CPA-to-CCA Transformation for KDM Security

Fuyuki Kitagawa[1] and Takahiro Matsuda[2]

[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

Abstract

We show that chosen plaintext attacks (CPA) security is equivalent to chosen ciphertext attacks (CCA) security for key-dependent message (KDM) security. Concretely, we show how to construct a public-key encryption (PKE) scheme that is KDM-CCA secure with respect to all functions computable by circuits of a-priori bounded size, based only on a PKE scheme that is KDM-CPA secure with respect to projection functions. Our construction works for KDM security in the single user setting.

Our main result is achieved by combining the following two steps. First, we observe that by combining the results and techniques from the recent works by Lombardi et al. (CRYPTO 2019), and by Kitagawa et al. (CRYPTO 2019), we can construct a reusable designated-verifier non-interactive zero-knowledge (DV-NIZK) argument system 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. This observation leads to the first reusable DV-NIZK argument system under the learning-parity-with-noise (LPN) assumption. Then, as the second and main technical step, we show a generic construction of a KDM-CCA secure PKE scheme using an IND-CPA secure PKE scheme, a reusable DV-NIZK argument system, and an SKE scheme satisfying one-time KDM security with respect to projection functions. Since the classical Naor-Yung paradigm (STOC 1990) with a DV-NIZK argument system does not work for proving KDM security, we propose a new construction methodology to achieve this generic construction.

Moreover, we show how to extend our generic construction and achieve KDM-CCA security in the multi-user setting, by additionally requiring the underlying SKE scheme in our generic construction to satisfy a weak form of KDM security against related-key attacks (RKA-KDM security) instead of one-time KDM security. From this extension, we obtain the first KDM-CCA secure PKE schemes in the multi-user setting under the CDH or LPN assumption.

Keywords: public-key encryption, key-dependent message security, chosen ciphertext security, designated-verifier non-interactive zero-knowledge argument.

Contents

1 Introduction

1.1 Background

The most basic security notion for public-key encryption (PKE) is indistinguishability against chosen plaintext attacks (IND-CPA security) [GM82]. Intuitively, IND-CPA security guarantees that an adversary can obtain no information about a message from its encryption, except for its length. However, in practice, PKE schemes should satisfy the stronger notion of indistinguishability against chosen ciphertext attacks (IND-CCA security) [NY90, RS92]. IND-CCA security implies non-malleability [DDN91, BDPR98], and provides security guarantees against active adversaries [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 from both the theoretical and practical points of view.

In the literature, a number of efforts have been made for (implicitly or explicitly) tackling the problem.[1] Among them, we highlight the two very recent works that make solid progress. Koppula and Waters [KW18] showed that an IND-CCA secure PKE scheme can be constructed from an IND-CPA secure one by using a pseudorandom generator (PRG) satisfying a special security notion. This additional primitive is called a *hinting PRG*. Subsequently, Kitagawa, Matsuda, and Tanaka [KMT19] showed that a transformation from an IND-CPA secure PKE scheme to an IND-CCA secure one is also possible by using a secret-key encryption (SKE) scheme satisfying one-time key-dependent message security [BRS03] instead of a hinting PRG.

We further study the question of CPA security vs CCA security. Many previous works focusing on this question (some of which we review in Section 1.3) sought an additional assumption that bridges IND-CPA security and IND-CCA security. In this work, we tackle the question from a somewhat different angle. Concretely, we aim at finding a security notion under which CPA security and CCA security are equivalent. As far as we know, such an equivalence is not known for any security notion for PKE schemes (e.g., leakage resilience, key-dependent message security, and selective opening security). Finding such a security notion is an important question in the theoretical study of public-key cryptography. Moreover, we believe that clarifying for what types of notions CPA security and CCA security are equivalent potentially gives us new insights for the major open question on the equivalence between IND-CPA security and IND-CCA security.

Based on the above motivation, in this work, we study the equivalence of CPA security and CCA security for *key-dependent message (KDM) security* [BRS03]. Informally, KDM security guarantees that an encryption scheme can securely encrypt messages that depend on its own secret key. We can see some connections between IND-CCA security and KDM-CPA security from several previous results [MH15, HK15, KMT19], and thus KDM security can be considered as one of the best candidates for which CPA security and CCA security could be shown equivalent. Moreover, KDM security is important and interesting enough to be studied in its own right since it has found a number of applications in both theoretical and practical studies in cryptography, e.g., anonymous credentials [CL02], formal methods [ABHS05], hard-disc encryption [BHHO08], fully homomorphic encryption [Gen09], non-interactive zero-knowledge proofs [CCRR18, CCH+18, CLW18], and homomorphic secret-sharing [BKS19].

[1] We review some of them in Section 1.3.

1.2 Our Results

As noted above, we study the equivalence between CPA security and CCA security for KDM security. Then, we obtain the following main theorem.

Theorem 1 (Informal) *Assume that there exists a KDM-CPA secure PKE scheme. Then, there exists a KDM-CCA secure PKE scheme.*

We show this theorem for KDM-CPA security and KDM-CCA security in the single user setting. The underlying scheme needs to be KDM-CPA secure with respect to functions called *projection functions* ($\mathcal{P}$-KDM-CPA secure). The family of projection functions is one of the simplest classes of functions, and KDM security with respect to this function class has been widely studied [BHHO08, ACPS09, BG10, BLSV18, DGHM18]. The resulting scheme is KDM-CCA secure with respect to all functions computable by circuits of a-priori bounded size.

We obtain Theorem 1 by combining the following two steps.

