Game 4, except for how A_i's for the positions $i \in \mathcal{S}_{\mathsf{one}}$ are generated. Specifically, in this game, A_i for $i \in \mathcal{S}_{\mathsf{one}}$ is also generated as in Equation 2.

Note that due to this change, all of $(\mathsf{A}_i)_{i\in[n]}$ are generated as in Equation 2. Furthermore, $\mathsf{T}_i^* = \mathsf{k}_i^{*0}$ holds for every $i \in [n]$, no matter whether $\mathsf{s}_i^* = 0$ or $\mathsf{s}_i^* = 1$. Indeed, this is the case for the positions $i \in \mathcal{S}_{\mathsf{zero}}$ by design. For the positions $i \in \mathcal{S}_{\mathsf{one}}$, we have

$$\mathsf{T}_i^* = \mathsf{k}_i^{*1} + \mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}^* = \mathsf{k}_i^{*1} + (\mathsf{k}_i^{*0} - \mathsf{k}_i^{*1} - \mathsf{B} \cdot \mathsf{h}^*) + \mathsf{B} \cdot \mathsf{h}^* = \mathsf{k}_i^{*0}.$$

Hence, in this game, values dependent on s^* appear only in the plaintext of $\mathsf{ct}_{\mathsf{SKE}}^*$ (i.e. $(\mathsf{r}_i^{*(\mathsf{s}_i^*)})_{i\in[n]} \| \mathsf{k}_1^*$).

Game 6: Same as Game 5, except that the information of the challenge bit b is erased from the SKE ciphertext $\mathsf{ct}_{\mathsf{SKE}}^*$. Specifically, in this game, $\mathsf{ct}_{\mathsf{SKE}}^*$ in the challenge ciphertext CT^* is generated by $\mathsf{ct}_{\mathsf{SKE}}^* \leftarrow \mathsf{E}(\mathsf{s}^*, 0^{n\cdot\lambda+\ell})$, instead of $\mathsf{ct}_{\mathsf{SKE}}^* \leftarrow \mathsf{E}(\mathsf{s}^*, (\mathsf{r}_i^{*(\mathsf{s}_i^*)})_{i\in[n]} \| \mathsf{k}_1^*)$.

For $j \in [6]$, let $\mathtt{SUC}_j$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $b' = b$ occurs) in Game j. By definition, we have $\mathsf{Adv}^{\mathsf{cca}}_{\mathsf{KEM}_{\mathsf{cca}},\mathcal{A}}(\lambda) = 2 \cdot |\Pr[\mathtt{SUC}_1] - 1/2|$. Thus, the triangle inequality implies

$$\mathsf{Adv}^{\mathsf{cca}}_{\mathsf{KEM}_{\mathsf{cca}},\mathcal{A}}(\lambda) \leq 2 \cdot \left(\sum_{j\in[5]} |\Pr[\mathtt{SUC}_j] - \Pr[\mathtt{SUC}_{j+1}]| + \left| \Pr[\mathtt{SUC}_6] - \frac{1}{2} \right| \right). \tag{3}$$

In the following, we show how the terms appearing in Equation 3 are bounded.

Lemma 1 *There exist PPT adversaries* $\mathcal{B}_{\mathsf{tcr}}$, $\{\mathcal{B}^{j}_{\mathsf{cpa}}\}_{j\in[2]}$, *and* $\mathcal{B}'_{\mathsf{cpa}}$ *satisfying*

$$|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \le \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{Hash},\mathcal{B}_{\mathsf{tcr}}}(\lambda) + \sum_{j\in[2]} \mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}^{j}_{\mathsf{cpa}}}(\lambda) + q_{\mathsf{dec}} \cdot \mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}'_{\mathsf{cpa}}}(\lambda) + 3\epsilon + n\cdot 2^{-\lambda+1} + n(q_{\mathsf{dec}}+1)\cdot 2^{-4\lambda}. \quad (4)$$

The proof of this lemma needs to use a deferred analysis (up to Game 4). We postpone the proof of this lemma to the end of the proof of this theorem.

Lemma 2 *There exists a PPT adversary* $\mathcal{B}^{3}_{\mathsf{cpa}}$ *such that* $|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]| = \mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}^{3}_{\mathsf{cpa}}}(\lambda)$.

Proof of Lemma 2. Using $\mathcal{A}$ as a building block, we show how to construct a PPT adversary $\mathcal{B}^{3}_{\mathsf{cpa}}$ that attacks the n-challenge IND-CPA security of KEM with the claimed advantage. The description is as follows.

$\mathcal{B}^{3}_{\mathsf{cpa}}(\mathsf{pk}', (\mathsf{ct}'^{*}_{i}, \mathsf{k}'^{*}_{i,\beta})_{i\in[n]})$**:** (where $\beta \in \{0,1\}$ denotes $\mathcal{B}^{3}_{\mathsf{cpa}}$'s challenge bit) First, $\mathcal{B}^{3}_{\mathsf{cpa}}$ runs $\mathsf{s}^* = (\mathsf{s}^*_1, \dots, \mathsf{s}^*_n) \leftarrow \mathsf{K}(1^\lambda)$, and sets $\mathsf{pk}^1 \leftarrow \mathsf{pk}'$ and $(\mathsf{ct}^{*1}_i, \mathsf{k}^{*1}_i) \leftarrow (\mathsf{ct}'^{*}_i, \mathsf{k}'^{*}_{i,\beta})$ for the positions $i \in \mathcal{S}_{\mathsf{zero}}$. Then, $\mathcal{B}^{3}_{\mathsf{cpa}}$ generates the remaining values in $\mathsf{PK} = (\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i)_{i\in[n]}, \mathsf{B}, \mathsf{hk})$, $\mathsf{SK} = (\mathsf{sk}^0, \mathsf{PK})$, $\mathsf{CT}^* = ((\mathsf{ct}^{*0}_i, \mathsf{ct}^{*1}_i, \mathsf{T}^*_i)_{i\in[n]}, \mathsf{ct}^*_{\mathsf{SKE}})$, and k^*_b in exactly the same way as Game 3 does, and runs $b' \leftarrow \mathcal{A}^{\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK},\cdot)}(\mathsf{PK}, \mathsf{CT}^*, \mathsf{k}^*_b)$. (Note that sk^0 is generated by $\mathcal{B}^{3}_{\mathsf{cpa}}$ ltself.) Finally, $\mathcal{B}^{3}_{\mathsf{cpa}}$ terminates with output $\beta' \leftarrow (b' \overset{?}{=} b)$.

If $\mathcal{B}^{3}_{\mathsf{cpa}}$'s challenge bit β is 0 (resp. 1), $\mathcal{B}^{3}_{\mathsf{cpa}}$ perfectly simulates Game 2 (resp. Game 3) for $\mathcal{A}$. In particular, if $\beta = 0$, then $\mathsf{k}^{*1}_i = \mathsf{k}'^{*}_{i,0}$ for every $i \in \mathcal{S}_{\mathsf{zero}}$ is chosen uniformly at random from $\{0,1\}^{4\lambda}$, and thus regardless of what values k^{*0}_i, B, and h^* take, $\mathsf{A}_i = \mathsf{k}^{*0}_i - \mathsf{k}^{*1}_i - \mathsf{B}\cdot\mathsf{h}^*$ is distributed uniformly at random in $\{0,1\}^{4\lambda}$. Under this situation, the probability that $b' = b$ holds (and thus $\mathcal{B}^{3}_{\mathsf{cpa}}$ outputs $\beta' = 1$) is exactly $\Pr[\mathtt{SUC}_2]$, i.e. we have $\Pr[\beta' = 1|\beta = 0] = \Pr[\mathtt{SUC}_2]$. Similarly, we have $\Pr[\beta' = 1|\beta = 1] = \Pr[\mathtt{SUC}_3]$. Hence, we have

$$\mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}^{3}_{\mathsf{cpa}}}(\lambda) = \big|\Pr[\beta' = 1|\beta = 0] - \Pr[\beta' = 1|\beta = 1]\big| = |\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]|,$$

as desired. □ (**Lemma 2**)

Lemma 3 $|\Pr[\mathtt{SUC}_3] - \Pr[\mathtt{SUC}_4]| \le 2\epsilon + n\cdot 2^{-\lambda+1}$ *holds.*

Proof of Lemma 3. Note that Game 3 and Game 4 proceed identically unless $\mathcal{A}$ makes a decapsulation query $\mathsf{CT} = ((\mathsf{ct}^0_i, \mathsf{ct}^1_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ such that $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}^0_i, \mathsf{ct}^1_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}) \ne \mathsf{h}^*$ and $\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK}, \mathsf{CT}) \ne \mathsf{AltDecap}(\mathsf{SK}', \mathsf{CT})$ hold simultaneously. We call such a decapsulation query *bad*. In the following, we will show that if PK is not "bad" in the sense specified below, a bad decapsulation query does not exist in Game 3 and Game 4, and the probability that PK becomes bad is bounded by $2\epsilon + n\cdot 2^{-\lambda+1}$. This will prove the lemma.

Fix the following values in Game 3:

- $(\mathsf{pk}^0, \mathsf{sk}^0), (\mathsf{pk}^1, \mathsf{sk}^1) \in \mathsf{Sup}(\mathsf{KKG}(1^\lambda))$ such that pk^0 and pk^1 are not erroneous, and $\mathsf{hk} \in \mathsf{Sup}(\mathsf{HKG}(1^\lambda))$.
- $\mathsf{s}^* = (\mathsf{s}^*_1, \dots, \mathsf{s}^*_n) \in \mathsf{Sup}(\mathsf{K}(1^\lambda))$, $\mathsf{r}^{*0}_1, \dots, \mathsf{r}^{*0}_n, \mathsf{r}^{*1}_1, \dots, \mathsf{r}^{*1}_n \in \{0,1\}^\lambda$, $\mathsf{k}^*_1 \in \{0,1\}^\ell$, and $\mathsf{r}^*_{\mathsf{SKE}} \in \mathcal{R}_{\mathsf{SKE}}$.

- $(\mathsf{ct}_i^{*v}, \mathsf{k}_i^{*v}) = \mathsf{Encap}(\mathsf{pk}^v; \mathsf{r}_i^{*v})$ for all $(i, v) \in [n] \times \{0,1\}$.
- $\mathsf{ct}^*_{\mathsf{SKE}} = \mathsf{E}(\mathsf{s}^*, (\mathsf{r}_i^{*(\mathsf{s}_i^*)})_{i\in[n]} \| \mathsf{k}_1^*; \mathsf{r}^*_{\mathsf{SKE}})$ and $\mathsf{h}^* = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i^{*0}, \mathsf{ct}_i^{*1})_{i\in[n]} \| \mathsf{ct}^*_{\mathsf{SKE}})$.

Let $\mathcal{C}$ be the ciphertext space of KEM. To define the notion of "badness" for a public key, we introduce two types of functions based on the above fixed values.

- For each $i \in \mathcal{S}_{\mathsf{zero}}$ and $v \in \{0,1\}$, we define the function $\widehat{g}_{i,v} : \{0,1\}^\lambda \times \mathcal{C} \times (\{0,1\}^\lambda \setminus \{\mathsf{h}^*\}) \to \{0,1\}^{4\lambda} \cup \{\bot\}$ by

$$\widehat{g}_{i,v}(\mathsf{r}, \mathsf{ct}', \mathsf{h}) : \left[\begin{array}{l} (\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Encap}(\mathsf{pk}^v; \mathsf{r});\ \mathsf{k}' \leftarrow \mathsf{Decap}(\mathsf{sk}^{1-v}, \mathsf{ct}'); \\ \text{If } \mathsf{k}' = \bot \text{ then return } \bot \text{ else return } \frac{(\mathsf{k}-\mathsf{k}')\cdot(-1)^v - \mathsf{k}_i^{*0} + \mathsf{k}_i^{*1}}{\mathsf{h}-\mathsf{h}^*} \end{array} \right].$$

  We say that a string $\mathsf{B} \in \{0,1\}^{4\lambda}$ is *bad* if B belongs to the image of $\widehat{g}_{i,v}$ for some $(i,v) \in \mathcal{S}_{\mathsf{zero}} \times \{0,1\}$. Due to the property that the image size of $\mathsf{Decap}(\mathsf{sk}^{1-v}, \cdot)$ is bounded by 2^λ, the image size of $\widehat{g}_{i,v}$ (excluding $\bot$) is at most $2^{3\lambda}$ for every $i \in \mathcal{S}_{\mathsf{zero}}$ and $v \in \{0,1\}$. Hence, when choosing $\mathsf{B} \xleftarrow{\mathsf{r}} \{0,1\}^{4\lambda}$, the probability that B is bad is at most $|\mathcal{S}_{\mathsf{zero}}| \cdot 2 \cdot \frac{2^{3\lambda}}{2^{4\lambda}} = |\mathcal{S}_{\mathsf{zero}}| \cdot 2^{-\lambda+1}$.

- For each $\mathsf{B}' \in \{0,1\}^{4\lambda}$ and $v \in \{0,1\}$, we define the function $g_{\mathsf{B}',v} : \{0,1\}^\lambda \times \mathcal{C} \times \{0,1\}^\lambda \to \{0,1\}^{4\lambda} \cup \{\bot\}$ by

$$g_{\mathsf{B}',v}(\mathsf{r}, \mathsf{ct}', \mathsf{h}) : \left[\begin{array}{l} (\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Encap}(\mathsf{pk}^v; \mathsf{r});\ \mathsf{k}' \leftarrow \mathsf{Decap}(\mathsf{sk}^{1-v}, \mathsf{ct}'); \\ \text{If } \mathsf{k}' = \bot \text{ then return } \bot \text{ else return } (\mathsf{k} - \mathsf{k}') \cdot (-1)^v - \mathsf{B}' \cdot \mathsf{h} \end{array} \right].$$

  For each $\mathsf{B}' \in \{0,1\}^{4\lambda}$, we say that a string $\mathsf{A}' \in \{0,1\}^{4\lambda}$ is *bad with respect to* B' if A' belongs to the image of $g_{\mathsf{B}',0}$ or that of $g_{\mathsf{B}',1}$. Again, due to the property that the image size of $\mathsf{Decap}(\mathsf{sk}^{1-v}, \cdot)$ is bounded by 2^λ, the image size of $g_{\mathsf{B},v}$ (excluding $\bot$) is at most $2^{3\lambda}$ for every $\mathsf{B}' \in \{0,1\}^{4\lambda}$ and $v \in \{0,1\}$. Hence, for any fixed $\mathsf{B}' \in \{0,1\}^{4\lambda}$, when choosing $\mathsf{A}_i \xleftarrow{\mathsf{r}} \{0,1\}^{4\lambda}$ for all $i \in \mathcal{S}_{\mathsf{one}}$, the probability that some of $\{\mathsf{A}_i\}_{i\in\mathcal{S}_{\mathsf{one}}}$ is bad with respect to B' is at most $|\mathcal{S}_{\mathsf{one}}| \cdot 2 \cdot \frac{2^{3\lambda}}{2^{4\lambda}} = |\mathcal{S}_{\mathsf{one}}| \cdot 2^{-\lambda+1}$.

We say that a public key PK generated in Game 3 is *bad* if (1) either pk^0 or pk^1 is erroneous, or (2) either B is bad or A_i for some $i \in \mathcal{S}_{\mathsf{one}}$ is bad with respect to B. By the union bound, the probability that PK is bad in Game 3 is bounded by $2\epsilon + |\mathcal{S}_{\mathsf{zero}}| \cdot 2^{-\lambda+1} + |\mathcal{S}_{\mathsf{one}}| \cdot 2^{-\lambda+1} = 2\epsilon + n \cdot 2^{-\lambda+1}$.

To complete the proof, in the following we show that if $\mathsf{PK} = (\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i)_{i\in[n]}, \mathsf{B}, \mathsf{hk})$ is not bad, then for any ciphertext $\mathsf{CT} = ((\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ such that $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i^0, \mathsf{ct}_i^1)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}) \neq \mathsf{h}^*$, we always have $\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK}, \mathsf{CT}) = \mathsf{AltDecap}(\mathsf{SK}', \mathsf{CT})$.

Let $\mathsf{CT} = ((\mathsf{ct}_i^0, \mathsf{ct}_i^1, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ be an arbitrary ciphertext satisfying $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i^0, \mathsf{ct}_i^1)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}) \neq \mathsf{h}^*$. For each $i \in [n]$, define

$$\mathsf{s}_i := 1 - \Big(\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i^0) \overset{?}{=} \mathsf{T}_i \Big), \qquad \text{and}$$
$$\mathsf{s}'_i := \Big(\mathsf{Decap}(\mathsf{sk}^1, \mathsf{ct}_i^1) \overset{?}{=} \mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h} \Big).$$

We consider two cases and show that $\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK}, \mathsf{CT}) = \mathsf{AltDecap}(\mathsf{SK}', \mathsf{CT})$ holds in either case.

- **Case 1: For all positions $i \in [n]$, there exists a pair $(\mathsf{r}, v) \in \{0,1\}^\lambda \times \{0,1\}$ satisfying $\mathsf{Encap}(\mathsf{pk}^v; \mathsf{r}) = (\mathsf{ct}_i^v, \mathsf{T}_i - v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}))$.**

  In this case, we show that $\mathsf{s}_i = \mathsf{s}'_i$ holds for all $i \in [n]$. This in turn implies that the output of $\mathsf{Decap}_{\mathsf{cca}}$ and that of $\mathsf{AltDecap}$ agree since these algorithms proceed identically after they respectively compute s.

  Fix $i \in [n]$. The condition of this case directly implies $\mathsf{Decap}(\mathsf{sk}^v, \mathsf{ct}_i^v) = \mathsf{T}_i - v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h})$. This in turn implies that if $v = 0$ then we have $\mathsf{s}_i = 0$, while if $v = 1$ then we have $\mathsf{s}'_i = 1$. In the following, we will show that

  $$\mathsf{k}' := \mathsf{Decap}(\mathsf{sk}^{1-v}, \mathsf{ct}_i^{1-v}) \neq \mathsf{T}_i - (1-v) \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) \tag{5}$$

  holds, which implies that if $v = 0$ then we have $\mathsf{s}'_i = 0$, while if $v = 1$ then we have $\mathsf{s}_i = 1$. Hence, combined together, we will obtain the desired conclusion $\mathsf{s}_i = \mathsf{s}'_i$ (regardless of the value of v). Also, if $\mathsf{k}' = \bot$, then Equation 5 is obviously satisfied. Thus, below we consider the case $\mathsf{k}' \neq \bot$.

  The argument for showing Equation 5 differs depending on whether $i \in \mathcal{S}_{\mathsf{zero}}$ or $i \in \mathcal{S}_{\mathsf{one}}$. If $i \in \mathcal{S}_{\mathsf{zero}}$, then since B is not bad, it is not in the image of $\widehat{g}_{i,v}$. Hence, we have

  $$\begin{aligned} & \mathsf{B} \neq \widehat{g}_{i,v}(\mathsf{r}, \mathsf{ct}_i^{1-v}, \mathsf{h}) = \frac{\Big(\mathsf{T}_i - v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) - \mathsf{k}'\Big) \cdot (-1)^v - \mathsf{k}_i^{*0} + \mathsf{k}_i^{*1}}{\mathsf{h} - \mathsf{h}^*} \\ \iff \quad & \mathsf{k}' \neq \mathsf{T}_i - v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) - (-1)^v \cdot \Big((\mathsf{k}_i^{*0} - \mathsf{k}_i^{*1} - \mathsf{B} \cdot \mathsf{h}^*) + \mathsf{B} \cdot \mathsf{h}\Big) \\ & \overset{(*)}{=} \mathsf{T}_i - \Big(v + (-1)^v\Big) \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) \overset{(**)}{=} \mathsf{T}_i - (1-v) \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}), \end{aligned}$$

  where the equality (*) uses $\mathsf{A}_i = \mathsf{k}_i^{*0} - \mathsf{k}_i^{*1} - \mathsf{B} \cdot \mathsf{h}^*$, which is how A_i is generated for the positions $i \in \mathcal{S}_{\mathsf{zero}}$ in Game 3; The equality (**) is due to $v + (-1)^v = 1 - v$ for $v \in \{0,1\}$.

  Similarly, if $i \in \mathcal{S}_{\mathsf{one}}$, then since A_i is not bad with respect to B, it is not in the image of $g_{\mathsf{B},v}$. Hence, we have

  $$\begin{aligned} & \mathsf{A}_i \neq g_{\mathsf{B},v}(\mathsf{r}, \mathsf{ct}_i^{1-v}, \mathsf{h}) = \Big(\mathsf{T}_i - v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) - \mathsf{k}'\Big) \cdot (-1)^v - \mathsf{B} \cdot \mathsf{h} \\ \iff \quad & \mathsf{k}' \neq \mathsf{T}_i - v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) - (-1)^v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) \\ & = \mathsf{T}_i - \Big(v + (-1)^v\Big) \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) = \mathsf{T}_i - (1-v) \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}), \end{aligned}$$

  where the last equality is again due to $v + (-1)^v = 1 - v$ for $v \in \{0,1\}$.

  We have seen that $\mathsf{Decap}(\mathsf{sk}^{1-v}, \mathsf{ct}_i^{1-v}) \neq \mathsf{T}_i - (1-v) \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h})$ holds regardless of whether $i \in \mathcal{S}_{\mathsf{zero}}$ or $i \in \mathcal{S}_{\mathsf{one}}$, as required. Hence, as mentioned earlier, $\mathsf{s}_i = \mathsf{s}'_i$ holds for all $i \in [n]$, and consequently we have $\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK}, \mathsf{CT}) = \mathsf{AltDecap}(\mathsf{SK}', \mathsf{CT})$.

- **Case 2: There exists a position $i \in [n]$ for which there exists no pair $(\mathsf{r}, v) \in \{0,1\}^\lambda \times \{0,1\}$ satisfying $\mathsf{Encap}(\mathsf{pk}^v; \mathsf{r}) = (\mathsf{ct}_i^v, \mathsf{T}_i - v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}))$.**

  In this case, both $\mathsf{Decap}_{\mathsf{cca}}$ and $\mathsf{AltDecap}$ return $\bot$. Indeed, the condition of this case implies that there exists a position $i \in [n]$ for which there exists no $\mathsf{r} \in \{0,1\}^\lambda$ satisfying $\mathsf{Encap}(\mathsf{pk}^{\mathsf{s}_i}; \mathsf{r}) = (\mathsf{ct}_i^{\mathsf{s}_i}, \mathsf{T}_i - \mathsf{s}_i \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}))$. Hence, the validity check done in the last step of $\mathsf{Decap}_{\mathsf{cca}}$ cannot be satisfied at the position i, and thus $\mathsf{Decap}_{\mathsf{cca}}$ outputs $\bot$. Exactly the same argument applies to $\mathsf{AltDecap}$, and thus it also outputs $\bot$. Hence, in this case we have $\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK}, \mathsf{CT}) = \mathsf{AltDecap}(\mathsf{SK}', \mathsf{CT}) = \bot$.

As seen above, if PK is not bad, then for any CT with $\mathsf{h} \neq \mathsf{h}^*$, we have $\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK}, \mathsf{CT}) = \mathsf{AltDecap}(\mathsf{SK}', \mathsf{CT})$, as desired. $\square$ (**Lemma 3**)

Lemma 4 *There exists a PPT adversary* $\mathcal{B}^4_{\mathsf{cpa}}$ *such that* $|\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| = \mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}^4_{\mathsf{cpa}}}(\lambda)$.