Reusable DV-NIZK Based on One-Time KDM Secure SKE. A *designated-verifier non-interactive zero-knowledge* (DV-NIZK) argument system is a relaxation of a standard NIZK argument system in the common reference string model (CRS-NIZK, for short), and allows a verifier to have its own public/secret key pair; The public key is used to generate a proof non-interactively, which can be verified by using the corresponding secret key. A DV-NIZK argument system is said to be *reusable* if its soundness (resp. zero-knowledge property) is maintained even if an adversary can make multiple verification (resp. proving) queries. It was recently shown by Lombardi, Quach, Rothblum, Wichs, and Wu [LQR+19a] that a reusable DV-NIZK argument system can be constructed from the combination of an IND-CPA secure PKE scheme and a hinting PRG introduced by Koppula and Waters [KW18].

As the first step for Theorem 1, we observe that we can construct a reusable DV-NIZK argument system based on an IND-CPA secure PKE scheme and an SKE scheme that is one-time KDM secure with respect to projection functions (one-time $\mathcal{P}$-KDM secure), by combining the results and techniques from the recent works by Lombardi et al. [LQR+19a] and Kitagawa et al. [KMT19].

In fact, this is somewhat obvious from the results [LQR+19a, KMT19] and not our main contribution. However, this observation leads to the following interesting implications. A one-time $\mathcal{P}$-KDM secure SKE scheme can be constructed based on the polynomial hardness of the constant-noise learning-parity-with-noise (LPN) assumption [ACPS09]. Moreover, we can construct an IND-CPA secure PKE scheme based on the polynomial hardness of the low-noise LPN assumption [Ale03] or the sub-exponential hardness of the constant-noise LPN assumption [YZ16]. Thus, combined together, our observation leads to the first reusable DV-NIZK argument system based on either the polynomial hardness of the low-noise LPN assumption or the sub-exponential hardness of the constant-noise LPN assumption.

We note that the exact same observation (i.e. a reusable DV-NIZK argument system based on IND-CPA secure PKE and one-time $\mathcal{P}$-KDM secure SKE, and the LPN-based instantiation) was very recently made independently and concurrently by Lombardi et al. in the latest update of their paper [LQR+19b] (dated on May 23, 2019).

Generic Construction of KDM-CCA Secure PKE Using Reusable DV-NIZK. Then, as the second and main technical step for Theorem 1, we show a generic construction of KDM-CCA secure PKE based on the following five building blocks.

- An IND-CPA secure PKE scheme

- An IND-CCA secure PKE scheme
- A one-time $\mathcal{P}$-KDM secure SKE scheme
- A garbling scheme
- A reusable DV-NIZK argument system

In the first step above, we show how to construct a reusable DV-NIZK argument system from an IND-CPA secure PKE scheme and a one-time $\mathcal{P}$-KDM secure SKE scheme. Also, IND-CCA secure PKE can be constructed from the same building blocks [KMT19]. Moreover, a garbling scheme can be constructed from one-way functions [Yao86], which is in turn implied by other building blocks. Therefore, through our generic construction, we can construct a KDM-CCA secure PKE scheme based on an IND-CPA secure PKE scheme and a one-time $\mathcal{P}$-KDM secure SKE scheme. Since both of the underlying primitives are implied by $\mathcal{P}$-KDM-CPA secure PKE, we obtain Theorem 1.

One might think that if we can use a reusable DV-NIZK argument system, a KDM-CPA secure PKE scheme can easily be transformed into a KDM-CCA secure one by the Naor-Yung paradigm [NY90]. In fact, if the goal is to achieve an IND-CCA secure PKE scheme, then it is possible to replace a CRS-NIZK argument system in the Naor-Yung paradigm with a reusable DV-NIZK argument system. Furthermore, Camenisch, Chandran, and Shoup [CCS09] showed that (a slight variant of) the Naor-Yung paradigm with a CRS-NIZK argument system can be used to transform a KDM-CPA secure PKE scheme into a KDM-CCA secure one. Unfortunately, however, things are not so easy if we aim at achieving KDM-CCA security using a reusable DV-NIZK argument system via the Naor-Yung paradigm (or its existing variants). The main cause of difficulty is that if we apply the standard Naor-Yung paradigm using a DV-NIZK argument system, the secret verification key of the DV-NIZK argument system is included in the secret key of the resulting scheme, and circularity involving a DV-NIZK argument system occurs in the KDM-CCA security game. Our main technical contribution is circumventing this difficulty. We will detail the difficulty as well as our techniques in Section 2.

KDM-CCA Security in the Multi-User Setting Based on New Assumptions. Although our main focus in this work is on showing that KDM-CPA security and KDM-CCA security are equivalent, through the above results, we obtain the *first* KDM-CCA secure PKE schemes based on the computational Diffie-Hellman (CDH) assumption and the LPN assumption, since KDM-CPA secure PKE schemes can be constructed under these assumptions [BLSV18, Döt15, DGHM18]. These schemes satisfy only KDM-CCA security in the single user setting, since so does our generic construction, as noted earlier.

We then show how to extend our generic construction and achieve a PKE scheme satisfying KDM-CCA security in the multi-user setting under the CDH and LPN assumptions. This is done by requiring the underlying SKE scheme in our generic construction to satisfy a variant of *KDM security against related-key attacks (RKA-KDM security)* [App13], instead of one-time KDM security. (We also require a mild property that a secret key is a uniformly distributed random string.) An SKE scheme satisfying our definition of RKA-KDM security can be constructed based on the (polynomial hardness of) constant-noise LPN assumption [App13]. Moreover, we show how to construct an SKE scheme satisfying our RKA-KDM security notion based on hash encryption [DGHM18, BLSV18], which in turn can be based on the CDH assumption. This construction is an extension of a KDM-CPA secure PKE scheme based on batch encryption proposed by Brakerski, Lombardi, Segev, and Vaikuntanathan [BLSV18].