Proof of Lemma 4. The lemma can be shown similarly to Lemma 2. The difference is that the reduction algorithm $\mathcal{B}^4_{\mathsf{cpa}}$ in the proof for this lemma embeds its given public key pk' to pk^0, and uses the given challenge ciphertext/session-key pairs $\{(\mathsf{ct}'^*_i, \mathsf{k}'^*_{i,\beta})\}_{i \in [n]}$ as $(\mathsf{ct}^{*0}_i, \mathsf{k}^{*0}_i)$ in the positions $i \in \mathcal{S}_{\mathsf{one}}$. The remaining values of PK, SK', CT^*, and k_b are generated by $\mathcal{B}^4_{\mathsf{cpa}}$ itself in exactly the same way as in Game 5. (Note that $\mathsf{SK}' = (\mathsf{sk}^1, \mathsf{PK})$, and thus sk^0 is not needed.) Then $\mathcal{B}^4_{\mathsf{cpa}}$ runs $b' \leftarrow \mathcal{A}^{\mathsf{AltDecap}(\mathsf{SK}',\cdot)}(\mathsf{PK}, \mathsf{CT}^*, \mathsf{k}_b)$, and terminates with output $\beta' \leftarrow (b' \overset{?}{=} b)$. $\mathcal{B}^4_{\mathsf{cpa}}$ perfectly simulates Game 4 (resp. Game 5) if its challenge bit β is 0 (resp. 1), and thus we can derive $\mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}^4_{\mathsf{cpa}}}(\lambda) = |\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]|$. $\square$ (**Lemma 4**)

Lemma 5 *There exists a PPT adversary* $\mathcal{B}_{\mathsf{kdm}}$ *that makes a single KDM-encryption query and satisfies* $|\Pr[\mathtt{SUC}_5] - \Pr[\mathtt{SUC}_6]| = \mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{P},\mathcal{B}_{\mathsf{kdm}}}(\lambda)$.

Proof of Lemma 5. Using $\mathcal{A}$ as a building block, we show how to construct a PPT adversary $\mathcal{B}_{\mathsf{kdm}}$ that attacks the one-time $\mathcal{P}$-KDM security of SKE with the claimed advantage. The description of $\mathcal{B}_{\mathsf{kdm}}$ is as follows:

$\mathcal{B}^{\mathcal{O}_{\mathsf{kdm}}(\cdot,\cdot)}_{\mathsf{kdm}}(1^\lambda)$**:** Firstly, $\mathcal{B}_{\mathsf{kdm}}$ picks $\mathsf{r}^{*0}_1, \dots, \mathsf{r}^{*0}_n, \mathsf{r}^{*1}_1, \dots, \mathsf{r}^{*1}_n \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and $\mathsf{k}^*_1 \xleftarrow{\mathsf{r}} \{0,1\}^\ell$. Then, $\mathcal{B}_{\mathsf{kdm}}$ defines f_1 to be the function that takes $z = (z_1, \dots, z_n) \in \{0,1\}^n$ as input and outputs $(\mathsf{r}^{*(z_i)}_i)_{i \in [n]} \| \mathsf{k}^*_1 \in \{0,1\}^{n \cdot \lambda + \ell}$, and f_0 to be the constant zero-function with output length $n \cdot \lambda + \ell$. Note that f_0 and f_1 can be expressed by a projection function (each bit of $\mathsf{r}^{*(z_i)}$ is dependent only on z_i, and k^*_1 is independent of z), and thus $f_0, f_1 \in \mathcal{P}$. $\mathcal{B}_{\mathsf{kdm}}$ submits (f_0, f_1) as a KDM-encryption query, and receives the challenge ciphertext $\mathsf{ct}^*_{\mathsf{SKE}}$. Next, $\mathcal{B}_{\mathsf{kdm}}$ generates the remaining values of PK, SK', CT^*, and k^*_b by itself as in Game 5, where the secret key sk used to generate $\mathcal{B}_{\mathsf{kdm}}$'s challenge ciphertext $\mathsf{ct}^*_{\mathsf{SKE}}$ is regarded as s^* in Game 5. (Recall that in Game 5, values dependent on s^* appear only in the plaintext of $\mathsf{ct}^*_{\mathsf{SKE}}$.) Then, $\mathcal{B}_{\mathsf{kdm}}$ runs $b' \leftarrow \mathcal{A}^{\mathsf{AltDecap}(\mathsf{SK}',\cdot)}(\mathsf{PK}, \mathsf{CT}^*, \mathsf{k}_b)$, and terminates with output $\beta' \leftarrow (b' \overset{?}{=} b)$.

Let $\beta \in \{0,1\}$ be $\mathcal{B}_{\mathsf{kdm}}$'s challenge bit, and we view the secret key sk in $\mathcal{B}_{\mathsf{kdm}}$'s KDM experiment as s^* in the experiment simulated for $\mathcal{A}$ by $\mathcal{B}_{\mathsf{kdm}}$. Then, it is not hard to see that if $\beta = 1$ (resp. $\beta = 0$), then $\mathcal{B}_{\mathsf{kdm}}$ perfectly simulates Game 5 (resp. Game 6) for $\mathcal{A}$. In particular, if $\beta = 1$, then $\mathsf{ct}^*_{\mathsf{SKE}}$ is an encryption of $f_1(\mathsf{s}^*) = (\mathsf{r}^{*(\mathsf{s}^*_i)}_i)_{i \in [n]} \| \mathsf{k}^*_1$, which is exactly how it is generated in Game 5. On the other hand, if $\beta = 0$, then $\mathsf{ct}^*_{\mathsf{SKE}}$ is an encryption of $f_0(\mathsf{s}^*) = 0^{n \cdot \lambda + \ell}$, which is exactly how it is generated in Game 6. Under this situation, the probability that $b' = b$ holds (and thus $\mathcal{B}_{\mathsf{kdm}}$ outputs $\beta' = 1$) is exactly $\Pr[\mathtt{SUC}_5]$ (resp. $\Pr[\mathtt{SUC}_6]$). Hence, we have

$$\mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{P},\mathcal{B}_{\mathsf{kdm}}}(\lambda) = \left|\Pr[\beta' = 1 | \beta = 1] - \Pr[\beta' = 1 | \beta = 0]\right| = |\Pr[\mathtt{SUC}_5] - \Pr[\mathtt{SUC}_6]|,$$

as desired. $\square$ (**Lemma 5**)

Lemma 6 $\Pr[\mathtt{SUC}_6] = 1/2$ *holds.*

Proof of Lemma 6. This lemma is true because in Game 6, the information of the challenge bit b is completely erased from $\mathcal{A}$'s view. $\square$ (**Lemma 6**)

Due to Lemmas 1 to 6 and Equation 3, we can conclude that there exist PPT adversaries $\mathcal{B}_{\mathsf{tcr}}$, $\{\mathcal{B}^j_{\mathsf{cpa}}\}_{j\in[4]}$, $\mathcal{B}'_{\mathsf{cpa}}$, and $\mathcal{B}_{\mathsf{kdm}}$ (that makes a single KDM-encryption query) satisfying Equation 1, as desired.

Finally, we give the proof of Lemma 1.

Proof of Lemma 1. We say that a decapsulation query $\mathsf{CT} = ((\mathsf{ct}^0_i, \mathsf{ct}^1_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ made by $\mathcal{A}$ in Game $j \in [4]$ is *hash-bad* if $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}^0_i, \mathsf{ct}^1_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}) = \mathsf{h}^*$ and $\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK}, \mathsf{CT}) \neq \bot$ hold simultaneously. We categorize a hash-bad decapsulation query into the following two mutually exclusive types:

- Type-1: $(\mathsf{ct}^0_i, \mathsf{ct}^1_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}} \neq (\mathsf{ct}^{*0}_i, \mathsf{ct}^{*1}_i)_{i\in[n]} \| \mathsf{ct}^*_{\mathsf{SKE}}$
- Type-2: $(\mathsf{ct}^0_i, \mathsf{ct}^1_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}} = (\mathsf{ct}^{*0}_i, \mathsf{ct}^{*1}_i)_{i\in[n]} \| \mathsf{ct}^*_{\mathsf{SKE}}$

For $j \in [4]$ and $k \in [2]$, let $\mathtt{HB}^k_i$ be the event that $\mathcal{A}$ makes at least one Type-k hash-bad decapsulation query in Game j. Observe that if pk^0 is not erroneous and $\mathcal{A}$ does not make a hash-bad query, then Game 1 and Game 2 proceed identically. Thus, we have

$$|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \leq \epsilon + \Pr[\mathtt{HB}^1_2] + \Pr[\mathtt{HB}^2_2].$$

Furthermore, note that by definition, a Type-1 hash-bad query can be directly used to break the target collision resistance of Hash. That is, we can construct a PPT adversary $\mathcal{B}_{\mathsf{tcr}}$ satisfying $\Pr[\mathtt{HB}^1_2] = \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{Hash},\mathcal{B}_{\mathsf{tcr}}}(\lambda)$. Since the construction of $\mathcal{B}_{\mathsf{tcr}}$ is straightforward, we omit the detail.

It remains to show how $\Pr[\mathtt{HB}^2_2]$ is bounded. Note that $\mathcal{A}$'s decapsulation queries CT must satisfy $\mathsf{CT} \neq \mathsf{CT}^*$, and thus if CT is a Type-2 hash-bad query, then there must exist a position $j \in [n]$ for which $\mathsf{T}_j \neq \mathsf{T}^*_j$ holds. For a ciphertext $\mathsf{CT} = ((\mathsf{ct}^{*0}_i, \mathsf{ct}^{*1}_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}^*_{\mathsf{SKE}})$ satisfying the conditions of a Type-2 hash-bad query, define $\mathtt{Diff}_{\mathsf{CT}} := \{j \in [n] | \mathsf{T}_j \neq \mathsf{T}^*_j\}$, and for each $i \in [n]$, let $\mathsf{s}_i = 1 - (\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}^{*0}_i) \overset{?}{=} \mathsf{T}_i)$. We observe that for the positions $i \in \mathtt{Diff}_{\mathsf{CT}}$, the following holds. (For now, let us forget about the possibility of pk^0 being erroneous.)

- If $\mathsf{s}^*_i = 0$, then $\mathsf{Encap}(\mathsf{pk}^0; \mathsf{r}^{*0}_i) = (\mathsf{ct}^{*0}_i, \mathsf{T}^*_i)$. This and $\mathsf{T}_i \neq \mathsf{T}^*_i$ imply $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}^{*0}_i) \neq \mathsf{T}_i$, which in turn implies $\mathsf{s}_i = 1$. Then, $\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK}, \mathsf{CT}) \neq \bot$ implies that there must exist $\mathsf{r} \in \{0,1\}^\lambda$ such that $\mathsf{Encap}(\mathsf{pk}^1; \mathsf{r}) = (\mathsf{ct}^{*1}_i, \mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^*) = (\mathsf{ct}^{*1}_i, \mathsf{k}^{*1}_i)$ holds (and this r is recovered during the computation of $\mathsf{Decap}_{\mathsf{cca}}$). Hence, if $\mathcal{A}$ makes a Type-2 hash-bad decapsulation query CT and there exists a position $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{zero}}$, then the query implies the recovery of the session-key k^{*1}_i corresponding to ct^{*1}_i.

- On the other hand, if $\mathsf{s}^*_i = 1$, then $\mathsf{Encap}(\mathsf{pk}^1; \mathsf{r}^{*1}_i) = (\mathsf{ct}^{*1}_i, \mathsf{k}^{*1}_i) = (\mathsf{ct}^{*1}_i, \mathsf{T}^*_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^*)$. We argue that $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}^{*0}_i) = \mathsf{T}_i$ holds, and thus we have $\mathsf{s}_i = 0$. Assume towards a contradiction that $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}^{*0}_i) \neq \mathsf{T}_i$ holds. Then, it implies $\mathsf{s}_i = 1$. In order for $\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK}, \mathsf{CT}) \neq \bot$ to hold, there must exist $\mathsf{r} \in \{0,1\}^\lambda$ such that $\mathsf{Encap}(\mathsf{pk}^1; \mathsf{r}) = (\mathsf{ct}^{*1}_i, \mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^*)$ holds (and this r is recovered during the computation of $\mathsf{Decap}_{\mathsf{cca}}$). However, ct^{*1}_i is by definition an encapsulation of $\mathsf{k}^{*1}_i = \mathsf{T}^*_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^*$. Thus, we have $\mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^* = \mathsf{T}^*_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^*$, which implies $\mathsf{T}_i = \mathsf{T}^*_i$, but it contradicts $\mathsf{T}_i \neq \mathsf{T}^*_i$. Hence, if $\mathsf{s}^*_i = 1$, then we must have $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}^{*0}_i) = \mathsf{k}^{*0}_i = \mathsf{T}_i$. Therefore, if $\mathcal{A}$ makes a Type-2 hash-bad decapsulation query CT and there exists a position $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{one}}$, then the query implies the recovery of the session-key k^{*0}_i corresponding to ct^{*0}_i.

Based on the above observation, we classify a Type-2 hash-bad decapsulation query CT into the following two sub-types:

- Type-2a: There exists a position $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{zero}}$.
- Type-2b: There exists a position $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{one}}$.

For $j \in \{2, 3, 4\}$, let $\mathtt{HB}_j^{2a}$ (resp. $\mathtt{HB}_j^{2b}$) be the event that $\mathcal{A}$ makes at least one Type-2a (resp. Type-2b) hash-bad decapsulation query. By the definition of the events, we have

$$\Pr[\mathtt{HB}_2^2] \leq \Pr[\mathtt{HB}_2^{2a}] + \Pr[\mathtt{HB}_2^{2b}].$$

Hence, it remains to show how $\Pr[\mathtt{HB}_2^{2a}]$ and $\Pr[\mathtt{HB}_2^{2b}]$ are bounded. We first show that there exists a PPT adversary $\mathcal{B}^1_{\mathsf{cpa}}$ that attacks the n-challenge IND-CPA security of KEM and satisfies

$$\Pr[\mathtt{HB}_2^{2a}] \leq \mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}^1_{\mathsf{cpa}}}(\lambda) + n \cdot 2^{-4\lambda}. \tag{6}$$

The description of $\mathcal{B}^1_{\mathsf{cpa}}$ is as follows.

$\mathcal{B}^1_{\mathsf{cpa}}(\mathsf{pk}, (\mathsf{ct}'^*_i, \mathsf{k}'^*_{i,\beta}))$**:** (where $\beta \in \{0, 1\}$ is $\mathcal{B}^1_{\mathsf{cpa}}$'s challenge bit) $\mathcal{B}^1_{\mathsf{cpa}}$ first runs $\mathsf{s}^* = (\mathsf{s}^*_1, \dots, \mathsf{s}^*_n) \leftarrow \mathsf{K}(1^\lambda)$, and sets $\mathsf{pk}^1 \leftarrow \mathsf{pk}'$ and $\mathsf{ct}^{*1}_i \leftarrow \mathsf{ct}'^*_i$ for the positions $i \in \mathcal{S}_{\mathsf{zero}}$. $\mathcal{B}^1_{\mathsf{cpa}}$ then generates the remaining values of PK, SK, CT^*, and k^*_b by itself exactly as in Game 2, and runs $\mathcal{A}^{\mathsf{Decap}_{\mathsf{cca}}(\mathsf{SK},\cdot)}(\mathsf{PK}, \mathsf{CT}^*, \mathsf{k}^*_b)$. (Note that the randomness r^{*1}_i behind $\mathsf{ct}^{*1}_i = \mathsf{ct}'^*_i$ for the positions $i \in \mathcal{S}_{\mathsf{zero}}$ is not needed for generating CT^*.) When $\mathcal{A}$ terminates, $\mathcal{B}^1_{\mathsf{cpa}}$ checks whether $\mathcal{A}$ has made a Type-2a hash-bad decapsulation query CT.

- If a Type-2a hash-bad query has been made, then let $\mathsf{CT} = ((\mathsf{ct}^{*0}_i, \mathsf{ct}^{*1}_i, \mathsf{T}_i)_{i \in [n]}, \mathsf{ct}^*_{\mathsf{SKE}})$ be the found query. (If $\mathcal{A}$ has made multiple Type-2a hash-bad queries, then $\mathcal{B}^1_{\mathsf{cpa}}$ uses the first one.) If there exists a position $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{zero}}$ for which $\mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^* = \mathsf{k}'^*_{i,\beta}$ holds, then $\mathcal{B}^1_{\mathsf{cpa}}$ sets $\beta' \leftarrow 1$, and otherwise $\mathcal{B}'_{\mathsf{cpa}}$ sets $\beta' \leftarrow 0$.
- If no Type-2a hash-bad query has been made, then $\mathcal{B}^1_{\mathsf{cpa}}$ sets $\beta' \leftarrow 0$.

Finally, $\mathcal{B}^1_{\mathsf{cpa}}$ terminates with output β'.

It is straightforward to see that $\mathcal{B}^1_{\mathsf{cpa}}$ simulates Game 2 perfectly for $\mathcal{A}$ (regardless of its challenge bit β). Hence, the probability that $\mathcal{A}$ makes a Type-2a hash-bad query in the experiment simulated by $\mathcal{B}^1_{\mathsf{cpa}}$ is exactly $\Pr[\mathtt{HB}_2^{2a}]$. Next, recall our observation above that if $\mathcal{A}$ makes a Type-2a hash-bad query $\mathsf{CT} = ((\mathsf{ct}^{*0}_i, \mathsf{ct}^{*1}_i, \mathsf{T}_i)_{i \in [n]}, \mathsf{ct}^*_{\mathsf{SKE}})$, then $\mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^* = \mathsf{k}^{*1}_i$ holds for some position $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{zero}}$. Since $\mathcal{B}^1_{\mathsf{cpa}}$ embeds each of its given challenge ciphertexts ct'^*_i as ct^{*1}_i for the positions $i \in \mathcal{S}_{\mathsf{zero}}$, if $\beta = 1$, then we have the correspondence $\mathsf{k}^{*1}_i = \mathsf{k}'^*_{i,1}$ for the positions $i \in \mathcal{S}_{\mathsf{zero}}$. Thus, if $\beta = 1$, then there exists at least one position $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{zero}}$ for which $\mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^* = \mathsf{k}^{*1}_i = \mathsf{k}'^*_{i,1}$ holds, and $\mathcal{B}^1_{\mathsf{cpa}}$ outputs $\beta' = 1$. Hence, we can conclude that $\Pr[\beta' = 1 | \beta = 1] = \Pr[\mathtt{HB}_{\mathcal{B}}^{2a}]$ holds. On the other hand, $\mathsf{k}'^*_{i,0}$ for every $i \in [n]$ is chosen uniformly at random from $\{0, 1\}^{4\lambda}$, and is independent of $\mathcal{A}$'s view. Thus, the probability that there exists $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{zero}}$ for which $\mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^* = \mathsf{k}'^*_{i,0}$ holds (and $\mathcal{B}^1_{\mathsf{cpa}}$ outputs $\beta' = 1$ in case $\beta = 0$), is at most $n \cdot 2^{-4\lambda}$ by the union bound. That is, we have $\Pr[\beta' = 1 | \beta = 0] \leq n \cdot 2^{-4\lambda}$. Hence, we have

$$\mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}^1_{\mathsf{cpa}}}(\lambda) = \left|\Pr[\beta' = 1 | \beta = 1] - \Pr[\beta' = 1 | \beta = 0]\right| \geq \Pr[\mathtt{HB}_2^{2a}] - n \cdot 2^{-4\lambda},$$

which is equivalent to Equation 6.

It remains to bound $\Pr[\mathtt{HB}_2^{2b}]$. By the triangle inequality, we have

$$\Pr[\mathtt{HB}_2^{2b}] \leq \sum_{j \in \{2,3\}} \left| \Pr[\mathtt{HB}_j^{2b}] - \Pr[\mathtt{HB}_j^{2b}] \right| + \Pr[\mathtt{HB}_4^{2b}].$$

Here, with essentially the same argument as in the proof of Lemma 2, we have $\left|\Pr[\mathtt{HB}_2^{2b}] - \Pr[\mathtt{HB}_3^{2b}]\right| = \mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}^2_{\mathsf{cpa}}}(\lambda)$ for an appropriate PPT adversary $\mathcal{B}^2_{\mathsf{cpa}}$. More specifically, the reduction algorithm $\mathcal{B}^2_{\mathsf{cpa}}$ here runs in exactly the same way as $\mathcal{B}^3_{\mathsf{cpa}}$ in the proof of Lemma 2, with the difference that $\mathcal{B}^2_{\mathsf{cpa}}$ outputs 1 if and only if $\mathcal{A}$ has made a Type-2b hash-bad decapsulation query, which can be detected by using sk^0 that $\mathcal{B}^2_{\mathsf{cpa}}$ possesses.

Furthermore, with the same argument as in the proof of Lemma 3, we have $\left|\Pr[\mathtt{HB}_3^{2b}] - \Pr[\mathtt{HB}_4^{2b}]\right| \leq 2\epsilon + n \cdot 2^{-\lambda+1}$.

Finally, we show that there exists a PPT adversary $\mathcal{B}'_{\mathsf{cpa}}$ that attacks the n-challenge IND-CPA security of KEM and satisfies

$$\Pr[\mathtt{HB}_4^{2b}] \leq q_{\mathsf{dec}} \cdot \left(\mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}'_{\mathsf{cpa}}}(\lambda) + n \cdot 2^{-4\lambda}\right). \tag{7}$$

The description of $\mathcal{B}'_{\mathsf{cpa}}$ is somewhat symmetric to $\mathcal{B}^1_{\mathsf{cpa}}$ that we showed above, and is as follows.

$\mathcal{B}'_{\mathsf{cpa}}(\mathsf{pk}', (\mathsf{ct}'^*_i, \mathsf{k}'^*_{i,\beta})_{i\in[n]})$**:** (where $\beta \in \{0,1\}$ is $\mathcal{B}'_{\mathsf{cpa}}$'s challenge bit) $\mathcal{B}'_{\mathsf{cpa}}$ first runs $\mathsf{s}^* = (\mathsf{s}^*_1, \ldots, \mathsf{s}^*_n) \leftarrow \mathsf{K}(1^\lambda)$, and sets $\mathsf{pk}^0 \leftarrow \mathsf{pk}'$ and $\mathsf{ct}^{*0}_i \leftarrow \mathsf{ct}'^*_i$ for the positions $i \in \mathcal{S}_{\mathsf{one}}$. $\mathcal{B}'_{\mathsf{cpa}}$ then generates the remaining values of PK, SK', CT^*, and k^*_b by itself exactly as in Game 4, and runs $\mathcal{A}^{\mathsf{AltDecap}(\mathsf{SK}',\cdot)}(\mathsf{PK}, \mathsf{CT}^*, \mathsf{k}^*_b)$. (Note that the randomness r^{*0}_i behind $\mathsf{ct}^{*0}_i = \mathsf{ct}'^*_i$ for the positions $i \in \mathcal{S}_{\mathsf{one}}$ is not needed for generating CT^*.) When $\mathcal{A}$ terminates, $\mathcal{B}'_{\mathsf{cpa}}$ picks one of $\mathcal{A}$'s decapsulation queries uniformly, and let $\mathsf{CT} = ((\mathsf{ct}^0_i, \mathsf{ct}^1_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ be the chosen query.[4] Then, $\mathcal{B}'_{\mathsf{cpa}}$ checks if there exists a position $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{one}}$ for which $\mathsf{T}_i = \mathsf{k}'^*_{i,\beta}$ holds. If this is the case, $\mathcal{B}'_{\mathsf{cpa}}$ sets $\beta' \leftarrow 1$, otherwise $\mathcal{B}'_{\mathsf{cpa}}$ sets $\beta' \leftarrow 0$. Finally, $\mathcal{B}'_{\mathsf{cpa}}$ terminates with output β'.

As in the analysis of $\mathcal{B}^1_{\mathsf{cpa}}$, it is straightforward to see that $\mathcal{B}'_{\mathsf{cpa}}$ simulates Game 4 perfectly for $\mathcal{A}$ (regardless of its challenge bit β). Hence, the probability that $\mathcal{A}$ makes a Type-2b hash-bad query in the experiment simulated by $\mathcal{B}'_{\mathsf{cpa}}$ is exactly $\Pr[\mathtt{HB}_4^{2b}]$. Furthermore, since $\mathcal{B}'_{\mathsf{cpa}}$ picks one of $\mathcal{A}$'s decapsulation queries uniformly, conditioned on the event that $\mathcal{A}$ makes a Type-2b hash bad query, the probability that $\mathcal{B}'_{\mathsf{cpa}}$ chooses such a query is at least $1/q_{\mathsf{dec}}$. Moreover, recall our observation above that if $\mathcal{A}$ makes a Type-2b hash-bad query $\mathsf{CT} = ((\mathsf{ct}^{*0}_i, \mathsf{ct}^{*1}_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}^*_{\mathsf{SKE}})$, then $\mathsf{T}_i = \mathsf{k}^{*0}_i$ holds for some position $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{one}}$. Since $\mathcal{B}'_{\mathsf{cpa}}$ uses each of its given challenge ciphertexts ct'^*_i as ct^{*0}_i for the positions $i \in \mathcal{S}_{\mathsf{one}}$, if $\beta = 1$, then we have the correspondence $\mathsf{k}^{*0}_i = \mathsf{k}'^*_{i,1}$ for all $i \in \mathcal{S}_{\mathsf{one}}$. Thus, if $\beta = 1$ and $\mathcal{B}'_{\mathsf{cpa}}$ chooses a Type-2b hash-bad query CT, then there exists at least one position $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{one}}$ for which $\mathsf{T}_i = \mathsf{k}^{*0}_i = \mathsf{k}'^*_{i,1}$ holds, and $\mathcal{B}'_{\mathsf{cpa}}$ outputs $\beta' = 1$. Hence, we can conclude that $\Pr[\beta' = 1|\beta = 1] \geq (1/q_{\mathsf{dec}}) \cdot \Pr[\mathtt{HB}_4^{2b}]$ holds. On the other hand, $\mathsf{k}'^*_{i,0}$ for every $i \in [n]$ is chosen uniformly at random from $\{0,1\}^{4\lambda}$, and is independent of $\mathcal{A}$'s view. Thus, the probability that there exists $i \in \mathtt{Diff}_{\mathsf{CT}} \cap \mathcal{S}_{\mathsf{one}}$ for which $\mathsf{T}_i = \mathsf{k}'^*_{i,0}$ holds (and $\mathcal{B}'_{\mathsf{cpa}}$ outputs $\beta' = 1$ in case $\beta = 0$), is at most $n \cdot 2^{-4\lambda}$ by the union bound. That is, we have $\Pr[\beta' = 1|\beta = 0] \leq n \cdot 2^{-4\lambda}$. Hence, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{mcpa}}_{\mathsf{KEM},n,\mathcal{B}'_{\mathsf{cpa}}}(\lambda) &= \left|\Pr[\beta' = 1|\beta = 1] - \Pr[\beta' = 1|\beta = 0]\right| \\ &\geq \frac{1}{q_{\mathsf{dec}}} \cdot \Pr[\mathtt{HB}_4^{2b}] - n \cdot 2^{-4\lambda},\end{aligned}$$

[4]Note that $\mathcal{B}'_{\mathsf{cpa}}$ embeds its given public key pk' to pk^0, and thus it does not have the corresponding secret key sk^0. Hence, it cannot check if $\mathcal{A}$'s decapsulation query satisfies the condition of a Type-2b hash-bad query (checking which requires sk^0). This is the reason why we make $\mathcal{B}'_{\mathsf{cpa}}$ pick $\mathcal{A}$'s query randomly.

which is equivalent to Equation 7.

Putting everything together, we obtain PPT adversaries $\mathcal{B}_{\mathsf{tcr}}$, $\{\mathcal{B}^j_{\mathsf{cpa}}\}_{j\in[2]}$, and $\mathcal{B}'_{\mathsf{cpa}}$ satisfying Equation 4, as required. □ (**Lemma 1**)

□ (**Theorem 2**)

5 Impossibility of Shielding Black-Box Constructions

Gertner et al. [GMM07] showed that there exists no shielding black-box construction of an IND-CCA1 secure PKE scheme from an IND-CPA secure one. (We recall the formal definitions of PKE and its security notions in Appendix A.) Recall that a shielding black-box construction of a PKE scheme $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ from another PKE scheme $\mathsf{pke} = (\mathsf{kg}, \mathsf{enc}, \mathsf{dec})$ is such that the decryption algorithm Dec in PKE does *not* use the encryption algorithm enc of pke. Put differently, we have $\mathsf{PKE}^{\mathsf{pke}} = (\mathsf{KG}^{\mathsf{kg},\mathsf{enc},\mathsf{dec}}, \mathsf{Enc}^{\mathsf{kg},\mathsf{enc},\mathsf{dec}}, \mathsf{Dec}^{\mathsf{kg},\mathsf{dec}})$.

In this section, we extend Gertner et al.'s result and show the following result.

Theorem 4 *There exists no shielding black-box construction of an IND-CCA1 secure PKE scheme from a $\mathcal{P}$-KDM secure PKE scheme.*

This theorem is proved as a corollary of Theorems 5 and 6 stated below.

We emphasize that this result does not contradict our result in Section 4.1 (in particular, Corollary 2), because our construction $\mathsf{KEM}_{\mathsf{cca}}$ is a non-shielding black-box construction in which the decapsulation algorithm $\mathsf{Decap}_{\mathsf{cca}}$ uses the encapsulation algorithm Encap of the underlying IND-CPA secure KEM.

We also note that our result seems incomparable to a similar result by Hajiabadi and Kapron [HK15], who showed that a PKE scheme satisfying a form of *randomness-dependent-message (RDM) security* is a primitive from which a non-shielding black-box construction of an IND-CCA secure PKE scheme is possible while shielding black-box constructions of an IND-CCA1 secure PKE scheme are impossible. (We note that their definition of RDM security is different from the original definition proposed by Birrell, Chung, Pass, and Telang [BCPT13], and tailored to their setting.[5])

Our impossibility of shielding black-box constructions is shown based largely on the framework and technique of [GMM07] and the technique of [HK15]. Informally, [GMM07] defined a distribution $\mathbf{\Phi}$ of an oracle $\mathbf{O} = (\mathbf{O}_1, \mathbf{O}_2)$ such that $\mathbf{O}_1$ syntactically constitutes a PKE scheme, $\mathbf{O}_2$ is an attacker's "breaking" oracle, and they showed that the following two items hold with high probability over the choice of $\mathbf{O} = (\mathbf{O}_1, \mathbf{O}_2) \leftarrow \mathbf{\Phi}$:

1. $\mathbf{O}_1$ constitutes an IND-CPA secure PKE scheme against any computationally unbounded adversary $\mathcal{A}^{\mathbf{O}_1,\mathbf{O}_2}$ that makes polynomially many queries.

2. The IND-CCA1 security of any candidate shielding black-box construction $\mathsf{PKE}^{\mathbf{O}_1}$ is broken (with more than a constant advantage) by some computationally unbounded adversary $\mathcal{A}'^{\mathbf{O}_1,\mathbf{O}_2}$ with polynomially many queries.

[5]Roughly speaking, RDM security used by Hajiabadi and Kapron requires that n ciphertexts encrypting the bit-decomposition of $\mathsf{r} = (\mathsf{r}_1, \dots, \mathsf{r}_n)$ are indistinguishable from n ciphertexts that all encrypt 0 even if they are all encrypted under the same random coin r itself. In the actual definition, an adversary is given multiple sets of the above n ciphertexts. This setting is somewhat unnatural in the usage of PKE, and a PKE scheme satisfying this security notion immediately implies a TDF with one-wayness under correlated products.

These two items imply (via a standard argument used in black-box separation results) the impossibility of shielding black-box constructions of an IND-CCA1 secure PKE scheme from an IND-CPA secure one.

Since we use exactly the same distribution $\mathbf{\Phi}$ of oracles $\mathbf{O}$ used by Gertner et al., and the second item was already shown by them, for our result, we only need to prove an extension of the first item, namely, $\mathbf{O}_1$ constitutes a $\mathcal{P}$-KDM secure PKE scheme with high probability over the choice of $\mathbf{O} = (\mathbf{O}_1, \mathbf{O}_2) \leftarrow \mathbf{\Phi}$.

In the following, we first recall the definition of the distribution $\mathbf{\Phi}$ of oracles $\mathbf{O}$ used by Gertner et al., then state their result corresponding to the item 2 above. Finally, we state our result corresponding to the item 1 above.

Definition 8 (Oracle Distribution for Separation [GMM07]) *Consider an oracle* $\mathbf{O}$ *consisting of the suboracles* $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$ *that are defined for each length parameter* $n \in \mathbb{N}$ *and satisfy the following syntax*[6]*:*

$\mathbf{g} : \{0,1\}^n \to \{0,1\}^{3n}$**:** *This is an injective function. This oracle can be thought of as the key generation process that takes a secret key* $\mathsf{sk} \in \{0,1\}^n$ *as input and outputs a public key* $\mathsf{pk} \in \{0,1\}^{3n}$.

$\mathbf{e} : \{0,1\}^{3n} \times \{0,1\} \times \{0,1\}^n \to \{0,1\}^{3n}$**:** *For each* $\mathsf{pk} \in \{0,1\}^{3n}$, $\mathbf{e}(\mathsf{pk}, \cdot, \cdot) : \{0,1\} \times \{0,1\}^n \to \{0,1\}^{3n}$ *is an injective function. This oracle can be thought of as the encryption process that takes a public key* $\mathsf{pk} \in \{0,1\}^{3n}$*, a plaintext* $\mathsf{m} \in \{0,1\}$*, and a randomness* $\mathsf{r} \in \{0,1\}^n$ *as input, and outputs a ciphertext* $\mathsf{ct} \in \{0,1\}^{3n}$.

$\mathbf{d} : \{0,1\}^n \times \{0,1\}^{3n} \to \{0,1,\perp\}$**:** *This oracle takes* $\mathsf{sk} \in \{0,1\}^n$ *and* $\mathsf{ct} \in \{0,1\}^{3n}$ *as input, and if there exists* $(\mathsf{pk}, \mathsf{m}, \mathsf{r}) \in \{0,1\}^{3n} \times \{0,1\} \times \{0,1\}^n$ *such that* $\mathsf{pk} = \mathbf{g}(\mathsf{sk})$ *and* $\mathsf{ct} = \mathbf{e}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$*, then it outputs* m*. Otherwise, this oracle outputs* $\perp$*. This oracle can be thought of as the decryption process.*

$\mathbf{w} : \{0,1\}^{3n} \times \{0,1\}^n \to \{0,1\}^{3n \times n} \cup \{\perp\}$**:** *This oracle is associated with a "randomness-deriving" function* $F_{\mathbf{w}} : \{0,1\}^{3n} \times \{0,1\}^n \to \{0,1\}^{n \times n}$.[7] *This oracle takes* $\mathsf{pk} \in \{0,1\}^{3n}$ *and an index* $z \in \{0,1\}^n$ *as input, and if there exists no* $\mathsf{sk} \in \{0,1\}^n$ *such that* $\mathsf{pk} = \mathbf{g}(\mathsf{sk})$*, then the oracle outputs* $\perp$*. Otherwise, let* $\mathsf{sk} = (\mathsf{s}_1, \dots, \mathsf{s}_n) \in \{0,1\}^n$ *be such that* $\mathsf{pk} = \mathbf{g}(\mathsf{sk})$*. The oracle computes* $(\mathsf{r}_1, \dots, \mathsf{r}_n) \leftarrow F_{\mathbf{w}}(\mathsf{pk}, z)$*, and then* $\mathsf{ct}_i \leftarrow \mathbf{e}(\mathsf{pk}, \mathsf{s}_i, \mathsf{r}_i)$ *for every* $i \in [n]$*. Finally, the oracle outputs* $(\mathsf{ct}_i)_{i \in [n]}$*. This oracle is a "weakening" oracle that helps breaking the IND-CCA1 security of any shielding construction.*

$\mathbf{u} : \{0,1\}^{3n} \times \{0,1\}^{3n} \to \{\top, \perp\}$**:** *This oracle takes* $\mathsf{pk} \in \{0,1\}^{3n}$ *and* $\mathsf{ct} \in \{0,1\}^{3n}$ *as input, and if there exists* $(\mathsf{sk}, \mathsf{m}, \mathsf{r}) \in \{0,1\}^n \times \{0,1\} \times \{0,1\}^n$ *such that* $\mathsf{pk} = \mathbf{g}(\mathsf{sk})$ *and* $\mathsf{ct} = \mathbf{e}(\mathsf{pk}, \mathsf{m}, \mathsf{r})$*, then the oracle outputs* $\top$*. Otherwise, the oracle outputs* $\perp$*. This oracle can be thought of as the validity checking process of a ciphertext* ct *with respect to a public key* pk.

We define the distribution $\mathbf{\Phi}$ *of an oracle* $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$ *as follows: For each* $n \in \mathbb{N}$*, pick* $\mathbf{g}$*,* $\mathbf{e}$*, and* $F_{\mathbf{w}}$ *uniformly at random, and then define* $\mathbf{d}$*,* $\mathbf{w}$*, and* $\mathbf{u}$ *satisfying the above syntax.*[8]

Note that $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ in $\mathbf{O}$ naturally constitutes a 1-bit PKE scheme. Gertner et al. [GMM07] showed the following result, which states that with high probability over $\mathbf{O} \leftarrow \mathbf{\Phi}$, the IND-CCA1

[6]Among $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$, $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ (resp. $(\mathbf{w}, \mathbf{u})$) corresponds to $\mathbf{O}_1$ (resp. $\mathbf{O}_2$) in the above explanation.

[7]The purpose of $F_{\mathbf{w}}$ is to make $\mathbf{w}$ deterministic (after chosen according to the distribution $\mathbf{\Phi}$). When an oracle $\mathbf{O}$ is chosen from $\mathbf{\Phi}$, $F_{\mathbf{w}}$ will work as a truly random function. This treatment is done implicitly in [GMM07].

[8]Note that the behavior of $\mathbf{O}$ is completely determined by $\mathbf{g}$, $\mathbf{e}$, and $F_{\mathbf{w}}$ used in $\mathbf{w}$.

security of any candidate shielding black-box construction from the PKE $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ (defined in $\mathbf{O}$) is broken with more than a constant advantage by some adversary making polynomially many queries.

Theorem 5 (Corollary of Theorem 2 in [GMM07]) *Let* $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *be a shielding construction of a* 1*-bit PKE scheme based on another* 1*-bit PKE scheme. For each* $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u}) \in \mathsf{Sup}(\mathbf{\Phi})$*, let* $\mathsf{PKE}^{\mathbf{g},\mathbf{e},\mathbf{d}} := (\mathsf{KG}^{\mathbf{g},\mathbf{e},\mathbf{d}}, \mathsf{Enc}^{\mathbf{g},\mathbf{e},\mathbf{d}}, \mathsf{Dec}^{\mathbf{g},\mathbf{d}})$*. Then, there exists a computationally unbounded adversary* $\mathcal{A}$ *that makes at most polynomially many queries and satisfies the following for all sufficiently large* $\lambda \in \mathbb{N}$*:*

$$\Pr_{\mathbf{O}=(\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{w},\mathbf{u})\leftarrow\mathbf{\Phi}}\left[\mathsf{Adv}^{\mathsf{cca1}}_{\mathsf{PKE}^{\mathbf{g},\mathbf{e},\mathbf{d}},\mathcal{A}^{\mathbf{O}}}(\lambda) \geq \frac{1}{2}\right] \geq 1 - \frac{4}{\lambda}.$$

We now show our theorem, which states that with overwhelming probability over the choice of $\mathbf{O} \leftarrow \mathbf{\Phi}$, $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ constitutes a 1-bit $\mathcal{P}$-KDM secure PKE scheme (secure in the presence of multiple KDM-encryption queries). Since the bit-by-bit encryption preserves $\mathcal{P}$-KDM security, the existence of a 1-bit (many-time) $\mathcal{P}$-KDM secure PKE scheme implies a $\mathcal{P}$-KDM secure PKE scheme that can encrypt plaintexts of arbitrary length in the black-box sense.

Theorem 6 *For any computationally unbounded adversary* $\mathcal{A}$ *that makes at most polynomially many queries, there exist negligible functions* $\mu(\cdot)$ *and* $\mu'(\cdot)$ *such that for all sufficiently large* $\lambda \in \mathbb{N}$*, we have*

$$\Pr_{\mathbf{O}=(\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{w},\mathbf{u})\leftarrow\mathbf{\Phi}}\left[\mathsf{Adv}^{\mathsf{kdm}}_{(\mathbf{g},\mathbf{e},\mathbf{d}),\mathcal{P},\mathcal{A}^{\mathbf{O}}}(\lambda) \leq \mu(\lambda)\right] \geq 1 - \mu'(\lambda).$$

We remark that Theorems 5 and 6 imply Theorem 4 via a standard technique in black-box separation results (using the Borel-Cantelli lemma) (see, e.g. [HR04]).

Proof of Theorem 6. Let $\mathcal{A}$ be any computationally unbounded adversary that makes $q = q(\lambda)$ queries (that is the total of queries to $\mathbf{O}$ and to the KDM-encryption oracle $\mathcal{O}_{\mathsf{kdm}}$). We will show that

$$\left|\Pr\left[\mathcal{A}^{\mathbf{O},\mathcal{O}_{\mathsf{kdm}}}(\mathsf{pk}^* = \mathbf{g}(\mathsf{sk}^*)) = b\right] - \frac{1}{2}\right| \leq \frac{\mathsf{poly}(q)}{2^{\lambda}}, \tag{8}$$

where the probability is over $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u}) \leftarrow \mathbf{\Phi}$, $\mathsf{sk}^* \xleftarrow{\mathsf{r}} \{0,1\}^{\lambda}$, $b \xleftarrow{\mathsf{r}} \{0,1\}$, and the randomness used by $\mathcal{A}$ and the KDM-encryption oracle $\mathcal{O}_{\mathsf{kdm}}$. (Recall that $\mathcal{O}_{\mathsf{kdm}}$ takes two projection functions $f_0, f_1 : \{0,1\}^{\lambda} \to \{0,1\}$ as input, picks $\mathsf{r} \xleftarrow{\mathsf{r}} \{0,1\}^{\lambda}$, and returns $\mathsf{ct} = \mathbf{e}(\mathsf{pk}^*, f_b(\mathsf{sk}^*), \mathsf{r})$.) Note that Equation 8 implies the theorem by a simple averaging argument. We call the KDM experiment including the choice of the oracle $\mathbf{O} \leftarrow \mathbf{\Phi}$ the *extended KDM experiment*.

First, we make some assumptions for simplifying the proof.

- Without loss of generality, we can assume that $\mathcal{A}$ never makes the same query twice to $\mathbf{O}$, since each suboracle in $\mathbf{O}$ is deterministic.
- Before making a $\mathbf{d}$-query $(\mathsf{sk}, *)$, $\mathcal{A}$ first makes a $\mathbf{g}$-query sk.
- By definition, $\mathbf{O}$ for each length parameter $n \in \mathbb{N}$ is independent of one another, and only $n = \lambda$ is relevant to Equation 8. Hence, for simplicity, we can assume that $\mathcal{A}$ queries to $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$ only for the length parameter $n = \lambda$. (Queries with other length parameters $n \neq \lambda$ can be simulated by $\mathcal{A}$ itself.)

In the extended KDM experiment, let $\mathtt{SUC}_{\mathcal{A}}$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit b, and let $\mathtt{Q}_{\mathcal{A}}$ be the event that $\mathcal{A}$ submits sk^* as a $\mathbf{g}$-query. By definition, the left hand side of Equation 8 is $|\Pr[\mathtt{SUC}_{\mathcal{A}}] - 1/2|$.

For showing Equation 8, we first show that queries to $\mathbf{d}$, $\mathbf{w}$, and $\mathbf{u}$ do not help $\mathcal{A}$ much. More specifically, we show that for $\mathcal{A}^{\mathbf{O},\mathcal{O}_{\mathsf{kdm}}}$, there exists another adversary $\mathcal{B}^{\mathbf{g},\mathbf{e},\mathcal{O}_{\mathsf{kdm}}}$ that does not have access to $(\mathbf{d},\mathbf{w},\mathbf{u})$, makes at most $q' = \mathsf{poly}(q)$ queries in total, and satisfies

$$\left|\Pr[\mathtt{SUC}_{\mathcal{A}}] - \frac{1}{2}\right| \leq \left|\Pr[\mathtt{SUC}_{\mathcal{B}}] - \frac{1}{2}\right| + \frac{1}{2}\Pr[\mathtt{Q}_{\mathcal{B}}] + \frac{\mathsf{poly}(q)}{2^{\lambda}}, \tag{9}$$

where $\mathtt{SUC}_{\mathcal{B}}$ (resp. $\mathtt{Q}_{\mathcal{B}}$) is the event that $\mathcal{B}$ succeeds in guessing the challenge bit b (resp. makes a $\mathbf{g}$-query sk^*) in its extended KDM experiment.

To this end, we need some preparations. We introduce the following notation in the extended KDM experiment: Let $\mathsf{L}_{\mathtt{PK}}$ be the set consisting of pk^* and public keys pk that $\mathcal{A}$ has obtained by making a $\mathbf{g}$-query $\mathsf{sk} \in \{0,1\}^{\lambda}$. Similarly, let $\mathsf{L}_{\mathtt{PK\&CT}}$ be the set of all public key/ciphertext pairs $(\mathsf{pk},\mathsf{ct})$ that $\mathcal{A}$ has obtained by making an $\mathbf{e}$-query $(\mathsf{pk},\mathsf{m},\mathsf{r})$, a $\mathbf{w}$-query (pk,z), or a KDM-encryption query. Furthermore, we introduce the notion of *surprising* queries that could be made by $\mathcal{A}$ during the extended KDM experiment.

- A $\mathbf{d}$-query $(\mathsf{sk},\mathsf{ct})$ is said to be surprising if $(\mathbf{g}(\mathsf{sk}),\mathsf{ct}) \notin \mathsf{L}_{\mathtt{PK\&CT}}$ and ct belongs to the image of $\mathbf{e}(\mathsf{pk},\cdot,\cdot)$.
- A $\mathbf{w}$-query $(\mathsf{pk},*)$ is said to be surprising if $\mathsf{pk} \notin \mathsf{L}_{\mathtt{PK}}$ and pk belongs to the image of $\mathbf{g}(\cdot)$.
- A $\mathbf{u}$-query $(\mathsf{pk},\mathsf{ct})$ is said to be surprising if either (1) $\mathsf{pk} \notin \mathsf{L}_{\mathtt{PK}}$ and pk belongs to the image of $\mathbf{g}(\cdot)$, or (2) $(\mathsf{pk},\mathsf{ct}) \notin \mathsf{L}_{\mathtt{PK\&CT}}$ and ct belongs to the image of $\mathbf{e}(\mathsf{pk},\cdot,\cdot)$.

Now, we show the description of $\mathcal{B}^{\mathbf{g},\mathbf{e},\mathcal{O}_{\mathsf{kdm}}}$ as follows.

Given pk^* as input, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pk}^*)$. Then, $\mathcal{B}$ answers $\mathcal{A}$'s queries as follows.

- For a $\mathbf{g}$-query sk, an $\mathbf{e}$-query $(\mathsf{pk},\mathsf{m},\mathsf{r})$, and a KDM-encryption query (f_0,f_1), $\mathcal{B}$ uses its own corresponding oracles for answering them, except that if $\mathcal{B}$'s $\mathbf{g}$-query results in pk^*, then $\mathcal{B}$ stops the simulation, and terminates with a fair coin b'.
- For a $\mathbf{d}$-query $(\mathsf{sk},\mathsf{ct})$, $\mathcal{B}$ proceeds as follows. (Note that it is guaranteed that $\mathsf{pk} = \mathbf{g}(\mathsf{sk}) \in \mathsf{L}_{\mathtt{PK}} \backslash \{\mathsf{pk}^*\}$ holds, because we are assuming that sk must be previously asked as a $\mathbf{g}$-query and $\mathcal{B}$ must have stopped the simulation when it detects $\mathbf{g}(\mathsf{sk}) = \mathsf{pk}^*$.)
  - If $(\mathsf{pk}^*,\mathsf{ct}) \in \mathsf{L}_{\mathtt{PK\&CT}}$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
  - If $(\mathsf{pk},\mathsf{ct}) \notin \mathsf{L}_{\mathtt{PK\&CT}}$, then $\mathcal{B}$ again returns $\perp$ to $\mathcal{A}$.
  - If $(\mathsf{pk},\mathsf{ct}) \in \mathsf{L}_{\mathtt{PK\&CT}}$ and $\mathsf{pk} \neq \mathsf{pk}^*$, then due to how $\mathcal{B}$ answers $\mathcal{A}$'s $\mathbf{w}$-queries (described below), it is guaranteed that ct is an answer to one of $\mathcal{A}$'s previous $\mathbf{e}$-queries $(\mathsf{pk},\mathsf{m},\mathsf{r})$ for some $(\mathsf{m},\mathsf{r}) \in \{0,1\} \times \{0,1\}^{\lambda}$.[9] Thus, $\mathcal{B}$ returns this m to $\mathcal{A}$.