1.3 Related Work

Generic Constructions for KDM-CCA Secure PKE. To the best of our knowledge, the only existing generic methods for constructing KDM-CCA secure PKE, are the works by Camenisch, Chandran, and Shoup [CCS09], by Galindo, Herrantz, and Villar [GHV12], and by Kitagawa and Tanaka [KT18a]. Camenisch et al. [CCS09] showed how to construct a KDM-CCA secure PKE scheme from a KDM-CPA secure PKE scheme, an IND-CCA secure PKE scheme, and a CRS-NIZK proof (or argument) system. (We will touch it in Section 2.) Galindo et al. [GHV12] showed how to construct a KDM-CCA secure PKE scheme from an identity-based encryption scheme which satisfies so-called master-key-dependent message security, via the transformation by Canetti, Halevi, and Katz [CHK04]. However, the only known instantiation of Galindo et al.'s method can achieve security against adversaries that make an a-priori bounded number of master-key-KDM-encryption queries, which is translated to KDM-CCA security against adversaries that make an a-priori bounded number of KDM-encryption queries. Kitagawa and Tanaka [KT18a] showed how to construct a KDM-CCA secure PKE scheme based on a hash proof system [CS02] satisfying some homomorphic property. It is not obvious how to modify the methods of [GHV12, KT18a] to achieve a generic construction of a KDM-CCA secure PKE scheme starting from a KDM-CPA secure one.

Generic Constructions for IND-CCA Secure PKE. Here, we review the works that showed how to construct IND-CCA secure PKE schemes from an IND-CPA secure PKE scheme (or an equally fundamental primitive of a trapdoor function (TDF)) by assuming some additional structural/security properties on it and/or using some additional building blocks.

Dolev, Dwork, and Naor [DDN91] were the first to show the construction of an IND-CCA secure PKE scheme, from an IND-CPA secure scheme and a CRS-NIZK proof (or argument) system, based on the construction by Naor and Yung [NY90] that achieves weaker non-adaptive CCA (IND-CCA1) security.

Canetti, Halevi, and Katz [CHK04] showed how to transform an identity-based encryption scheme into an IND-CCA secure PKE scheme. Kiltz [Kil06] showed that the transformation is applicable to a weaker primitive of tag-based encryption.

Peikert and Waters [PW08] showed how to construct an IND-CCA secure PKE scheme from a lossy TDF. Subsequent works showed that TDFs with weaker security/functionality properties are sufficient for obtaining IND-CCA secure PKE schemes [RS09, KMO10, Wee10, YYHK16]. Hemenway and Ostrovsky [HO13] showed that one can construct a lossy TDF (and hence, an IND-CCA secure PKE scheme via [PW08]) from a lossy encryption scheme [BHY09] which can encrypt a message longer than an encryption-randomness.

Matsuda and Hanaoka [MH14a] showed how to construct an IND-CCA secure PKE scheme by using an IND-CPA secure PKE scheme and point obfuscation [Can97], and they [MH14b] showed another construction from an IND-CPA secure PKE scheme and a hash family satisfying one of universal computational extractors (UCE) assumptions [BHK13]. Dachman-Soled [Dac14] and Matsuda and Hanaoka [MH16] showed how to construct an IND-CCA secure PKE scheme from a PKE scheme which satisfies (weak) simulatability and the (standard model) plaintext awareness under the multiple keys setting.

Matsuda and Hanaoka [MH15] also showed that an IND-CCA secure PKE scheme can be built from the combination of a sender non-committing encryption scheme and a one-time KDM secure SKE scheme with respect to a-priori bounded size circuits. Hajiabadi and Kapron [HK15] showed how to construct an IND-CCA secure PKE scheme, from a 1-bit PKE scheme that satisfies circular security and has the structural property called reproducibility.

Very recently, Koppula and Waters [KW18] showed how to construct an IND-CCA secure

PKE scheme based on the combination of an IND-CPA secure PKE scheme and a hinting PRG. Building on [KW18], Kitagawa, Matsuda, and Tanaka [KMT19] showed how to construct an IND-CCA secure PKE scheme based on an IND-CPA secure PKE scheme and a one-time KDM secure SKE scheme with respect to projection functions, both of which are implied by a KDM-CPA secure PKE scheme with respect to projection functions.

1.4 Paper Organization

In Section 2, we give an overview of our techniques. In Section 3, we review definitions of cryptographic primitives. In Section 4 (and Appendix B), we explain how to construct a reusable DV-NIZK argument system from the combination of IND-CPA secure PKE and one-time KDM secure SKE with respect to projection functions. In Section 5, we present our main technical result: a CPA-to-CCA transformation for KDM security. In Section 6, we show that the construction given in Section 5 also achieves KDM-CCA security in the multi-user setting, if the building block SKE scheme additionally satisfies what we call passive RKA-KDM security whose formal definition is given in Section 3. In Section 7, we present a passively RKA-KDM secure SKE scheme from a hash encryption scheme. Finally, in Section 8, we summarize our results.

2 Technical Overview

In this section, we provide a technical overview of our main results. As mentioned in the introduction and will be detailed in Section 4, we can observe from the previous results [LQR+19a, KMT19] that a reusable DV-NIZK argument system can be constructed based on the combination of an IND-CPA secure PKE scheme and a one-time KDM secure SKE scheme. Thus, in this overview, we mainly focus on the generic construction of a PKE scheme that is KDM-CCA secure in the single user setting using a reusable DV-NIZK argument system. (From here on, we drop "reusable".) We also briefly explain how to extend it into the multi-user setting by using RKA-KDM secure SKE. We start with why we cannot achieve such a generic construction by using the standard Naor-Yung paradigm [NY90].

2.1 Naor-Yung Paradigm with DV-NIZK Fails for KDM

Camenisch, Chandran, and Shoup [CCS09] showed that the Naor-Yung paradigm with a CRS-NIZK argument system goes through for KDM security. We first review their construction, and then explain the problems that arise when replacing the underlying CRS-NIZK argument system with a DV-NIZK argument system.