  $\mathcal{B}$'s answer could be incorrect only when the query falls into the second case and $\mathcal{A}$'s query is surprising.
- For a $\mathbf{w}$-query (pk,z), $\mathcal{B}$ proceeds as follows:

[9] We note that ct in this case is also not a ciphertext returned as an answer to $\mathcal{A}$'s KDM-encryption query, because by definition $\mathcal{O}_{\mathsf{kdm}}$ could return ciphertexts only under pk^*.

- If $\mathsf{pk} = \mathsf{pk}^*$, then for each $i \in [\lambda]$, $\mathcal{B}$ defines the function $f^{(i)} : \{0,1\}^\lambda \to \{0,1\}$ that always outputs the i-th bit of its input, and submits a KDM-encryption query (f_0, f_1) with $f_0 = f_1 = f^{(i)}$ to obtain a ciphertext ct_i. (Note that by the definition of $\mathcal{O}_{\mathsf{kdm}}$, each ct_i is $\mathsf{ct}_i = \mathbf{e}(\mathsf{pk}^*, \mathsf{s}_i^*, \mathsf{r}_i)$ for a fresh $\mathsf{r}_i \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$.) Then, $\mathcal{B}$ returns $(\mathsf{ct}_i)_{i\in[\lambda]}$ to $\mathcal{A}$.[10]
- If $\mathsf{pk} \in \mathsf{L}_{\mathsf{PK}} \backslash \{\mathsf{pk}^*\}$, then let $\mathsf{sk} = (\mathsf{s}_1, \dots, \mathsf{s}_\lambda)$ be the secret key corresponding to pk that $\mathcal{A}$ has previously asked as a $\mathbf{g}$-query. For each $i \in [\lambda]$, $\mathcal{B}$ picks a fresh $\mathsf{r}_i \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, makes an $\mathbf{e}$-query $(\mathsf{pk}, \mathsf{s}_i, \mathsf{r})$, and receives ct_i. Then, $\mathcal{B}$ returns $(\mathsf{ct}_i)_{i\in[\lambda]}$ to $\mathcal{A}$.
- If $\mathsf{pk} \notin \mathsf{L}_{\mathsf{PK}}$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.

The distributions of the answers that $\mathcal{B}$ returns to $\mathcal{A}$ for the first two cases are perfect, and only the last case could cause a simulation error (in case the query is surprising).

- For a $\mathbf{u}$-query $(\mathsf{pk}, \mathsf{ct})$, if $\mathsf{pk} \in \mathsf{L}_{\mathsf{PK}}$ and $(\mathsf{pk}, \mathsf{ct}) \in \mathsf{L}_{\mathsf{PK\&CT}}$, then $\mathcal{B}$ returns $\top$ to $\mathcal{A}$. Otherwise (i.e. $\mathsf{pk} \notin \mathsf{L}_{\mathsf{PK}}$ or $(\mathsf{pk}, \mathsf{ct}) \notin \mathsf{L}_{\mathsf{PK\&CT}}$), $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.

  $\mathcal{B}$'s response could be incorrect only when the query falls into the second case and $\mathcal{A}$'s query is surprising.

When $\mathcal{A}$ terminates with output b', $\mathcal{B}$ also outputs b' and terminates.

Note that the number of queries made by $\mathcal{B}$ is at most $q' = \mathsf{poly}(q)$. As explained above, $\mathcal{B}$ simulates the extended KDM experiment perfectly for $\mathcal{A}$ unless $\mathcal{A}$ makes a surprising query and unless $\mathcal{A}$ submits sk^* as a $\mathbf{g}$-query. Recall that in the extended KDM experiment, $\mathbf{g} : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ and $\mathbf{e}(\mathsf{pk}, \cdot, \cdot) : \{0,1\} \times \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ for each $\mathsf{pk} \in \{0,1\}^{3\lambda}$ are random (almost) length-tripling injective functions. Hence, the probability that $\mathcal{A}$ makes a surprising query during the experiment is bounded by $\mathsf{poly}(q) \cdot 2^{-\lambda}$. Also, if $\mathcal{A}$ does not submit sk^* as a $\mathbf{g}$-query, then $\mathcal{B}$ uses $\mathcal{A}$'s output as it is, while if $\mathcal{A}$ does so, then $\mathcal{B}$ outputs a random bit. These imply that we have

$$\left|\Pr[\mathtt{SUC}_{\mathcal{A}} \wedge \overline{\mathtt{Q}_{\mathcal{A}}}] - \Pr[\mathtt{SUC}_{\mathcal{B}} \wedge \overline{\mathtt{Q}_{\mathcal{B}}}]\right| \le \frac{\mathsf{poly}(q)}{2^\lambda}, \tag{10}$$

$$\left|\Pr[\mathtt{Q}_{\mathcal{A}}] - \Pr[\mathtt{Q}_{\mathcal{B}}]\right| \le \frac{\mathsf{poly}(q)}{2^\lambda}, \tag{11}$$

$$\Pr[\mathtt{SUC}_{\mathcal{B}}] = \Pr[\mathtt{SUC}_{\mathcal{B}} \wedge \overline{\mathtt{Q}_{\mathcal{B}}}] + \frac{1}{2}\Pr[\mathtt{Q}_{\mathcal{B}}]. \tag{12}$$

Finally, notice that

$$\left|\Pr[\mathtt{SUC}_{\mathcal{A}}] - \frac{1}{2}\right| \le \left|\Pr[\mathtt{SUC}_{\mathcal{A}} \wedge \overline{\mathtt{Q}_{\mathcal{A}}}] + \frac{1}{2}\Pr[\mathtt{Q}_{\mathcal{A}}] - \frac{1}{2}\right| + \frac{1}{2}\Pr[\mathtt{Q}_{\mathcal{A}}]. \tag{13}$$

Combining Equations 10 to 13, we obtain Equation 9, as desired.

It remains to show how $|\Pr[\mathtt{SUC}_{\mathcal{B}}] - 1/2|$ and $\Pr[\mathtt{Q}_{\mathcal{B}}]$ are bounded. To show this, consider a modified experiment where $\mathcal{O}_{\mathsf{kdm}}$ never uses the same randomness r twice. The statistical distance between the distributions of $\mathcal{B}$'s view in the original and modified experiments is bounded by $q'^2 \cdot 2^{-\lambda} = \mathsf{poly}(q) \cdot 2^{-\lambda}$. Furthermore, in the modified experiment, unless $\mathcal{B}$ submits an $\mathbf{e}$-query $(\mathsf{pk}^*, *, \mathsf{r})$ that contains one of the randomnesses r used in $\mathcal{O}_{\mathsf{kdm}}$, $\mathcal{O}_{\mathsf{kdm}}$ is perfectly indistinguishable from an oracle that always returns a random value $\mathsf{ct} \in \{0,1\}^{3\lambda}$

[10]In the extended KDM experiment, $F_{\mathbf{w}}$ is a random function and $\mathcal{A}$ is assumed to never repeat a query. Thus, we can identify the computation $(\mathsf{r}_1, \dots, \mathsf{r}_\lambda) \leftarrow F_{\mathbf{w}}(\mathsf{pk}, z)$ done in $\mathbf{w}$ with picking $\mathsf{r}_1, \dots, \mathsf{r}_\lambda \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$. Hence, $\mathcal{B}$'s simulation of the response to a $\mathbf{w}$-query (pk, z) with $\mathsf{pk} = \mathsf{pk}^*$ is perfect.

(without reusing the same element twice), which is independent of the challenge bit b or sk^*, in which case $\mathcal{B}$ will have zero advantage and it can make a $\mathbf{g}$-query sk^* only with probability $q' \cdot 2^{-\lambda} = \mathsf{poly}(q) \cdot 2^{-\lambda}$. Finally, since $\mathbf{e}(\mathsf{pk}^*, \cdot, \cdot)$ is a random injective function and thus its output does not reveal any information on preimages, the probability that $\mathcal{B}$ can submit an $\mathbf{e}$-query that contains one of the randomnesses used in $\mathcal{O}_{\mathsf{kdm}}$ is bounded by $\mathsf{poly}(q') \cdot 2^{-\lambda} = \mathsf{poly}(q) \cdot 2^{-\lambda}$. Hence, we can conclude that $|\Pr[\mathtt{SUC}_{\mathcal{B}}] - 1/2| \leq \mathsf{poly}(q) \cdot 2^{-\lambda}$ and $\Pr[\mathtt{Q}_{\mathcal{B}}] \leq \mathsf{poly}(q) \cdot 2^{-\lambda}$. Combining these inequalities with Equation 9, we obtain Equation 8, as desired. This completes the proof.

☐ (**Theorem 6**)

6 TDF via KDM Security

In this section, we show our proposed TDF with adaptive one-wayness, which is an extension of our IND-CCA secure KEM presented in Section 4.

Specifically, in Section 6.1, we resent the formal description of our proposed TDF, state theorems regarding correctness and security, and give several remarks on the properties of our construction. Then, in Sections 6.2 and 6.3, we prove the correctness and adaptive one-wayness of our proposed construction, respectively.

6.1 Our Construction

Let $\ell = \ell(\lambda)$ be a polynomial. Our TDF uses the building blocks KEM, SKE, and Hash with the following properties:

- $\mathsf{KEM} = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ is a KEM such that (1) its session key space is $\{0,1\}^{3\lambda}$, (2) the randomness space of Encap is $\{0,1\}^{\lambda}$, and (3) the ciphertext space $\mathcal{C}$ forms an abelian group (where we use the additive notation) and satisfies $|\mathcal{C}| \geq 2^{2\lambda}$.

- $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ is an SKE scheme such that (1) it has the randomness-recovering decryption property (with the randomness-recovering decryption algorithm RD), (2) its secret key space is $\{0,1\}^n$ for some polynomial $n = n(\lambda)$, and (3) the plaintext space is $\{0,1\}^{n \cdot \lambda + \ell}$.

  We denote the randomness space of E by $\mathcal{R}_{\mathsf{SKE}}$.

- $\mathsf{Hash} = (\mathsf{HKG}, \mathsf{H})$ is a keyed hash function such that the range of H is $\{0,1\}^{\lambda}$, which we are going to assume to be target collision resistant.

  (The formal definition of a target collision resistant hash function is recalled in Appendix A.1.)

Using these building blocks, the proposed TDF $\mathsf{TDF} = (\mathsf{Setup}, \mathsf{Samp}, \mathsf{Eval}, \mathsf{Inv})$ is constructed as described in Figure 4. The domain $\mathcal{X}$ of TDF is $\mathcal{X} = \{0,1\}^n \times \{0,1\}^{n \cdot \lambda} \times \{0,1\}^{\ell} \times \mathcal{R}_{\mathsf{SKE}}$.

For the correctness and security of TDF, the following theorems hold.

Theorem 7 *Let $\epsilon = \epsilon(\lambda) \in [0,1]$. If* KEM *is ϵ-almost-all-keys correct and* SKE *has the randomness-recovering decryption property, then* TDF *is $(\epsilon + n \cdot 2^{-\lambda})$-almost-all-keys correct.*

Theorem 8 *Assume that* KEM *satisfies the pseudorandom ciphertext property and almost-all-keys correctness,* SKE *is one-time $\mathcal{P}$-KDM secure, and* Hash *is target collision resistant. Then,* TDF *is adaptively one-way.*

The proofs of Theorems 7 and 8 are given in Sections 6.2 and 6.3, respectively. Other than using additional functionality/security properties of the building blocks, the proofs for the above theorems go similarly to those for our IND-CCA secure KEM in Section 4.

```
$\mathsf{Setup}(1^\lambda)$ :
  $\forall v \in \{0,1\} : (\mathsf{pk}^v, \mathsf{sk}^v) \leftarrow \mathsf{KKG}(1^\lambda)$
  $\mathsf{A}_1, \dots, \mathsf{A}_n, \mathsf{B} \xleftarrow{\mathsf{r}} \{0,1\}^{3\lambda}$
  $\mathsf{C}_1, \dots, \mathsf{C}_n \xleftarrow{\mathsf{r}} \mathcal{C}$
  $\mathsf{hk} \leftarrow \mathsf{HKG}(1^\lambda)$
  $\mathsf{ek} \leftarrow (\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i, \mathsf{C}_i)_{i \in [n]}, \mathsf{B}, \mathsf{hk})$
  $\mathsf{td} \leftarrow (\mathsf{sk}^0, \mathsf{ek})$
  Return $(\mathsf{ek}, \mathsf{td})$.
```

```
$\mathsf{Samp}(1^\lambda)$ :
  $\mathsf{s} = (\mathsf{s}_1, \dots, \mathsf{s}_n) \leftarrow \mathsf{K}(1^\lambda)$
  $\mathsf{r}_1^{\mathsf{s}_1}, \dots, \mathsf{r}_n^{\mathsf{s}_n} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$
  $\mathsf{k} \xleftarrow{\mathsf{r}} \{0,1\}^\ell$
  Sample $\mathsf{r}_{\mathsf{SKE}} \in \mathcal{R}_{\mathsf{SKE}}$
                  in the same way as in $\mathsf{E}$.
  Return $\mathsf{x} \leftarrow (\mathsf{s}, (\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]}, \mathsf{k}, \mathsf{r}_{\mathsf{SKE}})$.
```

```
$\mathsf{Eval}(\mathsf{ek}, \mathsf{x})$ :
  $(\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i, \mathsf{C}_i)_{i \in [n]}, \mathsf{B}, \mathsf{hk}) \leftarrow \mathsf{ek}$
  $(\mathsf{s} = (\mathsf{s}_1, \dots, \mathsf{s}_n), (\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]}, \mathsf{k}, \mathsf{r}_{\mathsf{SKE}}) \leftarrow \mathsf{x}$
  $\mathsf{ct}_{\mathsf{SKE}} \leftarrow \mathsf{E}(\mathsf{s}, (\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]} \| \mathsf{k}; \mathsf{r}_{\mathsf{SKE}})$
  $\forall i \in [n]$ :
    $(\mathsf{ct}_i^{\mathsf{s}_i}, \mathsf{k}_i^{\mathsf{s}_i}) \leftarrow \mathsf{Encap}(\mathsf{pk}^{\mathsf{s}_i}; \mathsf{r}_i^{\mathsf{s}_i})$
    $\mathsf{ct}_i \leftarrow \mathsf{ct}_i^{\mathsf{s}_i} + \mathsf{s}_i \cdot \mathsf{C}_i \quad ^{(\ddagger)}$
      $= \begin{cases} \mathsf{ct}_i^0 & \text{if } \mathsf{s}_i = 0 \\ \mathsf{ct}_i^1 + \mathsf{C}_i & \text{if } \mathsf{s}_i = 1 \end{cases}$
  $\mathsf{h} \leftarrow \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i)_{i \in [n]} \| \mathsf{ct}_{\mathsf{SKE}})$
  $\forall i \in [n]$ :
    $\mathsf{T}_i \leftarrow \mathsf{k}_i^{\mathsf{s}_i} + \mathsf{s}_i \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) \quad ^{(\dagger)}$
      $= \begin{cases} \mathsf{k}_i^0 & \text{if } \mathsf{s}_i = 0 \\ \mathsf{k}_i^1 + \mathsf{A}_i + \mathsf{B} \cdot \mathsf{h} & \text{if } \mathsf{s}_i = 1 \end{cases}$
  Return $\mathsf{y} \leftarrow ((\mathsf{ct}_i, \mathsf{T}_i)_{i \in [n]}, \mathsf{ct}_{\mathsf{SKE}})$.
```

```
$\mathsf{Inv}(\mathsf{td}, \mathsf{y})$ :
  $(\mathsf{sk}^0, \mathsf{ek}) \leftarrow \mathsf{td}$
  $(\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i, \mathsf{C}_i)_{i \in [n]}, \mathsf{B}, \mathsf{hk}) \leftarrow \mathsf{ek}$
  $((\mathsf{ct}_i, \mathsf{T}_i)_{i \in [n]}, \mathsf{ct}_{\mathsf{SKE}}) \leftarrow \mathsf{y}$
  $\mathsf{h} \leftarrow \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i)_{i \in [n]} \| \mathsf{ct}_{\mathsf{SKE}})$
  $\forall i \in [n]$ :
    $\mathsf{s}_i \leftarrow 1 - (\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) \overset{?}{=} \mathsf{T}_i) \quad ^{(\star)}$
      $= \begin{cases} 0 & \text{if } \mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) = \mathsf{T}_i \\ 1 & \text{otherwise} \end{cases}$
  $\mathsf{s} \leftarrow (\mathsf{s}_1, \dots, \mathsf{s}_n) \in \{0,1\}^n$
  $(\mathsf{m}, \mathsf{r}_{\mathsf{SKE}}) \leftarrow \mathsf{RD}(\mathsf{s}, \mathsf{ct}_{\mathsf{SKE}})$
  Parse $\mathsf{m}$ as $(\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]} \in \{0,1\}^{n \cdot \lambda}$
                  and $\mathsf{k} \in \{0,1\}^\ell$.
  $\mathsf{x} \leftarrow (\mathsf{s}, (\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]}, \mathsf{k}, \mathsf{r}_{\mathsf{SKE}})$
  If $\mathsf{Eval}(\mathsf{ek}, \mathsf{x}) = \mathsf{y}$
             then return $\mathsf{x}$ else return $\perp$.
```

Figure 4: The proposed TDF TDF. $^{(\dagger)}$ $\mathsf{h} \in \{0,1\}^\lambda$ is treated as an element of $\{0,1\}^{3\lambda}$ by some canonical injective encoding (say, putting the prefix $0^{2\lambda}$), and the arithmetic is done over $\mathrm{GF}(2^{3\lambda})$ where we identify $\{0,1\}^{3\lambda}$ with $\mathrm{GF}(2^{3\lambda})$. $^{(\ddagger)}$ The addition is done over $\mathcal{C}$. $^{(\star)}$ We call this step the *find step*.

Adaptively One-Way TDFs Based on the LPN Assumptions. By instantiating the building blocks in our construction TDF properly, we obtain the first adaptively one-way TDF based on the sub-exponential hardness of the *constant-noise* LPN problem. We note that previously, even a TDF with ordinary one-wayness was not known based on the constant-noise LPN assumption. Specifically, the following LPN-based building blocks can be used.

- For KEM, we use the KEM-analogue of the IND-CPA secure PKE scheme based on the sub-exponential hardness of the constant-noise LPN problem proposed by Yu and Zhang [YZ16]. Their security analysis in fact shows that it satisfies the pseudorandom ciphertext property. However, the scheme does not satisfy almost-all-keys correctness as it is. Thus, we apply the transformation by Dwork, Naor, and Reingold [DNR04] that transforms any PKE scheme whose correctness is imperfect into one with almost-all-keys correctness. (This transformation preserves the pseudorandom ciphertext property of the underlying scheme.)

- For SKE, we can use the $\mathcal{P}$-KDM secure SKE scheme proposed by Applebaum et al. [ACPS09] based on the (polynomial) hardness of the constant-noise LPN problem. Their scheme clearly admits the randomness-recovering decryption property. In particular, whenever a plaintext is recovered in the decryption, the decryptor can also compute the "noise" used in the encryption process, which is the only encryption randomness of this scheme. In addition, their scheme can be easily made perfectly correct.

Moreover, we can also obtain the first adaptively one-way TDF based on the (polynomial) hardness of the *low-noise* LPN problem, by replacing the Yu-Zhang scheme in the above instantiation with the existing PKE schemes based on the low-noise LPN assumption [Ale03, DMN12, KMP14] (which all satisfy the pseudorandom ciphertext property). Previously, a TDF satisfying ordinary one-wayness based on the low-noise LPN assumption was proposed by Kiltz, Masny, and Pietrzak [KMP14].

Flexible Hard-core Bits k. We note that $\mathsf{k} \in \{0,1\}^\ell$ can be directly used as hard-core bits of an input $\mathsf{x} = (\mathsf{s} = (\mathsf{s}_1, \dots, \mathsf{s}_n), (\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]}, \mathsf{k}, \mathsf{r}_{\mathsf{SKE}})$, even in the presence of the inversion oracle. Its proof is a straightforward extension of the proof of Theorem 8, and thus omitted. Since an adaptively one-way TDF with ℓ-bit hard-core bits can be seen as an IND-CCA secure KEM with session-key space $\{0,1\}^\ell$, TDF can be viewed as an IND-CCA secure KEM in which the randomness used to generate a ciphertext is fully recovered during the decapsulation.

Additional Remarks. We remark that due to the structural similarity of our construction TDF to $\mathsf{KEM}_{\mathsf{cca}}$, several properties satisfied by $\mathsf{KEM}_{\mathsf{cca}}$ are inherited to TDF. Specifically, as in the case of our IND-CCA secure KEM $\mathsf{KEM}_{\mathsf{cca}}$, if we adopt the syntax that allows a system-wide public parameter shared by all users, pk^1, $(\mathsf{A}_i)_{i \in [n]}$, $(\mathsf{C}_i)_{i \in [n]}$, B, and hk in ek can be put in it, so that an evaluation key/trapdoor pair of each user consists only of $(\mathsf{pk}^0, \mathsf{sk}^0)$ of the underlying KEM KEM. Moreover, we can consider another variant of TDF in which the underlying $\mathcal{P}$-KDM secure SKE scheme is replaced with a hinting PRG, in a similar manner it is used in the Koppula-Waters construction [KW18].

Unlike $\mathsf{KEM}_{\mathsf{cca}}$, however, we can*not* make TDF perfectly correct even if we replace the underlying KEM KEM with the combination of a PKE scheme and a PRG. This is because the standard correctness of PKE does not guarantee anything about the decryption result of an element chosen randomly from the ciphertext space, which naturally occurs in the inversion process of TDF.

Simpler Construction with (Ordinary) One-wayness. If we only need to achieve (ordinary) one-wayness, then we can make some simplification to TDF. We give the simplified construction in Appendix C.

6.2 Proof of Correctness (Proof of Theorem 7)

Let $\mathsf{ek} = (\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i, \mathsf{C}_i)_{i \in [n]}, \mathsf{B}, \mathsf{hk})$ be an evaluation key. Using pk^0, pk^1, B, and $(\mathsf{C}_i)_{i \in [n]}$ in ek, for each $i \in [n]$, we define the function $f_i : \{0,1\}^{2\lambda} \to \{0,1\}^{3\lambda}$ by

$$f_i(\mathsf{r}, \mathsf{h}) : \left[\begin{array}{l} (\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Encap}(\mathsf{pk}^1; \mathsf{r});\ \ \mathsf{k}' \leftarrow \mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct} + \mathsf{C}_i); \\ \text{If } \mathsf{k}' = \bot \text{ then return } \bot \text{ else return } \mathsf{k}' - \mathsf{k} - \mathsf{B} \cdot \mathsf{h} \end{array} \right].$$

We say that an evaluation key ek is *bad* if (1) pk^0 is erroneous, or (2) some of $(\mathsf{A}_i)_{i \in [n]}$ belongs to the image of f_i. Note that the image size of f_i (excluding $\bot$) is at most $2^{2\lambda}$ for every $i \in [n]$. Since each A_i is chosen uniformly at random from $\{0,1\}^{3\lambda}$, when $\mathsf{Setup}(1^\lambda)$ is executed, the probability that bad ek is output is at most $\epsilon + n \cdot \frac{2^{2\lambda}}{2^{3\lambda}} = \epsilon + n \cdot 2^{-\lambda}$.

Now, consider the case that $(\mathsf{ek}, \mathsf{td})$ is output by Setup and ek is not bad. Let $\mathsf{x} = (\mathsf{s} = (\mathsf{s}_1, \dots, \mathsf{s}_n), (\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]}, \mathsf{k}, \mathsf{r}_{\mathsf{SKE}}) \in \{0,1\}^n \times (\{0,1\}^\lambda)^n \times \{0,1\}^\ell \times \mathcal{R}_{\mathsf{SKE}}$ be a domain element of

TDF, and let $\mathsf{y} = ((\mathsf{ct}_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}}) = \mathsf{Eval}(\mathsf{ek}, \mathsf{x})$. Then, for each $i \in [n]$, let $\mathsf{s}'_i := 1 - (\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) \overset{?}{=} \mathsf{T}_i)$.

Note that if $\mathsf{s}'_i = \mathsf{s}_i$ holds for all $i \in [n]$, then the result of the randomness-recovering decryption of $\mathsf{ct}_{\mathsf{SKE}}$ using $\mathsf{s}' = (\mathsf{s}'_1, \dots, \mathsf{s}'_n)$ as a secret key is exactly $(\mathsf{m} = ((\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]}, \mathsf{k}), \mathsf{r}_{\mathsf{SKE}})$ due to the property of RD. Thus, the validity check done in the last step of Inv never fails, and $\mathsf{Inv}(\mathsf{td}, \mathsf{y})$ will output x.

Hence, it remains to show that $\mathsf{s}'_i = \mathsf{s}_i$ holds for all $i \in [n]$.

- For the positions i with $\mathsf{s}_i = 0$, we have $\mathsf{Encap}(\mathsf{pk}^0; \mathsf{r}_i^0) = (\mathsf{ct}_i^0, \mathsf{k}_i^0) = (\mathsf{ct}_i, \mathsf{T}_i)$. Thus, the property that pk^0 is not erroneous implies $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) = \mathsf{T}_i$, and we have $\mathsf{s}'_i = 0$.
- For the positions i with $\mathsf{s}_i = 1$, we have $\mathsf{Encap}(\mathsf{pk}^1; \mathsf{r}_i^1) = (\mathsf{ct}_i^1, \mathsf{k}_i^1) = (\mathsf{ct}_i - \mathsf{C}_i, \mathsf{T}_i - \mathsf{A}_i - \mathsf{B}\cdot\mathsf{h})$, where $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}})$. If $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) = \bot$, then we clearly have $\mathsf{s}'_i = 1$. Otherwise (i.e. $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) \neq \bot$), since A_i is not in the image of f_i, we have
$$\begin{aligned} & \mathsf{A}_i \neq f_i(\mathsf{r}_i^1, \mathsf{h}) \\ & \quad = \mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i^1 + \mathsf{C}_i) - \mathsf{k}_i^1 - \mathsf{B}\cdot\mathsf{h} \\ & \quad = \mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) - (\mathsf{T}_i - \mathsf{A}_i - \mathsf{B}\cdot\mathsf{h}) - \mathsf{B}\cdot\mathsf{h} \\ \iff \quad & \mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) \neq \mathsf{T}_i. \end{aligned}$$
This again implies $\mathsf{s}'_i = 1$. Hence, regardless of $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) = \bot$ or not, we have $\mathsf{s}'_i = 1$.

The above shows that $\mathsf{s}'_i = \mathsf{s}_i$ holds for all $i \in [n]$.

Putting everything together, except for a probability at most $\epsilon + n \cdot 2^{-\lambda}$ over $(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{Inv}(\mathsf{td}, \mathsf{Eval}(\mathsf{ek}, \mathsf{x})) = \mathsf{x}$ holds for all $\mathsf{x} \in \mathcal{X}$. □ (**Theorem 7**)

6.3 Proof of Adaptive One-wayness (Proof of Theorem 8)

For simplicity, we first show the formal proof for the case $\ell = \omega(\log \lambda)$, and later explain how the proof can be carried out for the case $\ell = O(\log \lambda)$ as well. (The only difference is in the very last step of the proof.)

Let $\epsilon : \mathbb{N} \to [0, 1]$ be such that KEM is ϵ-almost-all-keys correct. Let $\mathcal{A}$ be any PPT adversary that attacks the adaptive one-wayness of TDF. We will show that for this $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_{\mathsf{tcr}}$, $\{\mathcal{B}^j_{\mathsf{prct}}\}_{j\in[2]}$, and $\mathcal{B}_{\mathsf{kdm}}$ (which makes a single KDM-encryption query) satisfying
$$\mathsf{Adv}^{\mathsf{aow}}_{\mathsf{TDF},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{Hash},\mathcal{B}_{\mathsf{tcr}}}(\lambda) + \sum_{j\in[2]} \mathsf{Adv}^{\mathsf{mprct}}_{\mathsf{KEM},n,\mathcal{B}^j_{\mathsf{prct}}}(\lambda) + \mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{P},\mathcal{B}_{\mathsf{kdm}}}(\lambda) + 3\epsilon + 3n \cdot 2^{-\lambda} + 2^{-\ell}, \quad (14)$$
which will prove the theorem for the case $\ell = \omega(\log \lambda)$.

Our proof is via a sequence of games argument using the following five games.

Game 1: This is the adaptive one-wayness experiment $\mathsf{Expt}^{\mathsf{aow}}_{\mathsf{TDF},\mathcal{A}}(\lambda)$. However, for making it easier to describe the subsequent games, we change the ordering of the operations for how the evaluation key/trapdoor pair $(\mathsf{ek}, \mathsf{td})$ and the challenge instance $\mathsf{y}^* = ((\mathsf{ct}_i^*, \mathsf{T}_i^*)_{i\in[n]}, \mathsf{ct}^*_{\mathsf{SKE}}) = \mathsf{Eval}(\mathsf{ek}, \mathsf{x}^*)$ are generated, so that the distribution of $(\mathsf{ek}, \mathsf{td}, \mathsf{y}^*)$ is identical to that in the original adaptive one-wayness experiment.

Specifically, the description of the game is as follows:

- Generate $\mathsf{ek} = (\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i, \mathsf{C}_i)_{i\in[n]}, \mathsf{B}, \mathsf{hk})$, $\mathsf{td} = (\mathsf{sk}^0, \mathsf{ek})$, and $\mathsf{y}^* = ((\mathsf{ct}_i^*, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}^*_{\mathsf{SKE}})$ as follows:

1. Compute $(\mathsf{pk}^v, \mathsf{sk}^v) \leftarrow \mathsf{KKG}(1^\lambda)$ for both $v \in \{0,1\}$, and pick $\mathsf{B} \xleftarrow{\mathsf{r}} \{0,1\}^{3\lambda}$.
2. Compute $\mathsf{s}_i^* = (\mathsf{s}_1^*, \dots, \mathsf{s}_n^*) \leftarrow \mathsf{K}(1^\lambda)$, pick $\mathsf{r}_1^{*(\mathsf{s}_1^*)}, \dots, \mathsf{r}_n^{*(\mathsf{s}_n^*)} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, $\mathsf{k}^* \xleftarrow{\mathsf{r}} \{0,1\}^\ell$, and $\mathsf{r}_{\mathsf{SKE}}^* \xleftarrow{\mathsf{r}} \mathcal{R}_{\mathsf{SKE}}$, and set $\mathsf{x}^* \leftarrow (\mathsf{s}^*, (\mathsf{r}_i^{*(\mathsf{s}_i^*)})_{i\in[n]}, \mathsf{k}^*, \mathsf{r}_{\mathsf{SKE}}^*)$.
3. Compute $\mathsf{ct}_{\mathsf{SKE}}^* \leftarrow \mathsf{E}(\mathsf{s}^*, (\mathsf{r}_i^{*(\mathsf{s}_i^*)})_{i\in[n]} \| \mathsf{k}^*; \mathsf{r}_{\mathsf{SKE}}^*)$.
4. Compute $(\mathsf{ct}_i^{*(\mathsf{s}_i^*)}, \mathsf{k}_i^{*(\mathsf{s}_i^*)}) \leftarrow \mathsf{Encap}(\mathsf{pk}^{\mathsf{s}_i^*}; \mathsf{r}_i^{*(\mathsf{s}_i^*)})$ for every $i \in [n]$.
5. Pick $\mathsf{C}_1, \dots, \mathsf{C}_n \xleftarrow{\mathsf{r}} \mathcal{C}$.
6. Compute $\mathsf{ct}_i^* \leftarrow \mathsf{ct}_i^{*(\mathsf{s}_i^*)} + \mathsf{s}_i^* \cdot \mathsf{C}_i$ for every $i \in [n]$.
7. Compute $\mathsf{hk} \leftarrow \mathsf{HKG}(1^\lambda)$ and $\mathsf{h}^* \leftarrow \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i^*)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}^*)$.
8. Pick $\mathsf{A}_1, \dots, \mathsf{A}_n \xleftarrow{\mathsf{r}} \{0,1\}^{3\lambda}$.
9. Compute $\mathsf{T}_i^* \leftarrow \mathsf{k}_i^{*(\mathsf{s}_i^*)} + \mathsf{s}_i^* \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}^*)$ for every $i \in [n]$.
10. Set $\mathsf{ek} \leftarrow (\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i, \mathsf{C}_i)_{i\in[n]}, \mathsf{B}, \mathsf{hk})$, $\mathsf{td} \leftarrow (\mathsf{sk}^0, \mathsf{sk}^1, \mathsf{ek})$, and $\mathsf{y}^* \leftarrow ((\mathsf{ct}_i^*, \mathsf{T}_i^*)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}}^*)$.

- Run $\mathcal{A}(\mathsf{ek}, \mathsf{y}^*)$. From here on, $\mathcal{A}$ may start making inversion queries.
- Inversion queries $\mathsf{y} = ((\mathsf{ct}_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ are answered as follows: First, compute $\mathsf{h} \leftarrow \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}})$. Next, compute $\mathsf{s}_i \leftarrow 1 - (\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) \overset{?}{=} \mathsf{T}_i)$ for every $i \in [n]$, and set $\mathsf{s} \leftarrow (\mathsf{s}_1, \dots, \mathsf{s}_n)$. Then, compute $(\mathsf{m}, \mathsf{r}_{\mathsf{SKE}}) \leftarrow \mathsf{RD}(\mathsf{s}, \mathsf{ct}_{\mathsf{SKE}})$ and parse m as $(\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]} \in (\{0,1\}^\lambda)^n$ and $\mathsf{k} \in \{0,1\}^\ell$. Set $\mathsf{x} \leftarrow (\mathsf{s}, (\mathsf{r}_i^{\mathsf{s}_i})_{i\in[n]}, \mathsf{k}, \mathsf{r}_{\mathsf{SKE}})$. Finally, if $\mathsf{Eval}(\mathsf{ek}, \mathsf{x}) = \mathsf{y}$ holds, then return x to $\mathcal{A}$. Otherwise, return $\perp$ to $\mathcal{A}$.
- At some point, $\mathcal{A}$ terminates with output a candidate inversion result $\mathsf{x}' \in \mathcal{X}$.

For convenience, in the following we will use the following sets:

$$\mathcal{S}_{\mathsf{zero}} := \left\{ j \in [n] \;\middle|\; \mathsf{s}_j^* = 0 \right\} \quad \text{and} \quad \mathcal{S}_{\mathsf{one}} := \left\{ j \in [n] \;\middle|\; \mathsf{s}_j^* = 1 \right\} = [n] \setminus \mathcal{S}_{\mathsf{zero}}.$$

Game 2: Same as Game 1, except for an additional rejection rule in the inversion oracle. Specifically, in this game, if $\mathcal{A}$'s inversion query $\mathsf{CT} = ((\mathsf{ct}_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ satisfies $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}) = \mathsf{h}^*$, then the inversion oracle immediately returns $\perp$ to $\mathcal{A}$.

Game 3: Same as Game 2, except for how $(\mathsf{C}_i)_{i\in[n]}$ and $(\mathsf{A}_i)_{i\in[n]}$ are generated.

Specifically, in this game, we additionally pick $\mathsf{r}_1^{*(1-\mathsf{s}_1^*)}, \dots, \mathsf{r}_n^{*(1-\mathsf{s}_n^*)} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, and compute $(\mathsf{ct}_i^{*(1-\mathsf{s}_i^*)}, \mathsf{k}_i^{*(1-\mathsf{s}_i^*)}) \leftarrow \mathsf{Encap}(\mathsf{pk}^{1-\mathsf{s}_i^*}; \mathsf{r}_i^{*(1-\mathsf{s}_i^*)})$ for every $i \in [n]$. Then, for the positions $i \in \mathcal{S}_{\mathsf{zero}}$, C_i's and A_i's are generated by

$$\mathsf{C}_i \leftarrow \mathsf{ct}_i^{*0} - \mathsf{ct}_i^{*1} \qquad \text{and} \qquad \mathsf{A}_i \leftarrow \mathsf{k}_i^{*0} - \mathsf{k}_i^{*1} - \mathsf{B} \cdot \mathsf{h}^*. \tag{15}$$

(At this point, C_i's and A_i's for the remaining positions $i \in \mathcal{S}_{\mathsf{one}}$ are unchanged.)

Game 4: Same as Game 3, except for the behavior of the inversion oracle. Specifically, in this game, for answering $\mathcal{A}$'s inversion queries $\mathsf{y} = ((\mathsf{ct}_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$, the oracle first computes $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}})$, and returns $\perp$ to $\mathcal{A}$ if $\mathsf{h} = \mathsf{h}^*$. (This rejection rule is the same as in Game 3.) Otherwise, the oracle uses the "alternative inversion algorithm" AltInv and the "alternative trapdoor" td' defined below for computing the inversion result x returned to $\mathcal{A}$.

AltInv takes $\mathsf{td}' := (\mathsf{sk}^1, \mathsf{ek})$ and y as input, and proceeds identically to $\mathsf{Inv}(\mathsf{td}, \mathsf{y})$, except that the "find step" (i.e. the step for computing s_i's) is replaced with the following procedure:

$$\forall i \in [n]: \quad \mathsf{s}_i \leftarrow \Big(\mathsf{Decap}(\mathsf{sk}^1, \mathsf{ct}_i - \mathsf{C}_i) \overset{?}{=} \mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h} \Big)$$
$$= \begin{cases} 1 & \text{if } \mathsf{Decap}(\mathsf{sk}^1, \mathsf{ct}_i - \mathsf{C}_i) = \mathsf{T}_i - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h} \\ 0 & \text{otherwise} \end{cases}.$$

Note that due to this change, the inversion oracle answers $\mathcal{A}$'s queries without using sk^0.

Game 5: Same as Game 4, except for how C_i's and A_i's for the positions $i \in \mathcal{S}_{\mathsf{one}}$ are generated. Specifically, in this game, C_i's and A_i's for $i \in \mathcal{S}_{\mathsf{one}}$ are also generated as in Equation 15.

Note that due to this change, all of $(\mathsf{C}_i)_{i\in[n]}$ and $(\mathsf{A}_i)_{i\in[n]}$ are generated as in Equation 15. Furthermore, $\mathsf{ct}_i^* = \mathsf{ct}_i^{*0}$ and $\mathsf{T}_i^* = \mathsf{k}_i^{*0}$ hold for every $i \in [n]$, no matter whether $\mathsf{s}_i^* = 0$ or $\mathsf{s}_i^* = 1$. Indeed, this is the case for the positions $i \in \mathcal{S}_{\mathsf{zero}}$ by design. For the positions $i \in \mathcal{S}_{\mathsf{one}}$, we have

$$\mathsf{ct}_i^* = \mathsf{ct}_i^{*1} + \mathsf{C}_i = \mathsf{ct}_i^{*1} + (\mathsf{ct}_i^{*0} - \mathsf{ct}_i^{*1}) = \mathsf{ct}_i^{*0},$$
$$\mathsf{T}_i^* = \mathsf{k}_i^{*1} + \mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}^* = \mathsf{k}_i^{*1} + (\mathsf{k}_i^{*0} - \mathsf{k}_i^{*1} - \mathsf{B} \cdot \mathsf{h}^*) + \mathsf{B} \cdot \mathsf{h}^* = \mathsf{k}_i^{*0}.$$

Hence, in this game, values dependent on s^* appear only in the plaintext of $\mathsf{ct}_{\mathsf{SKE}}^*$ (i.e. $(\mathsf{r}_i^{*(\mathsf{s}_i^*)})_{i\in[n]} \| \mathsf{k}^*$).

For $j \in [5]$, let $\mathtt{SUC}_j$ be the event that $\mathcal{A}$ succeeds in inverting the challenge instance (i.e. $\mathsf{x}' = \mathsf{x}^*$ occurs) in Game j. Then, by the definition of the events and the triangle inequality, we have

$$\mathsf{Adv}^{\mathsf{aow}}_{\mathsf{TDF},\mathcal{A}}(\lambda) = \Pr[\mathtt{SUC}_1] \le \sum_{j\in[4]} |\Pr[\mathtt{SUC}_j] - \Pr[\mathtt{SUC}_{j+1}]| + \Pr[\mathtt{SUC}_5]. \tag{16}$$

In the following, we show how the terms appearing in Equation 16 are bounded.

Lemma 7 *There exists a PPT adversary* $\mathcal{B}_{\mathsf{tcr}}$ *satisfying* $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \le \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{Hash},\mathcal{B}_{\mathsf{tcr}}}(\lambda) + \epsilon + n \cdot 2^{-\lambda}$.

Proof of Lemma 7. Unlike the proof of Lemma 1 in the proof of Theorem 2, the proof of this lemma need not use a deferred analysis.

We say that an inversion query $\mathsf{y} = ((\mathsf{ct}_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ made by $\mathcal{A}$ in Game $j \in [2]$ is *hash-bad* if $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}) = \mathsf{h}^*$ and $\mathsf{Inv}(\mathsf{td}, \mathsf{y}) \ne \bot$ hold simultaneously. We categorize a hash-bad inversion query into the following two mutually exclusive types:

- Type-1: $(\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}} \ne (\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}^*$
- Type-2: $(\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}} = (\mathsf{ct}_i^*)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}^*$

For $j, k \in [2]$, let $\mathtt{HB}_j^k$ be the event that $\mathcal{A}$ makes at least one Type-k hash-bad inversion query in Game j, and let $\mathtt{PKB}_j$ be the event that pk^0 is erroneous in Game j. Observe that if pk^0 is not erroneous and $\mathcal{A}$ does not make a hash-bad query, then Game 1 and Game 2 proceed identically. Thus, we have

$$|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \le \Pr[\mathtt{PKB}_1 \vee \mathtt{HB}_1^1 \vee \mathtt{HB}_1^2] \le \Pr[\mathtt{PKB}_1] + \Pr[\mathtt{HB}_1^1] + \Pr[\mathtt{HB}_1^2 | \overline{\mathtt{PKB}_1}].$$

Here, by definition, we have $\Pr[\mathtt{PKB}_1] = \epsilon$. Furthermore, note that a Type-1 hash-bad query can be directly used to break the target collision resistance of Hash. That is, we can construct a PPT adversary $\mathcal{B}_{\mathsf{tcr}}$ with $\Pr[\mathtt{HB}_1^1] = \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{Hash},\mathcal{B}_{\mathsf{tcr}}}(\lambda)$. Since the construction of $\mathcal{B}_{\mathsf{tcr}}$ is straightforward, we omit the detail.

It remains to show how $\Pr[\mathtt{HB}_1^2|\overline{\mathtt{PKB}_1}]$ is bounded. In the following, we show that if pk^0 is not erroneous, then except for a probability at most $n \cdot 2^{-\lambda}$ over the choice of $\mathsf{C}_1, \dots, \mathsf{C}_n \xleftarrow{\mathsf{r}} \mathcal{C}$, a Type-2 hash-bad query does not exist in Game 1.

Recall that $\mathcal{A}$'s inversion query y must satisfy $\mathsf{y} \neq \mathsf{y}^*$, and thus if y is a Type-2 hash-bad query, then there must exist a position $j \in [n]$ for which $\mathsf{T}_j \neq \mathsf{T}_j^*$ holds. For a "candidate" Type-2 hash-bad query of the form $\mathsf{y} = ((\mathsf{ct}_i^*, \mathsf{T}_i)_{i \in [n]}, \mathsf{ct}^*_{\mathsf{SKE}})$, define $\mathtt{Diff}_{\mathsf{y}} := \{j \in [n] | \mathsf{T}_j \neq \mathsf{T}_j^*\}$, and for each $i \in [n]$, let $\mathsf{s}_i := 1 - (\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i^*) \overset{?}{=} \mathsf{T}_i)$.

We observe that for the positions $i \in \mathtt{Diff}_{\mathsf{y}}$, the following holds.

- If $\mathsf{s}_i^* = 0$, then we have $\mathsf{Encap}(\mathsf{pk}^0; \mathsf{r}_i^{*0}) = (\mathsf{ct}_i^{*0}, \mathsf{k}_i^{*0}) = (\mathsf{ct}_i^*, \mathsf{T}_i^*)$. This, $\mathsf{T}_i \neq \mathsf{T}_i^*$, and pk^0 being non-erroneous imply $\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) \neq \mathsf{T}_i$, which in turn implies $\mathsf{s}_i = 1$. Then, in order for y of the above form to satisfy $\mathsf{Inv}(\mathsf{td}, \mathsf{y}) \neq \perp$, $\mathsf{ct}_i^* - \mathsf{C}_i$ must be in the image of $\mathsf{Encap}(\mathsf{pk}^1; \cdot)$ due to the validity check by re-evaluation done in the last step of Inv. However, since the image size of $\mathsf{Encap}(\mathsf{pk}^1; \cdot)$ is at most 2^λ, when choosing $\mathsf{C}_i \xleftarrow{\mathsf{r}} \mathcal{C}$, the probability that $\mathsf{ct}_i^* - \mathsf{C}_i$ belongs to the image of $\mathsf{Encap}(\mathsf{pk}^1; \cdot)$ is at most $\frac{2^\lambda}{|\mathcal{C}|} \leq 2^{-\lambda}$, where we use $|\mathcal{C}| \geq 2^{2\lambda}$. Hence, in case $\mathsf{s}_i^* = 0$, except for a probability at most $2^{-\lambda}$ over the choice of $\mathsf{C}_i \xleftarrow{\mathsf{r}} \mathcal{C}$, a Type-2 hash-bad inversion query y with $\mathsf{T}_i \neq \mathsf{T}_i^*$ does not exist.
- On the other hand, if $\mathsf{s}_i^* = 1$, then the argument here is symmetric to the case of $\mathsf{s}_i^* = 0$. Specifically, we have $\mathsf{Encap}(\mathsf{pk}^1; \mathsf{r}_i^{*1}) = (\mathsf{ct}_i^{*1}, \mathsf{k}_i^{*1}) = (\mathsf{ct}_i^* - \mathsf{C}_i, \mathsf{T}_i^* - \mathsf{A}_i - \mathsf{B} \cdot \mathsf{h}^*)$. This and $\mathsf{T}_i \neq \mathsf{T}_i^*$ imply that if $\mathsf{s}_i = 1$, then $\mathsf{Inv}(\mathsf{td}, \mathsf{y}) \neq \perp$ cannot hold due to the validity check by re-evaluation done in the last step of Inv. In case $\mathsf{s}_i = 0$, in order for a candidate Type-2 hash-bad inversion query $\mathsf{y} = ((\mathsf{ct}_i^*, \mathsf{T}_i)_{i \in [n]}, \mathsf{ct}^*_{\mathsf{SKE}})$ with $\mathsf{s}_i = 0$ to satisfy $\mathsf{Inv}(\mathsf{td}, \mathsf{y}) \neq \perp$, $\mathsf{ct}_i^* = \mathsf{ct}_i^{*1} + \mathsf{C}_i$ must be in the image of $\mathsf{Encap}(\mathsf{pk}^0; \cdot)$ due to the validity check by re-evaluation done in the last step of Inv. However, since the image size of $\mathsf{Encap}(\mathsf{pk}^0; \cdot)$ is at most 2^λ, when choosing $\mathsf{C}_i \xleftarrow{\mathsf{r}} \mathcal{C}$, the probability that $\mathsf{ct}_i^{*1} + \mathsf{C}_i$ belongs to the image of $\mathsf{Encap}(\mathsf{pk}^0; \cdot)$ is at most $\frac{2^\lambda}{|\mathcal{C}|} \leq 2^{-\lambda}$, where we use $|\mathcal{C}| \geq 2^{2\lambda}$. Hence, in case $\mathsf{s}_i^* = 1$, except for a probability at most $2^{-\lambda}$ over the choice of $\mathsf{C}_i \xleftarrow{\mathsf{r}} \mathcal{C}$, a Type-2 hash-bad inversion query y with $\mathsf{T}_i \neq \mathsf{T}_i^*$ does not exist.

By the union bound, conditioned on pk^0 being non-erroneous, except for a probability at most $n \cdot 2^{-\lambda}$ over the choice of $\mathsf{C}_1, \dots, \mathsf{C}_n \xleftarrow{\mathsf{r}} \mathcal{C}$, a Type-2 hash-bad inversion query does not exist in Game 1. Hence, we have $\Pr[\mathtt{HB}_1^2|\overline{\mathtt{PKB}_1}] \leq n \cdot 2^{-\lambda}$.