KDM-CCA PKE by Camenisch et al. [CCS09]. The construction uses a KDM-CPA secure PKE scheme PKE, an IND-CCA secure PKE scheme PKE', and a CRS-NIZK argument system NIZK.[2] Using these building blocks, we construct $\mathsf{PKE}_{\mathsf{NY}}$ as follows. A public key of $\mathsf{PKE}_{\mathsf{NY}}$ consists of $(\mathsf{pk}, \mathsf{pk}_{\mathsf{cca}}, \mathsf{crs})$, where pk and $\mathsf{pk}_{\mathsf{cca}}$ are public keys of PKE and PKE', respectively, and crs is a CRS of NIZK. The corresponding secret key is sk corresponding to pk. The secret key $\mathsf{sk}_{\mathsf{cca}}$ corresponding to $\mathsf{pk}_{\mathsf{cca}}$ is discarded and used only in the security proof. When encrypting a message m, $\mathsf{PKE}_{\mathsf{NY}}$ generates a ciphertext of the form

$$\Big(\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(m),\ \mathsf{ct}_{\mathsf{cca}} = \mathsf{Enc}'_{\mathsf{pk}_{\mathsf{cca}}}(m),\ \pi \Big),$$

[2] In their actual construction, a one-time signature scheme is also used. We ignore it in this overview for simplicity, since the problem we explain below is unrelated to it.

where Enc and Enc' denote the encryption algorithms of PKE and PKE', respectively, and π is a proof of NIZK proving that ct and $\mathsf{ct}_{\mathsf{cca}}$ encrypt the same message, generated by using m and random coins used to generate ct and $\mathsf{ct}_{\mathsf{cca}}$ as a witness. When decrypting the ciphertext, it first checks whether the proof π is accepted or not. If π is accepted, it decrypts ct by using sk, and recovers m.

Camenisch et al. showed that $\mathsf{PKE}_{\mathsf{NY}}$ is KDM-CCA secure for a function class $\mathcal{F}$ with respect to which the underlying PKE scheme PKE satisfies KDM-CPA security.[3]

Circularity Involving DV-NIZK. We now explain why the above construction technique by Camenisch et al. does not work if we use a DV-NIZK argument system instead of a CRS-NIZK argument system.

If we use a DV-NIZK argument system DVNIZK instead of NIZK as a building block of $\mathsf{PKE}_{\mathsf{NY}}$, then we need a secret key $\mathsf{sk}_{\mathsf{dv}}$ of DVNIZK to verify a proof contained in a ciphertext when decrypting the ciphertext. Thus, we have to include $\mathsf{sk}_{\mathsf{dv}}$ into the secret key of $\mathsf{PKE}_{\mathsf{NY}}$.

In this case, an encryption of a message of the form $f(\mathsf{sk}\|\mathsf{sk}_{\mathsf{dv}})$ is given to an adversary in the KDM-CCA security game, where f is a function chosen by the adversary as a KDM-encryption query. Then, there is a circularity problem involving not only encryption schemes but also DVNIZK, since *when encrypting a message* $f(\mathsf{sk}\|\mathsf{sk}_{\mathsf{dv}})$*, a proof of* DVNIZK *is generated to guarantee that encryptions of its own secret key* $\mathsf{sk}_{\mathsf{dv}}$ *are well-formed*. Even if such a circularity exists, we can use the zero-knowledge property of DVNIZK in the security proof since a reduction algorithm attacking the zero-knowledge property is given a secret verification key $\mathsf{sk}_{\mathsf{dv}}$ and thus can handle such a circularity. However, we cannot use its soundness property in the security proof unless we solve the circularity, because a secret verification key $\mathsf{sk}_{\mathsf{dv}}$ is not directly given to an adversary attacking the soundness of DVNIZK.

Due to this circularity problem involving a DV-NIZK argument system, it seems difficult to achieve a KDM-CCA secure PKE scheme using a DV-NIZK variant of the Naor-Yung paradigm.

2.2 How to Solve the Circularity Problem Involving DV-NIZK?

The circularity problem involving a DV-NIZK argument system of $\mathsf{PKE}_{\mathsf{NY}}$ occurs because in the security game, a message depending on $\mathsf{sk}_{\mathsf{dv}}$ is encrypted by encryption schemes the validity of whose ciphertexts is proved by the DV-NIZK argument system. In order to solve this circularity problem, we have to design a scheme so that it has an *indirection* that a message is not directly encrypted by encryption schemes related to a DV-NIZK argument system.

The most standard way to add such an indirection to encryption schemes would be to use the hybrid encryption methodology. However, it is difficult to use the hybrid encryption methodology to construct a KDM-CCA secure scheme, since it leads to a dead-lock in the sense that the key encapsulation mechanism and data encapsulation mechanism could encrypt each other's secret key in the presence of key-dependent messages.

Thus, we use a different technique. We use a *garbling scheme* [Yao86] to realize the indirection that a message is not directly encrypted by encryption schemes related to a DV-NIZK argument system.[4] Concretely, when encrypting a message m, we first garble a circuit into which m is hardwired. Then, we encrypt each of the labels generated together with the garbled circuit by a PKE scheme, and then generate a proof proving that the encryptions of the labels are well-formed by using a DV-NIZK argument system.

[3] We note that in this construction, NIZK need not satisfy the simulation soundness property [Sah99], and we can complete the proof based on the ordinary soundness (and zero-knowledge) property of NIZK.

[4] The following explanations assume that the reader is familiar with a garbling scheme. See Definition 7 in Section 3 for its formal definition.

In order to realize the above idea using a garbling scheme, we use a one-time KDM secure SKE scheme at the key generation to encrypt (and add to a public key) secret key components of the building block PKE schemes. With the help of the one-time KDM secure SKE scheme, a garbling scheme makes it possible to simulate an encryption of the secret key without directly using the secret key itself, and we can prove the (multi-time) KDM security of the resulting scheme, which has the indirection.

Below, we first show the KDM-CPA variant of our construction without using a DV-NIZK argument system. Then, we show how to extend it into a KDM-CCA secure one.

2.3 KDM-CPA Variant of Our Construction

In the following, we show how to construct a KDM-CPA secure PKE scheme $\mathsf{PKE}^{*}_{\mathsf{kdm}}$ from a garbling scheme, a one-time KDM secure SKE scheme SKE, and IND-CPA secure PKE schemes PKE and PKE'.

Construction Using Garbled Circuits. The key pair $(\mathsf{PK}, \mathsf{SK})$ of $\mathsf{PKE}^{*}_{\mathsf{kdm}}$ is generated as follows. It first generates a secret key $s = (s_1, \ldots, s_{\ell_\mathsf{s}}) \in \{0,1\}^{\ell_\mathsf{s}}$ of SKE. Next, it generates a key pair $(\mathsf{pk}', \mathsf{sk}')$ of PKE' and $2\ell_\mathsf{s}$ key pairs $(\mathsf{pk}_{j,\alpha}, \mathsf{sk}_{j,\alpha})_{j\in[\ell_\mathsf{s}],\alpha\in\{0,1\}}$ of PKE. Then, it encrypts $\ell_\mathsf{s}+1$ secret keys sk' and $(\mathsf{sk}_{j,s_j})_{j\in[\ell_\mathsf{s}]}$ into $\mathsf{ct}_{\mathsf{ske}}$ by SKE under the key s. The public-key PK consists of $2\ell_\mathsf{s}+1$ public keys pk' and $(\mathsf{pk}_{j,\alpha})_{j\in[\ell_\mathsf{s}],\alpha\in\{0,1\}}$, and $\mathsf{ct}_{\mathsf{ske}}$. The corresponding secret key SK is just s. Namely, PK and SK are of the form

$$\mathsf{PK} = \Big(\ (\mathsf{pk}_{j,\alpha})_{j\in[\ell_\mathsf{s}],\alpha\in\{0,1\}},\ \mathsf{pk}',\ \mathsf{ct}_{\mathsf{ske}} = \mathsf{E}_s(\mathsf{sk}', (\mathsf{sk}_{j,s_j})_{j\in[\ell_\mathsf{s}]}) \ \Big) \quad \text{and} \quad \mathsf{SK} = s,$$

respectively, where $\mathsf{E}_s(\cdot)$ denotes the encryption algorithm of SKE using the key s.

When encrypting a message m under PK, $\mathsf{PKE}^{*}_{\mathsf{kdm}}$ first garbles a constant circuit Q that has m hardwired and outputs it for any input of length ℓ_s.[5] This results in a single garbled circuit $\widetilde{\mathsf{Q}}$ and $2\ell_\mathsf{s}$ labels $(\mathsf{lab}_{j,\alpha})_{j\in[\ell_\mathsf{s}],\alpha\in\{0,1\}}$. Then, the encryption algorithm encrypts "0-labels" $\mathsf{lab}_{j,0}$ into $\mathsf{ct}_{j,\alpha}$ by $\mathsf{pk}_{j,\alpha}$ for every $j \in [\ell_\mathsf{s}]$ and $\alpha \in \{0,1\}$. We finally encrypt $\widetilde{\mathsf{Q}}$ and those encrypted labels $(\mathsf{ct}_{j,\alpha})_{j,\alpha}$ using pk'. The resulting ciphertext CT is of the form

$$\mathsf{CT} = \mathsf{Enc}'_{\mathsf{pk}'}\Big(\widetilde{\mathsf{Q}},\ (\mathsf{ct}_{j,0} = \mathsf{Enc}_{\mathsf{pk}_{j,0}}(\mathsf{lab}_{j,0}),\ \mathsf{ct}_{j,1} = \mathsf{Enc}_{\mathsf{pk}_{j,1}}(\mathsf{lab}_{j,0}))_{j\in[\ell_\mathsf{s}]}\Big),$$

where Enc and Enc' are the encryption algorithms of PKE and PKE', respectively.

When decrypting the ciphertext CT using the secret key $\mathsf{SK} = s$, we first retrieve the secret keys sk' and $(\mathsf{sk}_{j,s_j})_{j\in[\ell_\mathsf{s}]}$ from $\mathsf{ct}_{\mathsf{ske}}$ contained in PK. Then, using sk', we recover $\widetilde{\mathsf{Q}}$ and $(\mathsf{ct}_{j,\alpha})_{j\in[\ell_\mathsf{s}],\alpha\in\{0,1\}}$. Moreover, we recover the "0-label" $\mathsf{lab}_{j,0}$ from ct_{j,s_j} using sk_{j,s_j} for every $j \in [\ell_\mathsf{s}]$. Finally, we evaluate the recovered garbled circuit $\widetilde{\mathsf{Q}}$ with these ℓ_s "0-labels" by the evaluation algorithm of the garbling scheme. This results in m, since given 0^{ℓ_s}, Q outputs m.

Overview of the Security Proof of $\mathsf{PKE}^{*}_{\mathsf{kdm}}$. We explain how we prove the KDM-CPA security in the single user setting of $\mathsf{PKE}^{*}_{\mathsf{kdm}}$. Specifically, we explain that no adversary $\mathcal{A}$ can guess the challenge bit b with probability significantly greater than $1/2$ given an encryption of $f_b(\mathsf{SK}) = f_b(s)$, when $\mathcal{A}$ queries two functions (f_0, f_1) as a KDM-encryption query.[6]

[5] In the actual construction, we use a garbled circuit and labels that are generated by the simulator of the garbling scheme, instead of those generated by garbling a constant circuit. This makes the security proof simpler. We ignore this treatment here for the simplicity of the explanation.