Putting everything together, there exists a PPT adversary $\mathcal{B}_{\mathsf{tcr}}$ with the claimed advantage, as required. □ (**Lemma 7**)

Lemma 8 *There exists a PPT adversary* $\mathcal{B}^1_{\mathsf{prct}}$ *satisfying* $|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]| = \mathsf{Adv}^{\mathsf{mprct}}_{\mathsf{KEM},n,\mathcal{B}^1_{\mathsf{prct}}}(\lambda)$.

Proof of Lemma 8. Using $\mathcal{A}$ as a building block, we show how to construct a PPT adversary $\mathcal{B}^1_{\mathsf{prct}}$ that attacks the n-challenge pseudorandom ciphertext property of KEM with the claimed advantage. The description of $\mathcal{B}^1_{\mathsf{prct}}$ is as follows.

$\mathcal{B}^1_{\mathsf{prct}}(\mathsf{pk}', (\mathsf{ct}'^*_{i,\beta}, \mathsf{k}'^*_{i,\beta})_{i \in [n]})$**:** (where $\beta \in \{0,1\}$ denotes $\mathcal{B}^1_{\mathsf{prct}}$'s challenge bit) First, $\mathcal{B}^1_{\mathsf{prct}}$ runs $\mathsf{s}^* = (\mathsf{s}_1^*, \dots, \mathsf{s}_n^*) \xleftarrow{\mathsf{r}} \mathsf{K}(1^\lambda)$, and sets $\mathsf{pk}^1 \leftarrow \mathsf{pk}'$ and $(\mathsf{ct}_i^{*1}, \mathsf{k}_i^{*1}) \leftarrow (\mathsf{ct}'^*_{i,\beta}, \mathsf{k}'^*_{i,\beta})$ for the positions $i \in \mathcal{S}_{\mathsf{zero}}$. Then, $\mathcal{B}^1_{\mathsf{prct}}$ generates the remaining values in $\mathsf{ek} = (\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i, \mathsf{C}_i)_{i \in [n]}, \mathsf{B}, \mathsf{hk})$, $\mathsf{td} = (\mathsf{sk}^0, \mathsf{ek})$, $\mathsf{x}^* = (\mathsf{s}^*, (\mathsf{r}_i^{*(\mathsf{s}_i^*)})_{i \in [n]}, \mathsf{k}^*, \mathsf{r}^*_{\mathsf{SKE}})$, and $\mathsf{y}^* = ((\mathsf{ct}_i^*, \mathsf{T}_i^*)_{i \in [n]}, \mathsf{ct}^*_{\mathsf{SKE}})$ in exactly the same way as in Game 3, and runs $\mathsf{x}' \leftarrow \mathcal{A}^{\mathsf{Inv}(\mathsf{td},\cdot)}(\mathsf{ek}, \mathsf{y}^*)$. (Note that sk^0 is generated by $\mathcal{B}^1_{\mathsf{prct}}$ itself.) Finally, $\mathcal{B}^1_{\mathsf{prct}}$ terminates with output $\beta' \leftarrow (\mathsf{x}' \overset{?}{=} \mathsf{x}^*)$.

It is not hard to see that if $\mathcal{B}^1_{\mathsf{prct}}$'s challenge bit β is 0 (resp. 1), $\mathcal{B}^1_{\mathsf{prct}}$ perfectly simulates Game 2 (resp. Game 3) for $\mathcal{A}$. In particular, if $\beta = 0$, then $(\mathsf{ct}^{*1}_i, \mathsf{k}^{*1}_i) = (\mathsf{ct}'^*_{i,0}, \mathsf{k}'^*_{i,0})$ for every $i \in \mathcal{S}_{\mathsf{zero}}$ is chosen uniformly at random from $\mathcal{C} \times \{0,1\}^{3\lambda}$, and thus regardless of what values ct^{*0}_i, k^{*0}_i, B, and h^* take, $\mathsf{C}_i = \mathsf{ct}^{*0}_i - \mathsf{ct}^{*1}_i$ and $\mathsf{A}_i = \mathsf{k}^{*0}_i - \mathsf{k}^{*1}_i - \mathsf{B} \cdot \mathsf{h}^*$ are distributed uniformly over $\mathcal{C}$ and $\{0,1\}^{3\lambda}$, respectively. Under this situation, the probability that $\mathsf{x}' = \mathsf{x}^*$ holds (and thus $\mathcal{B}^1_{\mathsf{prct}}$ outputs $\beta' = 1$) is exactly $\Pr[\mathtt{SUC}_2]$, i.e. we have $\Pr[\beta' = 1 | \beta = 0] = \Pr[\mathtt{SUC}_2]$. Similarly, we have $\Pr[\beta' = 1 | \beta = 1] = \Pr[\mathtt{SUC}_3]$. Hence, we have

$$\mathsf{Adv}^{\mathsf{mprct}}_{\mathsf{KEM},n,\mathcal{B}^1_{\mathsf{prct}}}(\lambda) = \left|\Pr[\beta' = 1 | \beta = 0] - \Pr[\beta' = 1 | \beta = 1]\right| = \left|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]\right|,$$

as desired. $\square$ (**Lemma 8**)

Lemma 9 $|\Pr[\mathtt{SUC}_3] - \Pr[\mathtt{SUC}_4]| \leq 2\epsilon + n \cdot 2^{-\lambda+1}$ *holds.*

Proof of Lemma 9. Note that Game 3 and Game 4 proceed identically unless $\mathcal{A}$ makes an inversion query $\mathsf{y} = ((\mathsf{ct}_i, \mathsf{T}_i)_{i \in [n]}, \mathsf{ct}_{\mathsf{SKE}})$ such that $\mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}) \neq \mathsf{h}^*$ and $\mathsf{Inv}(\mathsf{td}, \mathsf{y}) \neq \mathsf{AltInv}(\mathsf{td}', \mathsf{y})$ hold simultaneously. We call such an inversion query *bad*. In the following, we will show that if ek is not "bad" in the sense specified below, a bad inversion query does not exist in Game 3 and Game 4, and the probability that ek becomes bad is bounded by $2\epsilon + n \cdot 2^{-\lambda+1}$. This will prove the lemma.

Fix the following values in Game 3:

- $(\mathsf{pk}^0, \mathsf{sk}^0), (\mathsf{pk}^1, \mathsf{sk}^1) \in \mathsf{Sup}(\mathsf{KKG}(1^\lambda))$ such that pk^0 and pk^1 are not erroneous, and $\mathsf{hk} \in \mathsf{Sup}(\mathsf{HKG}(1^\lambda))$.
- $\mathsf{s}^* = (\mathsf{s}^*_1, \dots, \mathsf{s}^*_n) \in \mathsf{Sup}(\mathsf{K}(1^\lambda))$, $\mathsf{r}^{*0}_1, \dots, \mathsf{r}^{*0}_n, \mathsf{r}^{*1}_1, \dots, \mathsf{r}^{*1}_n \in \{0,1\}^\lambda$, $\mathsf{k}^* \in \{0,1\}^\ell$, and $\mathsf{r}^*_{\mathsf{SKE}} \in \mathcal{R}_{\mathsf{SKE}}$.
- $\mathsf{x}^* = (\mathsf{s}^*, (\mathsf{r}^{*(\mathsf{s}^*_i)}_i)_{i\in[n]}, \mathsf{r}^*_{\mathsf{SKE}})$.
- $(\mathsf{ct}^{*v}_i, \mathsf{k}^{*v}_i) = \mathsf{Encap}(\mathsf{pk}^v; \mathsf{r}^{*v}_i)$ for all $(i, v) \in [n] \times \{0,1\}$.
- $\mathsf{C}_i = \mathsf{ct}^{*0}_i - \mathsf{ct}^{*1}_i$ for all $i \in \mathcal{S}_{\mathsf{zero}}$ and $\mathsf{C}_i \in \mathcal{C}$ for all $i \in \mathcal{S}_{\mathsf{one}}$.
- $\mathsf{ct}^*_i = \mathsf{ct}^{*(\mathsf{s}^*_i)}_i + \mathsf{s}^*_i \cdot \mathsf{C}_i$ for all $i \in [n]$.
- $\mathsf{ct}^*_{\mathsf{SKE}} = \mathsf{E}(\mathsf{s}^*, (\mathsf{r}^{*(\mathsf{s}^*_i)}_i)_{i\in[n]} \| \mathsf{k}^*; \mathsf{r}^*_{\mathsf{SKE}})$ and $\mathsf{h}^* = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}^*_i)_{i\in[n]} \| \mathsf{ct}^*_{\mathsf{SKE}})$.

To define the notion of "badness" for an evaluation key, we introduce two types of functions based on the above fixed values.

- For each $i \in \mathcal{S}_{\mathsf{zero}}$ and $v \in \{0,1\}$, we define the function $\widehat{g}_{i,v} : \{0,1\}^\lambda \times (\{0,1\}^\lambda \setminus \{\mathsf{h}^*\}) \to \{0,1\}^{3\lambda} \cup \{\bot\}$ by

$$\widehat{g}_{i,v}(\mathsf{r}, \mathsf{h}) : \left[\begin{array}{l} (\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Encap}(\mathsf{pk}^v; \mathsf{r});\ \mathsf{k}' \leftarrow \mathsf{Decap}(\mathsf{sk}^{1-v}, \mathsf{ct} - (1-v)\cdot \mathsf{C}_i); \\ \text{If } \mathsf{k}' = \bot \text{ then return } \bot \text{ else return } \frac{(\mathsf{k}-\mathsf{k}')\cdot(-1)^v - \mathsf{k}^{*0}_i + \mathsf{k}^{*1}_i}{\mathsf{h}-\mathsf{h}^*} \end{array} \right]$$

  We say that a string $\mathsf{B} \in \{0,1\}^{3\lambda}$ is *bad* if B belongs to the image of $\widehat{g}_{i,v}$ for some $(i, v) \in [n] \times \{0,1\}$. Note that the image size of $\widehat{g}_{i,v}$ (excluding $\bot$) is at most $2^{2\lambda}$ for every $i \in \mathcal{S}_{\mathsf{zero}}$ and $v \in \{0,1\}$. Hence, when choosing $\mathsf{B} \xleftarrow{\mathsf{r}} \{0,1\}^{3\lambda}$, the probability that B is bad is at most $|\mathcal{S}_{\mathsf{zero}}| \cdot 2 \cdot \frac{2^{2\lambda}}{2^{3\lambda}} = |\mathcal{S}_{\mathsf{zero}}| \cdot 2^{-\lambda+1}$.

- For each $\mathsf{B}' \in \{0,1\}^{3\lambda}$, $i \in [n]$, and $v \in \{0,1\}$, we define the function $g_{\mathsf{B}',i,v} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^{3\lambda} \cup \{\bot\}$ by
$$g_{\mathsf{B}',i,v}(\mathsf{r},\mathsf{h}) : \left[\begin{array}{l} (\mathsf{ct},\mathsf{k}) \leftarrow \mathsf{Encap}(\mathsf{pk}^v;\mathsf{r});\ \mathsf{k}' \leftarrow \mathsf{Decap}(\mathsf{sk}^{1-v}, \mathsf{ct} - (1-v)\cdot \mathsf{C}_i); \\ \text{If } \mathsf{k}' = \bot \text{ then return } \bot \text{ else return } (\mathsf{k}-\mathsf{k}')\cdot(-1)^v - \mathsf{B}'\cdot\mathsf{h} \end{array} \right].$$
For each pair $(\mathsf{B}', i) \in \{0,1\}^{3\lambda} \times [n]$, we say that a string $\mathsf{A}' \in \{0,1\}^{3\lambda}$ is *bad with respect to* (B', i) if A' belongs to the image of $g_{\mathsf{B}',i,0}$ or that of $g'_{\mathsf{B}',i,1}$. The image size of $g_{\mathsf{B}',i,v}$ (excluding $\bot$) is at most $2^{2\lambda}$ for every $(\mathsf{B}', i, v) \in \{0,1\}^{3\lambda} \times [n] \times \{0,1\}$. Hence, for any fixed $\mathsf{B}' \in \{0,1\}^{3\lambda}$, when choosing $\mathsf{A}_i \xleftarrow{\mathsf{r}} \{0,1\}^{3\lambda}$ for all $i \in \mathcal{S}_{\mathsf{one}}$, the probability that some A_i is bad with respect to (B', i) is at most $|\mathcal{S}_{\mathsf{one}}| \cdot 2 \cdot \frac{2^{2\lambda}}{2^{3\lambda}} = |\mathcal{S}_{\mathsf{one}}| \cdot 2^{-\lambda+1}$.

We say that an evaluation key ek generated in Game 3 is *bad* if (1) either pk^0 or pk^1 is erroneous, or (2) either B is bad or A_i for some $i \in \mathcal{S}_{\mathsf{one}}$ is bad with respect to (B, i). By the union bound, the probability that ek is bad in Game 3 is bounded by $2\epsilon + |\mathcal{S}_{\mathsf{zero}}| \cdot 2^{-\lambda+1} + |\mathcal{S}_{\mathsf{one}}| \cdot 2^{-\lambda+1} = 2\epsilon + n \cdot 2^{-\lambda+1}$.

To complete the proof, in the following we show that if $\mathsf{ek} = (\mathsf{pk}^0, \mathsf{pk}^1, (\mathsf{A}_i, \mathsf{C}_i)_{i\in[n]}, \mathsf{B}, \mathsf{hk})$ is not bad, then for any element $\mathsf{y} = ((\mathsf{ct}_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ (possibly outside the image of $\mathsf{Eval}(\mathsf{ek}, \cdot)$) satisfying $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}) \neq \mathsf{h}^*$, we always have $\mathsf{Inv}(\mathsf{td}, \mathsf{y}) = \mathsf{AltInv}(\mathsf{td}', \mathsf{y})$.

Let $\mathsf{y} = ((\mathsf{ct}_i, \mathsf{T}_i)_{i\in[n]}, \mathsf{ct}_{\mathsf{SKE}})$ be any value satisfying $\mathsf{h} = \mathsf{H}(\mathsf{hk}, (\mathsf{ct}_i)_{i\in[n]} \| \mathsf{ct}_{\mathsf{SKE}}) \neq \mathsf{h}^*$. For each $i \in [n]$, define
$$\mathsf{s}_i := 1 - \Big(\mathsf{Decap}(\mathsf{sk}^0, \mathsf{ct}_i) \overset{?}{=} \mathsf{T}_i \Big), \qquad \text{and}$$
$$\mathsf{s}'_i := \Big(\mathsf{Decap}(\mathsf{sk}^1, \mathsf{ct}_i - \mathsf{C}_i) \overset{?}{=} \mathsf{T}_i - \mathsf{A}_i - \mathsf{B}\cdot\mathsf{h} \Big).$$
We consider two cases and show that $\mathsf{Inv}(\mathsf{td}, \mathsf{y}) = \mathsf{AltInv}(\mathsf{td}', \mathsf{y})$ holds in either case.

- **Case 1: For all positions $i \in [n]$, there exists a pair $(\mathsf{r}, v) \in \{0,1\}^\lambda \times \{0,1\}$ satisfying $\mathsf{Encap}(\mathsf{pk}^v;\mathsf{r}) = (\mathsf{ct}_i^v - v\cdot\mathsf{C}_i, \mathsf{T}_i - v\cdot(\mathsf{A}_i + \mathsf{B}\cdot\mathsf{h}))$.**

  In this case, we show that $\mathsf{s}_i = \mathsf{s}'_i$ holds for all $i \in [n]$. This in turn implies that the output of Inv and that of AltInv agree since these algorithms proceed identically after they respectively compute s.

  Fix $i \in [n]$. The condition of this case directly implies $\mathsf{Decap}(\mathsf{sk}^v, \mathsf{ct}_i - v\cdot\mathsf{C}_i) = \mathsf{T}_i - v\cdot(\mathsf{A}_i + \mathsf{B}\cdot\mathsf{h})$. This in turn implies that if $v = 0$ then we have $\mathsf{s}_i = 0$, while if $v = 1$ then we have $\mathsf{s}'_i = 1$. In the following, we show that
$$\mathsf{k}' := \mathsf{Decap}(\mathsf{sk}^{1-v}, \mathsf{ct}_i - (1-v)\cdot\mathsf{C}_i) \neq \mathsf{T}_i - (1-v)\cdot(\mathsf{A}_i + \mathsf{B}\cdot\mathsf{h}) \tag{17}$$
  holds, which implies that if $v = 0$ then we have $\mathsf{s}'_i = 0$, while if $v = 1$ then we have $\mathsf{s}_i = 1$. Hence, combined together, we will obtain the desired conclusion $\mathsf{s}_i = \mathsf{s}'_i$ (regardless of the value of v). Also, if $\mathsf{k}' = \bot$, then Equation 17 is obviously satisfied. Thus, below we consider the case $\mathsf{k}' \neq \bot$.

  The argument for showing Equation 17 differs depending on whether $i \in \mathcal{S}_{\mathsf{zero}}$ or $i \in \mathcal{S}_{\mathsf{one}}$. If $i \in \mathcal{S}_{\mathsf{zero}}$, then since B is not bad, it is not in the image of $\widehat{g}_{i,v}$. Hence, we have
$$\begin{aligned} \mathsf{B} \neq \widehat{g}_{i,v}(\mathsf{r},\mathsf{h}) &= \frac{\Big(\mathsf{T}_i - v\cdot(\mathsf{A}_i + \mathsf{B}\cdot\mathsf{h}) - \mathsf{k}'\Big)\cdot(-1)^v - \mathsf{k}_i^{*0} + \mathsf{k}_i^{*1}}{\mathsf{h} - \mathsf{h}^*} \\ \iff \quad & \mathsf{k}' \neq \mathsf{T}_i - v\cdot(\mathsf{A}_i + \mathsf{B}\cdot\mathsf{h}) - (-1)^v \cdot \Big((\mathsf{k}_i^{*0} - \mathsf{k}_i^{*1} - \mathsf{B}\cdot\mathsf{h}^*) + \mathsf{B}\cdot\mathsf{h}\Big) \\ & \overset{(*)}{=} \mathsf{T}_i - \Big(v + (-1)^v\Big)\cdot(\mathsf{A}_i + \mathsf{B}\cdot\mathsf{h}) \overset{(**)}{=} \mathsf{T}_i - (1-v)\cdot(\mathsf{A}_i + \mathsf{B}\cdot\mathsf{h}), \end{aligned}$$

where the equality (*) uses $\mathsf{A}_i = \mathsf{k}_i^{*0} - \mathsf{k}_i^{*1} - \mathsf{B} \cdot \mathsf{h}^*$, which is how A_i is generated for the positions $i \in \mathcal{S}_{\mathsf{zero}}$ in Game 3; The equality (**) is due to $v + (-1)^v = 1 - v$ for $v \in \{0,1\}$.

Similarly, if $i \in \mathcal{S}_{\mathsf{one}}$, then since A_i is not bad with respect to (B, i), it is not in the image of $g_{\mathsf{B},i,v}$. Hence, we have

$$\begin{aligned} & \mathsf{A}_i \neq g_{\mathsf{B},i,v}(\mathsf{r},\mathsf{h}) = \Big(\mathsf{T}_i - v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) - \mathsf{k}'\Big) \cdot (-1)^v - \mathsf{B} \cdot \mathsf{h} \\ \iff \quad & \mathsf{k}' \neq \mathsf{T}_i - v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) - (-1)^v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) \\ & \quad = \mathsf{T}_i - \Big(v + (-1)^v\Big) \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}) = \mathsf{T}_i - (1-v) \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}), \end{aligned}$$

where the last equality is again due to $v + (-1)^v = 1 - v$ for $v \in \{0,1\}$.

We have seen that $\mathsf{Decap}(\mathsf{sk}^{1-v}, \mathsf{ct}_i - (1-v) \cdot \mathsf{C}_i) \neq \mathsf{T}_i - (1-v) \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h})$ holds regardless of whether $i \in \mathcal{S}_{\mathsf{zero}}$ or $i \in \mathcal{S}_{\mathsf{one}}$, as required. Hence, as mentioned earlier, $\mathsf{s}_i = \mathsf{s}'_i$ holds for all $i \in [n]$, and consequently we have $\mathsf{Inv}(\mathsf{td}, \mathsf{y}) = \mathsf{AltInv}(\mathsf{td}', \mathsf{y})$.

- **Case 2: There exists a position $i \in [n]$ for which there exists no pair $(\mathsf{r}, v) \in \{0,1\}^\lambda \times \{0,1\}$ satisfying $\mathsf{Encap}(\mathsf{pk}^v; \mathsf{r}) = (\mathsf{ct}_i - v \cdot \mathsf{C}_i, \mathsf{T}_i - v \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}))$.**

  In this case, both Inv and AltInv return $\perp$. Indeed, the condition of this case implies that there exists a position $i \in [n]$ for which there exists no $\mathsf{r} \in \{0,1\}^\lambda$ satisfying $\mathsf{Encap}(\mathsf{pk}^{\mathsf{s}_i}; \mathsf{r}) = (\mathsf{ct}_i^{\mathsf{s}_i} - \mathsf{s}_i \cdot \mathsf{C}_i, \mathsf{T}_i - \mathsf{s}_i \cdot (\mathsf{A}_i + \mathsf{B} \cdot \mathsf{h}))$. Hence, the validity check by re-evaluation done in the last step of Inv cannot be satisfied, and thus Inv outputs $\perp$. Exactly the same argument applies to AltInv, and thus it also outputs $\perp$. Hence, in this case we have $\mathsf{Inv}(\mathsf{td}, \mathsf{y}) = \mathsf{AltInv}(\mathsf{td}', \mathsf{y}) = \perp$.

As seen above, if ek is not bad, then for any y with $\mathsf{h} \neq \mathsf{h}^*$, we have $\mathsf{Inv}(\mathsf{td}, \mathsf{y}) = \mathsf{AltInv}(\mathsf{td}', \mathsf{y})$, as desired. $\square$ (**Lemma 9**)

Lemma 10 *There exists a PPT adversary* $\mathcal{B}^2_{\mathsf{prct}}$ *satisfying* $|\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| = \mathsf{Adv}^{\mathsf{mprct}}_{\mathsf{KEM},n,\mathcal{B}^2_{\mathsf{prct}}}(\lambda)$.

Proof of Lemma 10. The lemma can be shown very similarly to Lemma 8. The difference is that the reduction algorithm $\mathcal{B}^2_{\mathsf{prct}}$ in the proof for this lemma embeds its given public key pk' to pk^0, and uses the given challenge ciphertext/session-key pairs $\{(\mathsf{ct}'^*_{i,\beta}, \mathsf{k}'^*_{i,\beta})\}_{i \in [n]}$ as $(\mathsf{ct}_i^{*0}, \mathsf{k}_i^{*0})$ in the positions $i \in \mathcal{S}_{\mathsf{one}}$. The remaining values of ek, td', and y^* are generated by $\mathcal{B}^2_{\mathsf{prct}}$ itself in exactly the same way as in Game 5. Then $\mathcal{B}^2_{\mathsf{prct}}$ runs $\mathcal{A}^{\mathsf{AltInv}(\mathsf{td}',\cdot)}(\mathsf{ek}, \mathsf{y}^*)$, and terminates with output $\beta' \leftarrow (\mathsf{x}' \overset{?}{=} \mathsf{x}^*)$. This $\mathcal{B}^2_{\mathsf{prct}}$ perfectly simulates Game 4 (resp. Game 5) if its challenge bit β is 0 (resp. 1), and thus we can derive $\mathsf{Adv}^{\mathsf{mprct}}_{\mathsf{KEM},n,\mathcal{B}^2_{\mathsf{prct}}}(\lambda) = |\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]|$. $\square$ (**Lemma 10**)

Lemma 11 *There exists a PPT adversary* $\mathcal{B}_{\mathsf{kdm}}$ *that makes a single KDM-encryption query and satisfies* $\Pr[\mathtt{SUC}_5] \leq \mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{P},\mathcal{B}_{\mathsf{kdm}}}(\lambda) + 2^{-\ell}$.