[6] Usually, KDM security requires that an encryption of $f(\mathsf{SK})$ be indistinguishable from that of some constant message such as $0^{|f(\cdot)|}$ instead of requiring encryptions of $f_0(\mathsf{SK})$ and $f_1(\mathsf{SK})$ be indistinguishable, where f, f_0, and f_1 are functions chosen by adversaries. However, these definitions are equivalent if a function class with respect to which we consider KDM security contains constant functions, which is the case in this paper.

In this construction, the secret keys of PKE corresponding to s, namely $(\mathsf{sk}_{j,s_j})_{j\in[\ell_\mathsf{s}]}$, are encrypted in ct_ske, but the rest of the secret keys $(\mathsf{sk}_{j,1\oplus s_j})_{j\in[\ell_\mathsf{s}]}$ are hidden from $\mathcal{A}$'s view. Thus, in the security proof, we can always use the IND-CPA security of PKE under the public keys $(\mathsf{pk}_{j,1\oplus s_j})_{j\in[\ell_\mathsf{s}]}$. By combining the IND-CPA security of PKE under these keys with the security of the garbling scheme, we can change the security game so that the encryption of $f_b(s)$ given to $\mathcal{A}$ can be simulated without using s, without being noticed by $\mathcal{A}$. Concretely, in the modified security game, an encryption of $f_b(s)$ is generated as follows. We first generate $\widetilde{\mathsf{Q}}$ and $(\mathsf{lab}_{j,\alpha})_{j\in[\ell_\mathsf{s}],\alpha\in\{0,1\}}$ by garbling a circuit computing f_b, instead of a constant circuit Q in which $f_b(s)$ is hardwired. Then, we encrypt $\mathsf{lab}_{j,\alpha}$ into $\mathsf{ct}_{j,\alpha}$ by $\mathsf{pk}_{j,\alpha}$ for every $j\in[\ell_\mathsf{s}]$ and $\alpha\in\{0,1\}$. Finally, we encrypt $\widetilde{\mathsf{Q}}$ and those encrypted labels $(\mathsf{ct}_{j,\alpha})_{j,\alpha}$ using pk', and obtain $\mathsf{CT}=\mathsf{Enc}_{\mathsf{pk}'}(\widetilde{\mathsf{Q}},(\mathsf{ct}_{j,0},\mathsf{ct}_{j,1})_{j\in[\ell_\mathsf{s}]})$. We see that we now do not need s to generate CT. The explanation so far in fact works even when $\mathcal{A}$ makes multiple KDM-encryption queries.

After the above change, a ciphertext CT given to $\mathcal{A}$ does not have any information of s, and thus we can use the one-time KDM security of SKE. Although the message $(\mathsf{sk}',(\mathsf{sk}_{j,s_j})_{j\in[\ell_\mathsf{s}]})$ encrypted in ct_ske depends on the secret key s, by relying on the one-time KDM security of SKE, we can further change the security game so that ct_ske is generated as an encryption of some constant message such as the all-zero string. Then, since sk' is now hidden from $\mathcal{A}$'s view, we can argue that $\mathcal{A}$'s advantage in the final game is essentially $1/2$ based on the IND-CPA security of PKE'. This completes the proof for the KDM-CPA security of $\mathsf{PKE}^*_\mathsf{kdm}$.

Features of $\mathsf{PKE}^*_\mathsf{kdm}$. This KDM-CPA secure construction $\mathsf{PKE}^*_\mathsf{kdm}$ has some nice properties. First, all of the building blocks are implied by KDM-CPA secure PKE. (Recall that a garbling scheme can be realized from one-way functions [Yao86].) Moreover, through this construction, we can transform a one-time KDM-CPA secure scheme into a (multi-time) KDM-CPA secure PKE scheme. Also, the resulting scheme satisfies KDM-CPA security with respect to all functions computable by circuits of a-priori bounded size even though the underlying KDM-CPA secure scheme needs to satisfy a much weaker form of KDM-CPA security. Concretely, the underlying scheme needs to be only KDM-CPA secure with respect to projection functions, since the encrypted message $(\mathsf{sk}',(\mathsf{sk}_{j,s_j})_{j\in[\ell_\mathsf{s}]})$ can be seen as an output of a function $g(x_1,\ldots,x_{\ell_\mathsf{s}})=(\mathsf{sk}',(\mathsf{sk}_{j,x_j})_{j\in[\ell_\mathsf{s}]})$, which can be described as a projection function of an input $x=(x_1,\ldots,x_{\ell_\mathsf{s}})\in\{0,1\}^{\ell_\mathsf{s}}$ that has $(\mathsf{sk}',(\mathsf{sk}_{j,\alpha})_{j\in[\ell_\mathsf{s}],\alpha\in\{0,1\}})$ hardwired. From these facts, in the single user setting, the construction $\mathsf{PKE}^*_\mathsf{kdm}$ in fact improves the previous amplification methods for KDM-CPA secure schemes [App11, DGHM18, KT18b]. In addition, most importantly, $\mathsf{PKE}^*_\mathsf{kdm}$ can be easily extended into a KDM-CCA secure one by using a DV-NIZK argument system.

2.4 KDM-CCA Secure PKE Using DV-NIZK

We extend $\mathsf{PKE}^*_\mathsf{kdm}$ into a KDM-CCA secure PKE scheme $\mathsf{PKE}_\mathsf{kdm}$ by the following two steps.

First, we use a DV-NIZK argument system DVNIZK for proving that encrypted labels are well-formed. Concretely, we use it in the following manner. When generating a key pair $(\mathsf{PK},\mathsf{SK})$ of $\mathsf{PKE}_\mathsf{kdm}$, we additionally generates a key pair $(\mathsf{pk}_\mathsf{dv},\mathsf{sk}_\mathsf{dv})$ of DVNIZK, and add pk_dv to PK. Moreover, we encrypt sk_dv into ct_ske together with $(\mathsf{sk}',(\mathsf{sk}_{j,s_j})_{j\in[\ell_\mathsf{s}]})$ by using s. The secret key SK is still only $s=(s_1,\ldots,s_{\ell_\mathsf{s}})\in\{0,1\}^{\ell_\mathsf{s}}$. Namely, PK and SK are of the form

$$\mathsf{PK}=\Big(\ (\mathsf{pk}_{j,\alpha})_{j\in[\ell_\mathsf{s}],\alpha\in\{0,1\}},\ \mathsf{pk}',\ \mathsf{pk}_\mathsf{dv},\ \mathsf{ct}_\mathsf{ske}=\mathsf{E}_s(\mathsf{sk}',\mathsf{sk}_\mathsf{dv},(\mathsf{sk}_{j,s_j})_{j\in[\ell_\mathsf{s}]})\ \Big)\quad\text{and}\quad\mathsf{SK}=s,$$

respectively. When encrypting a message m, we first generate $\widetilde{\mathsf{Q}}$ and $(\mathsf{ct}_{j,0},\mathsf{ct}_{j,1})_{j\in[\ell_\mathsf{s}]}$ in the same way as $\mathsf{PKE}^*_\mathsf{kdm}$. Then, using pk_dv, we generate a proof π of DVNIZK proving that $\mathsf{ct}_{j,0}$

and $\mathsf{ct}_{j,1}$ encrypt the same message for every $j \in [\ell_{\mathsf{s}}]$, by using $\mathsf{lab}_{j,0}$ and random coins used to generate $\mathsf{ct}_{j,0}$ and $\mathsf{ct}_{j,1}$ as a witness.

Next, in order to make the entire part of the ciphertext non-malleable, we require that PKE' satisfy IND-CCA security instead of IND-CPA security, and encrypt $\widetilde{\mathsf{Q}}$, the encrypted labels $(\mathsf{ct}_{j,0}, \mathsf{ct}_{j,1})_{j\in[\ell_{\mathsf{s}}]}$, and the proof π, using pk' of PKE'. Therefore, the resulting ciphertext CT is of the form

$$\mathsf{CT} = \mathsf{Enc}'_{\mathsf{pk}'}\left(\widetilde{\mathsf{Q}},\ (\mathsf{ct}_{j,0} = \mathsf{Enc}_{\mathsf{pk}_{j,0}}(\mathsf{lab}_{j,0}),\ \mathsf{ct}_{j,1} = \mathsf{Enc}_{\mathsf{pk}_{j,1}}(\mathsf{lab}_{j,0}))_{j\in[\ell_{\mathsf{s}}]},\ \pi \right).$$

We perform the decryption of this ciphertext in the same way as before, except that we additionally check whether π is accepted or not by using $\mathsf{sk}_{\mathsf{dv}}$ retrieved from $\mathsf{ct}_{\mathsf{ske}}$, and if it is not accepted, the ciphertext is rejected.

As mentioned earlier (and will be detailed in Section 4), by combining the techniques from the two recent results [LQR+19a, KMT19], a DV-NIZK argument system can be based on the same building blocks. Moreover, an IND-CCA secure PKE scheme can also be based on the same building blocks [KMT19]. Thus, similarly to $\mathsf{PKE}^*_{\mathsf{kdm}}$, all the building blocks of $\mathsf{PKE}_{\mathsf{kdm}}$ can be based on the combination of an IND-CPA secure PKE scheme and a one-time KDM secure SKE scheme, which are in turn both implied by a KDM-CPA secure PKE scheme.

Overview of the Security Proof of $\mathsf{PKE}_{\mathsf{kdm}}$. At first glance, the circularity involving DVNIZK occurs when encrypting a key-dependent message $f(\mathsf{SK}) = f(s) = \mathsf{sk}_{\mathsf{dv}}$ by $\mathsf{PKE}_{\mathsf{kdm}}$, where f is a function that, given s as input, retrieves $\mathsf{sk}_{\mathsf{dv}}$ from $\mathsf{ct}_{\mathsf{ske}}$ by using s and outputs $\mathsf{sk}_{\mathsf{dv}}$. This is because DVNIZK is used to generate a proof that proves $\mathsf{ct}_{j,0}$ and $\mathsf{ct}_{j,1}$ encrypt the same label, and the labels may contain some information of the key-dependent message $f(s)$ since it is generated by garbling a constant circuit Q into which $f(s)$ is hardwired. However, due to the indirection that $\mathsf{sk}_{\mathsf{dv}}$ is not encrypted by encryption schemes the validity of whose ciphertexts is proved by the DV-NIZK argument system, we can solve the circularity and prove the KDM-CCA security of $\mathsf{PKE}_{\mathsf{kdm}}$ by adding some modifications to the proof for the KDM-CPA security of $\mathsf{PKE}^*_{\mathsf{kdm}}$ explained in the previous section.