Proof of Lemma 11. Using $\mathcal{A}$ as a building block, we show how to construct a PPT adversary $\mathcal{B}_{\mathsf{kdm}}$ that attacks the one-time $\mathcal{P}$-KDM security of SKE with the claimed advantage. The description of $\mathcal{B}_{\mathsf{kdm}}$ is as follows:

$\mathcal{B}_{\mathsf{kdm}}^{\mathcal{O}_{\mathsf{kdm}}(\cdot,\cdot)}(1^\lambda)$**:** Firstly, $\mathcal{B}_{\mathsf{kdm}}$ picks $\mathsf{r}_1^{*0},\dots,\mathsf{r}_n^{*0},\mathsf{r}_1^{*1},\dots,\mathsf{r}_n^{*1} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and $\mathsf{k}^* \xleftarrow{\mathsf{r}} \{0,1\}^\ell$. Then, $\mathcal{B}_{\mathsf{kdm}}$ defines f_1 to be the function that takes $z = (z_1,\dots,z_n) \in \{0,1\}^n$ as input and outputs $(\mathsf{r}_i^{*(z_i)})_{i\in[n]}\|\mathsf{k}^* \in \{0,1\}^{n\cdot\lambda+\ell}$, and f_0 to be the constant zero-function with output length $n\cdot\lambda+\ell$. Note that f_0 and f_1 can be expressed by a projection function, and thus $f_0, f_1 \in \mathcal{P}$. $\mathcal{B}_{\mathsf{kdm}}$ submits (f_0, f_1) as a KDM-encryption query, and receives the challenge ciphertext $\mathsf{ct}^*_{\mathsf{SKE}}$. Next, $\mathcal{B}_{\mathsf{kdm}}$ generates the remaining values of ek, td', and y^* by itself as in Game 5, where the secret key sk used to generate $\mathcal{B}_{\mathsf{kdm}}$'s challenge ciphertext $\mathsf{ct}^*_{\mathsf{SKE}}$ is regarded as s^* in Game 5. (Recall that in Game 5, values dependent on s^* appear only in the plaintext of $\mathsf{ct}^*_{\mathsf{SKE}}$.) Then, $\mathcal{B}_{\mathsf{kdm}}$ runs $\mathsf{x}' \leftarrow \mathcal{A}^{\mathsf{AltInv}(\mathsf{td}',\cdot)}(\mathsf{ek}, \mathsf{y}^*)$, where $x' = (\mathsf{s}', (\mathsf{r}'^{(\mathsf{s}'_i)_i})_{i\in[n]}, \mathsf{k}', \mathsf{r}'_{\mathsf{SKE}})$. Finally, $\mathcal{B}_{\mathsf{kdm}}$ terminates with output $\beta' \leftarrow (\mathsf{k}' \overset{?}{=} \mathsf{k}^*)$.[11]

Let $\beta \in \{0,1\}$ be $\mathcal{B}_{\mathsf{kdm}}$'s challenge bit. As mentioned above, we view the secret key sk in $\mathcal{B}_{\mathsf{kdm}}$'s $\mathcal{P}$-KDM experiment as s^* in the game simulated for $\mathcal{A}$ by $\mathcal{B}_{\mathsf{kdm}}$. Then, it is straightforward to see that if $\beta = 1$, then $\mathcal{B}_{\mathsf{kdm}}$ simulates Game 5 perfectly for $\mathcal{A}$. In particular, in this case, $\mathsf{ct}^*_{\mathsf{SKE}}$ is an encryption of $f_1(\mathsf{s}^*) = (\mathsf{r}_i^{*(\mathsf{s}^*_i)})_{i\in[n]}\|\mathsf{k}^*$, which is exactly how it is generated in Game 5. Hence, the probability that $\mathcal{A}$ succeeds in outputting $\mathsf{x}' = \mathsf{x}^* = (\mathsf{s}^*, (\mathsf{r}_i^{*(\mathsf{s}^*_i)})_{i\in[n]}, \mathsf{k}^*, \mathsf{r}^*)$ is exactly $\Pr[\mathtt{SUC}_5]$. Since $\mathsf{x}' = \mathsf{x}^*$ in particular implies $\mathsf{k}' = \mathsf{k}^*$, we have $\Pr[\mathsf{k}' = \mathsf{k}^* | \beta = 1] \ge \Pr[\mathtt{SUC}_5]$. On the other hand, if $\beta = 0$, then $\mathsf{ct}^*_{\mathsf{SKE}}$ is an encryption of $0^{n\cdot\lambda+\ell}$, and the information of k^* never appears in $\mathcal{A}$'s view. Since k^* is chosen uniformly from $\{0,1\}^\ell$, we have $\Pr[\mathsf{k}' = \mathsf{k}^* | \beta = 0] = 2^{-\ell}$. Hence, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{P},\mathcal{B}_{\mathsf{kdm}}}(\lambda) &= \left|\Pr[\beta' = 1|\beta = 1] - \Pr[\beta' = 1|\beta = 0]\right| \\ &= \left|\Pr[\mathsf{k}' = \mathsf{k}^*|\beta = 1] - \Pr[\mathsf{k}' = \mathsf{k}^*|\beta = 0]\right| \\ &\ge \Pr[\mathtt{SUC}_5] - 2^{-\ell},\end{aligned}$$

which proves the lemma. □ (**Lemma 11**)

Due to Lemmas 7 to 11 and Equation 16, we can conclude that there exist PPT adversaries $\mathcal{B}_{\mathsf{tcr}}$, $\{\mathcal{B}^2_{\mathsf{mprct}}\}_{j\in[2]}$, and $\mathcal{B}_{\mathsf{kdm}}$ (that makes a single KDM-encryption query) satisfying Equation 14.

Finally, we explain how to prove the theorem without using $\ell = \omega(\log\lambda)$ (including the case $\ell = 0$ and thus k is the empty string). The idea is to rely on the security of KEM a few more times. More specifically, consider an additional game, say Game 6, in which $\mathsf{ct}^*_{\mathsf{SKE}}$ encrypts a fixed value, say, $0^{n\cdot\lambda+\ell}$, instead of $(\mathsf{r}^{*(\mathsf{s}^*_i)})_{i\in[n]}\|\mathsf{k}^*$. (This is essentially the experiment simulated by $\mathcal{B}_{\mathsf{kdm}}$ in the case $\beta = 0$ in the proof of Lemma 11.) Then, notice that in Game 6, the information of each $\mathsf{r}_i^{*(\mathsf{s}^*_i)}$ is contained only as a ciphertext/session-key pair $(\mathsf{ct}_i^{*(\mathsf{s}_i)}, \mathsf{k}_i^{*(s^*_i)})$ in $\mathcal{A}$'s view. Thus, if $\mathcal{A}$ succeeds in outputting x' that contains some $\mathsf{r}_i^{*(\mathsf{s}^*_i)}$, it implies that such $\mathcal{A}$ is essentially breaking the pseudorandom ciphertext property (or even the IND-CPA security) of KEM. One subtle issue here is that in Game 6, we can rely on the security of KEM only for the positions $i \in \mathcal{S}_{\mathsf{zero}}$ (i.e. the ciphertexts ct_i^{*0} under pk^0) because sk^1 is needed for simulating Game 6 for $\mathcal{A}$. This subtlety can be overcome by considering one more game, say Game 7, in which the inversion oracle is changed back to $\mathsf{Inv}(\mathsf{td}, \cdot)$ (in a similar manner to Game 3), where sk^1 is now not needed and thus we can rely on the security of KEM for the positions $i \in \mathcal{S}_{\mathsf{one}}$ (i.e. the ciphertexts ct_i^{*1} under pk^1). The difference between $\mathcal{A}$'s success probability

[11]Note that $\mathcal{B}_{\mathsf{kdm}}$ does not have access to a secret key sk (which $\mathcal{B}_{\mathsf{kdm}}$ uses as s^* in the experiment simulated for $\mathcal{A}$) and the randomness $\mathsf{r}^*_{\mathsf{SKE}}$ behind the challenge ciphertext $\mathsf{ct}^*_{\mathsf{SKE}}$, and thus it cannot directly and exactly check whether $\mathcal{A}$ has succeeded in outputting $x' = \mathsf{x}^* = (\mathsf{s}^*, (\mathsf{r}_i^{*(\mathsf{s}^*_i)})_{i\in[n]}, \mathsf{k}^*, \mathsf{r}^*)$.

between Games 6 and 7 is bounded by $2\epsilon + n \cdot 2^{-\lambda+1}$ as in Lemma 9. From these observations, we can construct reduction algorithms that attacks the security of KEM from an adversary $\mathcal{A}$ that runs in Game 6 or Game 7. Furthermore, the transition from Game 5 to Game 6 can be straightforwardly done by the $\mathcal{P}$-KDM security of SKE. The difference from $\mathcal{B}_{\mathsf{kdm}}$ in the proof of Lemma 11 is that the reduction here checks if there exists $i \in [n]$ for which $\mathsf{r}_i'^{\mathsf{s}_i'} = \mathsf{r}_i^{*(\mathsf{s}_i^*)}$ holds (instead of checking $\mathsf{k}' = \mathsf{k}^*$). Hence, we can prove the adaptive one-wayness of TDF regardless of ℓ. □ (**Theorem 8**)

Acknowledgement A part of this work was supported by NTT Secure Platform Laboratories, JST OPERA JPMJOP1612, JST CREST JPMJCR14D6 and JPMJCR19F6, and JSPS KAKENHI JP16H01705 and JP17H01695.

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003.

[App11] Benny Applebaum. Key-dependent message security: Generic amplification and completeness. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 527–546. Springer, Heidelberg, May 2011.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552. Springer, Heidelberg, August 2007.

[BBS03] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemeas. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 85–99. Springer, Heidelberg, January 2003.

[BCPT13] Eleanor Birrell, Kai-Min Chung, Rafael Pass, and Sidharth Telang. Randomness-dependent message security. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 700–720. Springer, Heidelberg, March 2013.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 1–20. Springer, Heidelberg, August 2010.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2008.

[BHSV98] Mihir Bellare, Shai Halevi, Amit Sahai, and Salil P. Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 283–298. Springer, Heidelberg, August 1998.

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 1–12. Springer, Heidelberg, August 1998.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Heidelberg, April / May 2018.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BRS03] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Heidelberg, August 2003.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 3–31. Springer, Heidelberg, March 2018.

[DMN12] Nico Döttling, Jörn Müller-Quade, and Anderson C. A. Nascimento. IND-CCA secure cryptography based on a variant of the LPN problem. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 485–503. Springer, Heidelberg, December 2012.

[DNR04] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 342–360. Springer, Heidelberg, May 2004.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 53–68. Springer, Heidelberg, March 1999.

[GGH19] Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. New techniques for efficient trapdoor functions and applications. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 33–63. Springer, Heidelberg, May 2019.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982.

[GMM07] Yael Gertner, Tal Malkin, and Steven Myers. Towards a separation of semantic and CCA security for public key encryption. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 434–455. Springer, Heidelberg, February 2007.

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd FOCS*, pages 126–135. IEEE Computer Society Press, October 2001.

[HK15] Mohammad Hajiabadi and Bruce M. Kapron. Reproducible circularly-secure bit encryption: Applications and realizations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 224–243. Springer, Heidelberg, August 2015.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 92–105. Springer, Heidelberg, August 2004.

[KM19] Fuyuki Kitagawa and Takahiro Matsuda. CPA-to-CCA transformation for KDM security. *IACR Cryptology ePrint Archive*, 2019:609, 2019.

[KMO10] Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 673–692. Springer, Heidelberg, May / June 2010.

[KMP14] Eike Kiltz, Daniel Masny, and Krzysztof Pietrzak. Simple chosen-ciphertext security from low-noise LPN. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 1–18. Springer, Heidelberg, March 2014.

[KW18] Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. *IACR Cryptology ePrint Archive*, 2018:847, 2018. To appear in CRYPTO 2019.

[LQR$^+$19a] Alex Lombardi, Willy Quach, Ron D. Rothblum, Daniel Wichs, and David J. Wu. New constructions of reusable designated-verifier NIZKs. *IACR Cryptology ePrint Archive*, 2019:242, 2019. Dated on Feb 27, 2019. To appear in CRYPTO 2019.

[LQR$^+$19b] Alex Lombardi, Willy Quach, Ron D. Rothblum, Daniel Wichs, and David J. Wu. New constructions of reusable designated-verifier NIZKs. *IACR Cryptology ePrint Archive*, 2019:242, 2019. Dated on May 23, 2019. To appear in CRYPTO 2019.

[MH15] Takahiro Matsuda and Goichiro Hanaoka. Constructing and understanding chosen ciphertext security via puncturable key encapsulation mechanisms. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 561–590. Springer, Heidelberg, March 2015.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992.

[RS09] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 419–436. Springer, Heidelberg, March 2009.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 1–20. Springer, Heidelberg, February 2004.

[Wee16] Hoeteck Wee. KDM-security via homomorphic smooth projective hashing. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 159–179. Springer, Heidelberg, March 2016.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, November 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[YZ16] Yu Yu and Jiang Zhang. Cryptography with auxiliary input and trapdoor from constant-noise LPN. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 214–243. Springer, Heidelberg, August 2016.

A Other Definitions

A.1 Target Collision Resistant Hash Function

Definition 9 (Target Collision Resistant Hash Function) *A keyed hash function* Hash *consists of the two algorithms* $(\mathsf{HKG}, \mathsf{H})$*:* HKG *is the key generation algorithm that takes* 1^λ *as input, and outputs a hash key* hk*;* H *is the hash evaluation algorithm that takes* hk *and an element* $\mathsf{x} \in \{0,1\}^*$ *as input, and outputs a hash value* $\mathsf{y} \in \{0,1\}^\lambda$*.*

Hash *is said to be* target collision resistant *if for all PPT algorithms* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, we have*

$$\mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{Hash},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{l}(\mathsf{x},\mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda);\ \mathsf{hk} \leftarrow \mathsf{HKG}(1^\lambda);\ \mathsf{x}' \leftarrow \mathcal{A}_2(\mathsf{hk},\mathsf{st}) : \\ \mathsf{H}(\mathsf{hk},\mathsf{x}') = \mathsf{H}(\mathsf{hk},\mathsf{x}) \wedge \mathsf{x}' \neq \mathsf{x}\end{array}\right] = \mathsf{negl}(\lambda).$$

We can realize a target collision resistant hash function based on a one-way function [Rom90].

A.2 Public-Key Encryption

Definition 10 (Public-Key Encryption) *A public-key encryption (PKE) scheme* PKE *consists of the three PPT algorithms* $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$*:*

- KG *is the key generation algorithm that takes* 1^λ *as input, and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*. We assume that the security parameter* λ *determines the secret key space* $\mathcal{K}$ *and the message space* $\mathcal{M}$*.*

- Enc *is the encryption algorithm that takes a public key* pk *and a plaintext* m *as input, and outputs a ciphertext* ct*.*

```
$\mathsf{Expt}^{\mathsf{cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ :
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$
  $(\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{pk})$
  $b \xleftarrow{\mathsf{r}} \{0,1\}$
  $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b)$
  $b' \leftarrow \mathcal{A}_2^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{ct}^*, \mathsf{st})$
  Return $(b' \overset{?}{=} b)$.

$\mathsf{Expt}^{\mathsf{kdm}}_{\mathsf{PKE},\mathcal{F},\mathcal{A}}(\lambda)$ :
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$
  $b \xleftarrow{\mathsf{r}} \{0,1\}$
  $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{kdm}}(\cdot,\cdot)}(1^\lambda)$
  Return $(b' \overset{?}{=} b)$.

$\mathcal{O}_{\mathsf{kdm}}((f_0, f_1) \in \mathcal{F}^2)$ :
  $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, f_b(\mathsf{sk}))$
  Return $\mathsf{ct}$.
```

Figure 5: Security experiments for PKE: The IND-CCA experiment (left), the KDM security experiment (center), and the KDM-encryption oracle used in the KDM security experiment (right).

- Dec *is the (deterministic) decryption algorithm that takes a secret key* sk *and a ciphertext* ct *as input, and outputs a plaintext* m *or the invalid symbol* $\perp \notin \mathcal{M}$*.*

Let $\epsilon : \mathbb{N} \to [0,1]$*. We say that a PKE scheme* $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *is* ϵ-almost-all-keys correct *if we have*

$$\mathsf{Err}_{\mathsf{PKE}}(\lambda) := \Pr_{(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KG}(1^\lambda)}\left[\begin{array}{c}\exists(\mathsf{m},\mathsf{r}) \in \\ \mathcal{M}\times\mathcal{R}\end{array} \text{ s.t. } \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m}; \mathsf{r})) \neq \mathsf{m}\right] = \epsilon(\lambda).$$

Furthermore, we just say that PKE *is* correct *(resp.* almost-all-keys correct*) if* $\mathsf{Err}_{\mathsf{PKE}}(\lambda)$ *is zero (resp.* $\mathsf{negl}(\lambda)$*).*

Here, we recall IND-CCA, IND-CCA1, and KDM security for a PKE scheme that are treated in this paper. As in the case of SKE, for simplicity, here we only give the definition for the single key setting for KDM security.

Definition 11 (Security Notions for PKE) *Let* $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *be a PKE scheme with a secret key space* $\mathcal{K}$ *and a plaintext space* $\mathcal{M}$*, and let* $\mathcal{F}$ *be a function family with domain* $\mathcal{K}$ *and range* $\mathcal{M}$*.*

We say that PKE *is*

- IND-CCA secure *(a.k.a. IND-CCA2 secure) if for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, we have* $\mathsf{Adv}^{\mathsf{cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathsf{cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$*, where the experiment* $\mathsf{Expt}^{\mathsf{cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ *is defined as in Figure 5 (left), and in the experiment, it is required that* $|\mathsf{m}_0| = |\mathsf{m}_1|$ *and* $\mathcal{A}_2$ *is not allowed to submit* ct^* *to the decryption oracle* $\mathsf{Dec}(\mathsf{sk}, \cdot)$*.*
- IND-CCA1 secure *if for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, we have* $\mathsf{Adv}^{\mathsf{cca1}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathsf{cca1}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$*, where the experiment* $\mathsf{Expt}^{\mathsf{cca1}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ *is defined exactly as in* $\mathsf{Expt}^{\mathsf{cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$*, except that* $\mathcal{A}_2$ *is not given access to the decryption oracle* $\mathsf{Dec}(\mathsf{sk}, \cdot)$*.*
- $\mathcal{F}$-KDM secure *if for all PPT adversaries* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{PKE},\mathcal{F},\mathcal{A}}(\lambda) := 2\cdot|\Pr[\mathsf{Expt}^{\mathsf{kdm}}_{\mathsf{PKE},\mathcal{F},\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$*, where the experiment* $\mathsf{Expt}^{\mathsf{kdm}}_{\mathsf{PKE},\mathcal{F},\mathcal{A}}(\lambda)$ *is defined as in Figure 5 (center), and the KDM-encryption oracle* $\mathcal{O}_{\mathsf{kdm}}$ *is described in Figure 5 (right).*

As in the case of SKE, we will deal with the function families $\mathcal{P}$ and $\mathcal{B}_\ell$ as the function classes for KDM security of PKE. (See Section 3.2 for their definitions.)

A.3 Hinting PRG

Here, we review the definition of a hinting PRG [KW18]. We make a slight simplification to the syntax from [KW18] in that the "block length" (i.e. the output length of HEval) is fixed to be λ, instead of allowing it to be decided at the setup.

```
$\mathsf{Expt}^{\mathsf{hprg}}_{\mathsf{HPRG},\mathcal{A}}(\lambda)$ :
  $\mathsf{pp} \leftarrow \mathsf{HSetup}(1^\lambda)$
  $\mathsf{s} = (\mathsf{s}_1, \dots, \mathsf{s}_n) \xleftarrow{r} \{0,1\}^n$
  $\mathsf{y}_{0,0} \xleftarrow{r} \{0,1\}^\lambda$
  $\forall (i,v) \in [n] \times \{0,1\} : \mathsf{y}^v_{i,0} \xleftarrow{r} \{0,1\}^\lambda$
  $\mathsf{y}_{0,1} \leftarrow \mathsf{HEval}(\mathsf{pp}, \mathsf{s}, 0)$
  $\forall i \in [n]$ :
      $\mathsf{y}^{\mathsf{s}_i}_{i,1} \leftarrow \mathsf{HEval}(\mathsf{pp}, \mathsf{s}, i)$
      $\mathsf{y}^{1-\mathsf{s}_i}_{i,1} \xleftarrow{r} \{0,1\}^\lambda$
  $b \xleftarrow{r} \{0,1\}$
  $b' \leftarrow \mathcal{A}(\mathsf{pp}, (\mathsf{y}_{0,b}, (\mathsf{y}^v_{i,b})_{i \in [n], v \in \{0,1\}}))$
  Return $(b' \overset{?}{=} b)$.
```

```
$\mathsf{Expt}^{\mathsf{gc}}_{\mathsf{GC},\mathcal{A}}(\lambda)$ :
  $(C, \mathsf{x} = (\mathsf{x}_1, \dots, \mathsf{x}_n), \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda)$
  $(\widetilde{C}_0, (\mathsf{lab}^v_{i,0})_{i \in [n], v \in \{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)$
  $(\widetilde{C}_1, (\mathsf{lab}^{\mathsf{x}_i}_{i,1})_{i \in [n]}) \leftarrow \mathsf{Sim}(1^\lambda, |C|, C(\mathsf{x}))$
  $b \xleftarrow{r} \{0,1\}$
  $b' \leftarrow \mathcal{A}_2(\widetilde{C}_b, (\mathsf{lab}^{\mathsf{x}_i}_{i,b})_{i \in [n]}, \mathsf{st})$
  Return $(b' \overset{?}{=} b)$.
```

Figure 6: The security experiments for a hinting PRG (left) and that for a circuit garbling scheme (right).

Definition 12 (Hinting PRG) *A* hinting PRG HPRG *consists of the two PPT algorithms* (HSetup, HEval) *with the following syntax.*

- HSetup *is the setup algorithm that takes* 1^λ *as inputs, and outputs a public parameter* pp*. We assume that* pp *specifies the seed length* $n = n(\lambda)$*.*
- HEval *is the evaluation algorithm (for computing each "block") that takes a public parameter* pp*, a seed* $\mathsf{s} \in \{0,1\}^n$*, and an index* $i \in \{0\} \cup [n]$ *as input, and outputs a string* $\mathsf{y} \in \{0,1\}^\lambda$*.*[12]

Security *We say that* HPRG *is a secure hinting PRG if for all PPT adversaries* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{hprg}}_{\mathsf{HPRG},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathsf{hprg}}_{\mathsf{HPRG},\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$*, where the experiment* $\mathsf{Expt}^{\mathsf{hprg}}_{\mathsf{HPRG},\mathcal{A}}(\lambda)$ *is defined as in Figure 6 (left).*

A.4 Garbled Circuits

Here, we review the definition of a circuit garbling scheme [Yao86].

Definition 13 (Circuit Garbling) *Let* $\{\mathcal{C}_n\}_{n \in \mathbb{N}}$ *be a family of circuits where the input length of each circuit in* $\mathcal{C}_n$ *is* n*. A* circuit garbling scheme GC *consists of the three PPT algorithms* (Garble, Eval, Sim)*:*

- Garble *is the garbling algorithm that takes* 1^λ *and a circuit* $C \in \mathcal{C}_n$ *as input, and outputs a garbled circuit* $\widetilde{C}$ *together with* $2n$ *labels* $(\mathsf{lab}^v_i)_{i \in [n], v \in \{0,1\}}$*. For simplicity and without loss of generality, we assume that the length of each* lab^v_i *is* λ*.*
- Eval *is the (deterministic) evaluation algorithm that takes a garbled circuit* $\widetilde{C}$ *and* n *labels* $(\mathsf{lab}_i)_{i \in [n]}$ *as input, and outputs an evaluation result* y*.*
- Sim *is the simulator algorithm that takes* 1^λ*, the size parameter* size *(of a circuit), and a string* y *as input, and outputs a simulated garbled circuit* $\widetilde{C}$ *and* n *labels* $(\mathsf{lab}_i)_{i \in [n]}$*.*

[12]One can understand that a hinting PRG stretches a seed $\mathsf{s} \in \{0,1\}^n$ to a string $\mathsf{y}_0 \| \dots \| \mathsf{y}_n \in \{0,1\}^{(n+1)\cdot\lambda}$ where $\mathsf{y}_i = \mathsf{HEval}(\mathsf{pp}, \mathsf{s}, i)$ for each $i \in \{0\} \cup [n]$.

Correctness *A circuit garbling scheme* $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ *is said to be* correct *if for all* $\lambda, n \in \mathbb{N}$, *all circuits* $C \in \mathcal{C}_n$, *all strings* $\mathsf{x} = (\mathsf{x}_1, \dots, \mathsf{x}_n) \in \{0,1\}^n$, *and all* $(\widetilde{C}, (\mathsf{lab}_i^v)_{i \in [n], v \in \{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)$, *it holds that* $\mathsf{Eval}(\widetilde{C}, (\mathsf{lab}_i^{\mathsf{x}_i})_{i \in [n]}) = C(\mathsf{x})$.

Security *We say that a circuit garbling scheme* $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ *is secure if for all PPT adversaries* $\mathcal{A}$, *we have* $\mathsf{Adv}^{\mathsf{gc}}_{\mathsf{GC},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathsf{gc}}_{\mathsf{GC},\mathcal{A}}(\lambda) = 1] - 1/2|$, *where the experiment* $\mathsf{Expt}^{\mathsf{gc}}_{\mathsf{GC},\mathcal{A}}(\lambda)$ *is defined as in Figure 6 (right).*

We can realize a circuit garbling scheme for all efficiently computable circuits based on a one-way function [Yao86].

A.5 Standard PRG

Definition 14 (Standard PRG) *We say that an efficiently computable function* $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^\ell$ *(for some polynomial* $\ell = \ell(\lambda) > \lambda$*) is a secure* pseudorandom generator *(PRG) if for all PPT adversaries* $\mathcal{A}$, *we have*

$$\mathsf{Adv}^{\mathsf{prg}}_{\mathsf{G},\mathcal{A}}(\lambda) := \left| \Pr_{\mathsf{s} \xleftarrow{r} \{0,1\}^\lambda}[\mathcal{A}(\mathsf{G}(\mathsf{s})) = 1] - \Pr_{\mathsf{y} \xleftarrow{r} \{0,1\}^\ell}[\mathcal{A}(\mathsf{y}) = 1] \right| = \mathsf{negl}(\lambda).$$

B One-time KDM Secure SKE Based on Hinting PRG

In this section, we show how to construct an SKE scheme that is one-time $\mathcal{B}_{\mathsf{size}}$-KDM secure (i.e., one-time KDM secure with respect to circuits whose size is bounded by an a priori determined polynomial $\mathsf{size} = \mathsf{size}(\lambda)$), using a hinting PRG introduced by Koppula and Waters [KW18]. We first show our construction, then give a security proof (Theorem 9), and finally explain how Theorem 3 stated in Section 4.1 is proved. Our construction of SKE uses a hinting PRG, a circuit garbling scheme, and a standard PRG as building blocks, whose formal definitions are recalled in Appendix A.

Formally, let $m = m(\lambda)$ be a polynomial that denotes the plaintext length that we wish to encrypt by our SKE scheme, and let $\mathsf{size} = \mathsf{size}(\lambda) \geq m$ be any polynomial that denotes the size of circuits for which we aim at achieving $\mathcal{B}_{\mathsf{size}}$-KDM security. Then,

- Let $\mathsf{HPRG} = (\mathsf{HSetup}, \mathsf{HEval})$ be a hinting PRG scheme whose seed length is $n = n(\lambda)$.
- Let $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ be a circuit garbling scheme, where its label length is λ. Let $\ell = \ell(\lambda)$ denote a polynomial that denotes the size of a garbled circuit $\widetilde{C}$ output by $\mathsf{Garble}(1^\lambda, C)$ in case C is a circuit whose input length is m and $|C| = \mathsf{size}$.
- Let $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^\ell$ be a standard PRG.

Using these ingredients, we construct an SKE scheme $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ with plaintext space $\{0,1\}^m$ as described in Figure 7. In the figure, $\mathsf{P}[\mathsf{m}]$ denotes a "constant" circuit of size size that has $\mathsf{m} \in \{0,1\}^m$ hardwired, and it always outputs m for any n-bit input.

We remark that if we adopt the syntax of an SKE scheme in which there is a setup algorithm that generates a public parameter shared by all users, then the generation of the public parameter pp done in E can be moved to the setup, and pp can be removed from a ciphertext CT.

The correctness of SKE follows from that of GC. Moreover, SKE is one-time KDM secure with respect to functions computable by circuits of a priori bounded size size ($\mathcal{B}_{\mathsf{size}}$-KDM secure). More precisely, the following theorem holds.

```
$\mathsf{K}(1^\lambda)$ :
  Return $\mathsf{sk} = (\mathsf{s}_1, \ldots, \mathsf{s}_n) \xleftarrow{\mathsf{r}} \{0,1\}^n$.
```

```
$\mathsf{E}(\mathsf{sk} = (\mathsf{s}_1, \ldots, \mathsf{s}_n), \mathsf{m})$ :
  $\mathsf{pp} \leftarrow \mathsf{HSetup}(1^\lambda)$
  $\left(\widetilde{\mathsf{P}}, (\mathsf{lab}_i^v)_{i\in[n],v\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, \mathsf{P}[\mathsf{m}])$
  $\mathsf{y}_0 \leftarrow \mathsf{HEval}(\mathsf{pp}, \mathsf{sk}, 0)$
  $\mathsf{ct}_0 \leftarrow \widetilde{\mathsf{P}} \oplus \mathsf{G}(\mathsf{y}_0)$
  $\forall i \in [n]$ :
      $\mathsf{y}_i^{\mathsf{s}_i} \leftarrow \mathsf{HEval}(\mathsf{pp}, \mathsf{sk}, i)$
      $\mathsf{ct}_i^{\mathsf{s}_i} \leftarrow \mathsf{lab}_i^{\mathsf{s}_i} \oplus \mathsf{y}_i^{\mathsf{s}_i}$
      $\mathsf{ct}_i^{1-\mathsf{s}_i} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$
  Return $\mathsf{CT} \leftarrow (\mathsf{pp}, \mathsf{ct}_0, (\mathsf{ct}_i^v)_{i\in[n],v\in\{0,1\}})$.
```

```
$\mathsf{D}(\mathsf{sk} = (\mathsf{s}_1, \ldots, \mathsf{s}_n), \mathsf{CT})$ :
  $(\mathsf{pp}, \mathsf{ct}_0, (\mathsf{ct}_i^v)_{i\in[n],v\in\{0,1\}}) \leftarrow \mathsf{CT}$
  $\mathsf{y}_0 \leftarrow \mathsf{HEval}(\mathsf{pp}, \mathsf{sk}, 0)$
  $\widetilde{\mathsf{P}} \leftarrow \mathsf{ct}_0 \oplus \mathsf{G}(\mathsf{y}_0)$
  $\forall i \in [n]$ :
      $\mathsf{y}_i \leftarrow \mathsf{HEval}(\mathsf{pp}, \mathsf{sk}, i)$
      $\mathsf{lab}_i \leftarrow \mathsf{ct}_i^{\mathsf{s}_i} \oplus \mathsf{y}_i$
  Return $\mathsf{m} \leftarrow \mathsf{Eval}(\widetilde{\mathsf{P}}, (\mathsf{lab}_i)_{i\in[n]})$.
```

Figure 7: The construction of an SKE scheme SKE. In E, $\mathsf{P}[\mathsf{m}]$ is padded to size-bit.

Theorem 9 *Let $m = m(\lambda)$ and $\mathsf{size} = \mathsf{size}(\lambda) \geq m$ be any polynomials. Assume that HPRG is a secure hinting PRG, GC is a secure circuit garbling scheme, and G is a secure (standard) PRG. Then, SKE is a one-time $\mathcal{B}_{\mathsf{size}}$-KDM secure SKE scheme with m-bit plaintext space.*

Proof of Theorem 9. Let $\mathsf{size} = \mathsf{size}(\lambda)$ be a polynomial. Let $\mathcal{A}$ be an arbitrary PPT adversary that attacks the one-time $\mathcal{B}_{\mathsf{size}}$-KDM security of SKE. We proceed the proof via a sequence of games argument with six games. For every $j \in [6]$, let $\mathtt{SUC}_j$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $b' = b$ occurs) in Game j.

Game 1: This is the KDM experiment $\mathsf{Expt}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{B}_{\mathsf{size}},\mathcal{A}}(\lambda)$. The detailed description is as follows.

- Pick a secret-key $\mathsf{sk} = (\mathsf{s}_1, \ldots, \mathsf{s}_n) \xleftarrow{\mathsf{r}} \{0,1\}^n$ and the challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$, and run $\mathcal{A}(1^\lambda)$.
- Since we consider the one-time $\mathcal{B}_{\mathsf{size}}$-KDM security of SKE, $\mathcal{A}$ is allowed to make at most a single KDM-encryption query. $\mathcal{A}$'s KDM-encryption query $(f_0, f_1) \in (\mathcal{B}_{\mathsf{size}})^2$ is answered with $\mathsf{CT} = (\mathsf{pp}, \mathsf{ct}_0, (\mathsf{ct}_i^v)_{i\in[n],v\in\{0,1\}})$ computed as follows:
  1. Set $\mathsf{m} := f_b(\mathsf{sk})$ and generate $\mathsf{pp} \leftarrow \mathsf{HPRG}(1^\lambda)$.
  2. Compute $(\widetilde{\mathsf{P}}, (\mathsf{lab}_i^v)_{i\in[n],v\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, \mathsf{P}[\mathsf{m}])$.
  3. Compute $\mathsf{y}_0 \leftarrow \mathsf{HEval}(\mathsf{pp}, \mathsf{sk}, 0)$ and $\mathsf{ct}_0 \leftarrow \widetilde{\mathsf{P}} \oplus \mathsf{G}(\mathsf{y}_0)$
  4. For every $i \in [n]$, compute $\mathsf{y}_i^{\mathsf{s}_i} \leftarrow \mathsf{HEval}(\mathsf{pp}, \mathsf{sk}, i)$ and $\mathsf{ct}_i^{\mathsf{s}_i} \leftarrow \mathsf{lab}_i^{\mathsf{s}_i} \oplus \mathsf{y}_i^{\mathsf{s}_i}$, and pick $\mathsf{ct}_i^{1-\mathsf{s}_i} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$.
  5. Set $\mathsf{CT} \leftarrow (\mathsf{pp}, \mathsf{ct}_0, (\mathsf{ct}_i^v)_{i\in[n],v\in\{0,1\}})$.
- $\mathcal{A}$ terminates with output $b' \in \{0,1\}$.

Game 2: Same as Game 1, except that $(\widetilde{\mathsf{P}}, (\mathsf{lab}_i^{\mathsf{s}_i})_{i\in[n]})$ is computed by $(\widetilde{\mathsf{P}}, (\mathsf{lab}_i^{\mathsf{s}_i})_{i\in[n]}) \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{size}, f_b(\mathsf{sk}))$.

Note that in Game 1, the information of $(\mathsf{lab}_i^{1-\mathsf{s}_i})_{i\in[n]}$ is hidden from $\mathcal{A}$'s view. Hence, by the security of GC, we can derive $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| = \mathsf{negl}(\lambda)$.

Game 3: Same as Game 2, except that $(\widetilde{\mathsf{P}}, (\mathsf{lab}_i^{\mathsf{s}_i})_{i\in[n]})$ is computed by $(\widetilde{\mathsf{P}}, (\mathsf{lab}_i^v)_{i\in[n],v\in\{0,1\}}) \leftarrow \mathsf{GC}(1^\lambda, f_b)$.

By the security of GC again, we obtain $|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]| = \mathsf{negl}(\lambda)$.

Note that due to the change made in this game, sk is now used only for computing $\mathsf{y}_0 = \mathsf{HEval}(\mathsf{pp}, \mathsf{sk}, 0)$ and $(\mathsf{y}_i^{\mathsf{s}_i} = \mathsf{HEval}(\mathsf{pp}, \mathsf{sk}, i))_{i\in[n]}$.

Game 4: Same as Game 3, except that each $\mathsf{ct}_i^{1-\mathsf{s}_i}$ is computed by $\mathsf{ct}_i^{1-\mathsf{s}_i} \leftarrow \mathsf{lab}_i^{1-\mathsf{s}_i} \oplus \mathsf{y}_i^{1-\mathsf{s}_i}$ for every $i \in [n]$, where $\mathsf{y}_i^{1-\mathsf{s}_i} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$.

$(\mathsf{ct}_i^{1-\mathsf{s}_i})_{i\in[n]}$ in Game 3 and those in Game 4 are distributed identically from $\mathcal{A}$'s view. Thus, we have $|\Pr[\mathtt{SUC}_3] - \Pr[\mathtt{SUC}_4]| = 0$.

Game 5: Same as Game 4, except that y_0 and all of $(\mathsf{y}_i^{\mathsf{s}_i})_{i\in[n]}$ are chosen uniformly at random from $\{0,1\}^\lambda$.

By the security of HPRG, we have $|\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| = \mathsf{negl}(\lambda)$.

Game 6: Same as Game 5, except that $\mathsf{G}(\mathsf{y}_0)$ is replaced with $\mathsf{y}_0' \xleftarrow{\mathsf{r}} \{0,1\}^\ell$.

By the security of G, we have $|\Pr[\mathtt{SUC}_5] - \Pr[\mathtt{SUC}_6]| = \mathsf{negl}(\lambda)$.

Notice that in Game 6, ct_0 and all of $(\mathsf{ct}_i^v)_{i\in[n],v\in\{0,1\}}$ are distributed uniformly and independently of one another. This is because in this game, y_0' and $(\mathsf{y}_{i,v})_{i\in[n],v\in\{0,1\}}$ are all chosen uniformly and independently, and ct_0 and each of ct_i^v are generated as $\mathsf{ct}_0 = \widetilde{\mathsf{P}} \oplus \mathsf{y}_0'$ and $\mathsf{ct}_i^v = \mathsf{lab}_i^v \oplus \mathsf{y}_i^v$ for every $(i,v) \in [n] \times \{0,1\}$, respectively. Thus, the information of b is completely hidden from $\mathcal{A}$'s view, and we have $\Pr[\mathtt{SUC}_6] = 1/2$.

From the above arguments, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{SKE},\mathcal{B}_{\mathsf{size}},\mathcal{A}}(\lambda) &= 2 \cdot \left|\Pr[\mathtt{SUC}_1] - \frac{1}{2}\right| \\ &\le 2 \cdot \left(\sum_{j\in[5]} |\Pr[\mathtt{SUC}_j] - \Pr[\mathtt{SUC}_{j+1}]| + \left|\Pr[\mathtt{SUC}_6] - \frac{1}{2}\right|\right) = \mathsf{negl}(\lambda).\end{aligned}$$

Since the choice of $\mathcal{A}$ was arbitrary, we can conclude that SKE is one-time $\mathcal{B}_{\mathsf{size}}$-KDM secure. □ (**Theorem 9**)

Finally, we give the proof of Theorem 3 (stated in Section 4.1).

Proof of Theorem 3 (in Section 4.1). Firstly, the statement about the existence of $\mathcal{B}_{\mathsf{size}}$-KDM secure SKE in Theorem 3 is immediate from our construction SKE and Theorem 9.

To see that the statement about the fully black-box construction of a $\mathcal{P}$-KDM secure SKE scheme from a hinting PRG is true, we explain that if we focus only on $\mathcal{P}$-KDM security (as opposed to $\mathcal{B}_{\mathsf{size}}$-KDM security), then our construction SKE is a fully black-box construction from a hinting PRG scheme.

Firstly, it is clear that our construction SKE uses the building blocks (a hinting PRG HPRG, a circuit garbling scheme GC, and a standard PRG G) in a black-box manner. Since a circuit garbling scheme and a standard PRG can be constructed from a hinting PRG in a black-box manner, our construction SKE can be seen as being constructed using a hinting PRG in a black-box way.

It remains to see that the security reductions used for proving the $\mathcal{P}$-KDM security of SKE treats the underlying primitives (in this case, HPRG, GC, and G), and an adversary in a black-box manner. The security reductions in the proof of Theorem 9 obviously need not treat the building block primitives in a non-black-box manner. However, there is a subtle issue: Recall that in the KDM security game, a fully black-box reduction needs to treat not only an adversary itself, but also the adversary's KDM-encryption query as a black-box. However, as seen in the proof of

Theorem 9 above, the reduction algorithms that simulate Game 3 and/or the subsequent games would need to garble the adversary's KDM-encryption query f_b. This may seem to lead to a non-black-box treatment of the KDM-encryption query f_b. However, recall that a projection function is learnable in the sense that its canonical description can be completely recovered by making only polynomially many oracle queries (in its input length and output length) to it and observing the outputs. Thus, we can conduct the security proofs without treating the adversary and its KDM-encryption query in a non-black-box way. $\square$ (**Theorem 3**)

C (Ordinary) One-Way TDF

In this section, we show a TDF achieving (ordinary) one-wayness. The construction is a simpler variant of our adaptive one-way construction presented in Section 6. In particular, we need not use a target collision resistant hash function.

Formally, let $\ell = \ell(\lambda)$ be a polynomial. Our one-way TDF uses the building blocks KEM and SKE with the following properties:

- $\mathsf{KEM} = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ is a KEM such that (1) its session key space is $\{0,1\}^{2\lambda}$, (2) the randomness space of Encap is $\{0,1\}^{\lambda}$, and (3) the ciphertext space $\mathcal{C}$ forms an abelian group (where we use the additive notation) and satisfies $|\mathcal{C}| \geq 2^{2\lambda}$.

- $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ is an SKE scheme such that (1) it has the randomness-recovering decryption property (with the randomness-recovering decryption algorithm RD), (2) its secret key space is $\{0,1\}^n$ for some polynomial $n = n(\lambda)$, and (3) the plaintext space is $\{0,1\}^{n \cdot \lambda + \ell}$.

  We denote the randomness space of E by $\mathcal{R}_{\mathsf{SKE}}$.

Using these building blocks, our TDF $\mathsf{TDF}' = (\mathsf{Setup}', \mathsf{Samp}', \mathsf{Eval}', \mathsf{Inv}')$ with one-wayness is constructed as in Figure 8. The domain $\mathcal{X}$ of TDF' is $\mathcal{X} = \{0,1\}^n \times \{0,1\}^{n \cdot \lambda} \times \mathcal{R}_{\mathsf{SKE}}$.

As in the case of our adaptive one-way construction in Section 6, $\mathsf{k} \in \{0,1\}^{\ell}$ in a domain element can be used as hard-core bits.

The correctness and one-wayness of TDF' are guaranteed by the following theorems. We omit their proofs since they are very similar to (actually, only simpler than) those of Theorems 7 and 8.

Theorem 10 *Let $\epsilon = \epsilon(\lambda) \in [0,1]$. If KEM is ϵ-almost-all-keys correct and SKE has the randomness-recovering decryption property, then TDF' is $(\epsilon + n \cdot 2^{-\lambda})$-almost-all-keys correct.*

Theorem 11 *Assume that KEM satisfies the pseudorandom ciphertext property and almost-all-keys correctness, and SKE is one-time $\mathcal{P}$-KDM secure. Then, TDF' is one-way.*

```
$\mathsf{Setup}'(1^\lambda)$ :
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KKG}(1^\lambda)$
    $\mathsf{A}_1, \ldots, \mathsf{A}_n \xleftarrow{\mathsf{r}} \{0,1\}^{2\lambda}$
    $\mathsf{C}_1, \ldots, \mathsf{C}_n \xleftarrow{\mathsf{r}} \mathcal{C}$
    $\mathsf{ek} \leftarrow (\mathsf{pk}, (\mathsf{A}_i, \mathsf{C}_i)_{i \in [n]})$
    $\mathsf{td} \leftarrow (\mathsf{sk}, \mathsf{ek})$
    Return $(\mathsf{ek}, \mathsf{td})$.
```

```
$\mathsf{Samp}'(1^\lambda)$ :
    $\mathsf{s} = (\mathsf{s}_1, \ldots, \mathsf{s}_n) \leftarrow \mathsf{K}(1^\lambda)$
    $\mathsf{r}_1^{\mathsf{s}_1}, \ldots, \mathsf{r}_n^{\mathsf{s}_n} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$
    $\mathsf{k} \xleftarrow{\mathsf{r}} \{0,1\}^\ell$
    Sample $\mathsf{r}_{\mathsf{SKE}} \in \mathcal{R}_{\mathsf{SKE}}$
                in the same way as in $\mathsf{E}$.
    Return $\mathsf{x} \leftarrow (\mathsf{s}, (\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]}, \mathsf{k}, \mathsf{r}_{\mathsf{SKE}})$.
```

```
$\mathsf{Eval}'(\mathsf{ek}, \mathsf{x})$ :
    $(\mathsf{pk}, (\mathsf{A}_i, \mathsf{C}_i)_{i \in [n]}) \leftarrow \mathsf{ek}$
    $(\mathsf{s} = (\mathsf{s}_1, \ldots, \mathsf{s}_n), (\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]}, \mathsf{k}, \mathsf{r}_{\mathsf{SKE}}) \leftarrow \mathsf{x}$
    $\mathsf{ct}_{\mathsf{SKE}} \leftarrow \mathsf{E}(\mathsf{s}, (\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]} \| \mathsf{k}; \mathsf{r}_{\mathsf{SKE}})$
    $\forall i \in [n]$ :
        $(\mathsf{ct}_i^{\mathsf{s}_i}, \mathsf{k}_i^{s_i}) \leftarrow \mathsf{Encap}(\mathsf{pk}; \mathsf{r}_i^{\mathsf{s}_i})$
        $\mathsf{ct}_i \leftarrow \mathsf{ct}_i^{\mathsf{s}_i} + \mathsf{s}_i \cdot \mathsf{C}_i$   (‡)
            $= \begin{cases} \mathsf{ct}_i^0 & \text{if } \mathsf{s}_i = 0 \\ \mathsf{ct}_i^1 + \mathsf{C}_i & \text{if } \mathsf{s}_i = 1 \end{cases}$
        $\mathsf{T}_i \leftarrow \mathsf{k}_i^{\mathsf{s}_i} + \mathsf{s}_i \cdot \mathsf{A}_i$   (†)
            $= \begin{cases} \mathsf{k}_i^0 & \text{if } \mathsf{s}_i = 0 \\ \mathsf{k}_i^1 + \mathsf{A}_i & \text{if } \mathsf{s}_i = 1 \end{cases}$
    Return $\mathsf{y} \leftarrow ((\mathsf{ct}_i, \mathsf{T}_i)_{i \in [n]}, \mathsf{ct}_{\mathsf{SKE}})$.
```

```
$\mathsf{Inv}'(\mathsf{td}, \mathsf{y})$ :
    $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{td}$
    $(\mathsf{pk}, (\mathsf{A}_i, \mathsf{C}_i)_{i \in [n]}) \leftarrow \mathsf{ek}$
    $((\mathsf{ct}_i, \mathsf{T}_i)_{i \in [n]}, \mathsf{ct}_{\mathsf{SKE}}) \leftarrow \mathsf{y}$
    $\forall i \in [n]$ :
        $\mathsf{s}_i \leftarrow 1 - (\mathsf{Decap}(\mathsf{sk}, \mathsf{ct}_i) \stackrel{?}{=} \mathsf{T}_i)$
            $= \begin{cases} 0 & \text{if } \mathsf{Decap}(\mathsf{sk}, \mathsf{ct}_i) = \mathsf{T}_i \\ 1 & \text{otherwise} \end{cases}$
    $\mathsf{s} \leftarrow (\mathsf{s}_1, \ldots, \mathsf{s}_n) \in \{0,1\}^n$
    $(\mathsf{m}, \mathsf{r}_{\mathsf{SKE}}) \leftarrow \mathsf{RD}(\mathsf{s}, \mathsf{ct}_{\mathsf{SKE}})$
    Parse $\mathsf{m}$ as $(\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]} \in \{0,1\}^{n \cdot \lambda}$
                and $\mathsf{k} \in \{0,1\}^\ell$.
    Return $\mathsf{x} \leftarrow (\mathsf{s}, (\mathsf{r}_i^{\mathsf{s}_i})_{i \in [n]}, \mathsf{k}, \mathsf{r}_{\mathsf{SKE}})$.
```

Figure 8: The proposed TDF TDF' with one-wayness. (†) The arithmetic is done over $\mathrm{GF}(2^{2\lambda})$ where we identify $\{0,1\}^{2\lambda}$ with $\mathrm{GF}(2^{2\lambda})$. (‡) The addition is done over $\mathcal{C}$.