First of all, the zero-knowledge property of DVNIZK allows us to change the security game so that we use the simulator for the zero-knowledge property to generate the DV-NIZK key pair $(\mathsf{pk}_{\mathsf{dv}}, \mathsf{sk}_{\mathsf{dv}})$ at the key generation, and we use the simulator also for generating a fake proof π in a ciphertext when responding to KDM-encryption queries. Then, similarly to what we do in the proof for $\mathsf{PKE}^*_{\mathsf{kdm}}$, we can change the security game so that we do not need s for responding to KDM-encryption queries by using the security of the garbling scheme and the IND-CPA security of PKE under public keys $(\mathsf{pk}_{j,1\oplus s_j})_{j\in[\ell_{\mathsf{s}}]}$. However, differently from the proof for the KDM-CPA security of $\mathsf{PKE}^*_{\mathsf{kdm}}$, we cannot use the one-time KDM security of SKE immediately after this change. This is because we still need s for responding to decryption queries. More specifically, when responding to a decryption query, we have to decrypt the "s_j-side" ciphertext ct_{j,s_j} of PKE using sk_{j,s_j} for every $j \in [\ell_{\mathsf{s}}]$ to recover the labels of a garbled circuit.[7] Thus, before using the one-time KDM security of SKE, we change the security game so that we do not need s to respond to decryption queries by relying on the soundness of DVNIZK.

Concretely, we change the security game so that when responding to a decryption query CT, we *always* decrypt the "0-side" ciphertext $\mathsf{ct}_{j,0}$ of PKE using $\mathsf{sk}_{j,0}$ for every $j \in [\ell_{\mathsf{s}}]$. Although we cannot justify this change based solely on the soundness of DVNIZK, we can justify it by combining the soundness and zero-knowledge property of DVNIZK, the one-time KDM security

[7] Strictly speaking, we also use s to retrieve $(\mathsf{sk}', \mathsf{sk}_{\mathsf{dv}}, (\mathsf{sk}_{j,s_j})_{j\in[\ell_{\mathsf{s}}]})$ from $\mathsf{ct}_{\mathsf{ske}}$. However, we can omit this decryption process and use $(\mathsf{sk}', \mathsf{sk}_{\mathsf{dv}}, (\mathsf{sk}_{j,s_j})_{j\in[\ell_{\mathsf{s}}]})$ directly without changing the view of an adversary, and thus we ignore this issue here.

of SKE, and the IND-CCA security of PKE' using a *deferred analysis* technique. This technique of justifying changes for decryption queries using the deferred analysis originates in the context of expanding the message space of IND-CCA secure PKE schemes [HLW12], and was already shown to be useful in the context of KDM-CCA security [KMHT15, KT18a]. In fact, the indirection explained so far makes it possible to use the deferred analysis technique.

Once we change how decryption queries are answered in this way, we can complete the remaining part of the proof based on the one-time KDM security of SKE and the IND-CCA security of PKE' similarly to the proof for the KDM-CPA security of $\mathsf{PKE}^*_{\mathsf{kdm}}$.

For the formal description of our construction as well as the security proof, see Section 5.

Is It Essential to Encrypt $\mathsf{sk}_{\mathsf{dv}}$ into $\mathsf{ct}_{\mathsf{ske}}$? It is *not* essential to maintain $\mathsf{sk}_{\mathsf{dv}}$ (and sk') in the encrypted form $\mathsf{ct}_{\mathsf{ske}}$ by the key s and make SK consist only of s. In fact, we can consider a variant of $\mathsf{PKE}_{\mathsf{kdm}}$ such that we set $\mathsf{SK} := (s, \mathsf{sk}_{\mathsf{dv}}, \mathsf{sk}')$. In this case, we use $2 \cdot \ell_{\mathsf{SK}} = 2 \cdot (|s| + |\mathsf{sk}_{\mathsf{dv}}| + |\mathsf{sk}'|)$ key pairs of PKE, and we generate $\mathsf{ct}_{\mathsf{ske}}$ as an encryption of $(\mathsf{sk}_{j,\mathsf{SK}_j})_{j \in [\ell_{\mathsf{SK}}]}$ by s, where SK_j is the j-th bit of SK for every $j \in [\ell_{\mathsf{SK}}]$. Even if we adopt such a construction, we can realize an indirection that is sufficient to use the deferred analysis technique, and we can prove its KDM-CCA security similarly to the above.

The security proof for $\mathsf{PKE}_{\mathsf{kdm}}$ is simpler than that for the above variant. Moreover, as we will explain below, we need to encrypt $\mathsf{sk}_{\mathsf{dv}}$ and sk' and make $\mathsf{SK} = s$ when considering KDM-CCA security in the multi-user setting. For these reasons, we adopt the current construction of $\mathsf{PKE}_{\mathsf{kdm}}$.

2.5 Extension to KDM-CCA Security in the Multi-User Setting

We finally explain how to extend the above construction $\mathsf{PKE}_{\mathsf{kdm}}$ into a scheme that is KDM-CCA secure in the multi-user setting. In fact, we need not change the construction at all. The only difference is that we require a weak variant of *RKA-KDM security* [App13] for the underlying SKE scheme SKE, instead of one-time KDM security. We also require a mild property that a secret key is uniformly distributed over the secret key space $\{0,1\}^{\ell_{\mathsf{s}}}$.

Informally, an SKE scheme is said to be RKA-KDM secure if no adversary can guess the challenge bit b with probability significantly greater than $1/2$ given an encryption of $f_b(s)$ under the key $s \oplus \Delta \in \{0,1\}^{\ell_{\mathsf{s}}}$ when it queries two functions (f_0, f_1) and a key shift $\Delta \in \{0,1\}^{\ell_{\mathsf{s}}}$ as an RKA-KDM-encryption query. For our purpose, we need a much weaker form of RKA-KDM security where all key shifts are not chosen by an adversary, but generated uniformly at random in advance by the challenger. We call our RKA-KDM security *passive* RKA-KDM security. For its formal definition, see Definition 5 in Section 3.

In the security proof of the KDM-CCA security in the multi-user setting of $\mathsf{PKE}_{\mathsf{kdm}}$, there exist n key pairs of $\mathsf{PKE}_{\mathsf{kdm}}$ for some polynomial n of the security parameter. As a first step of the proof, we change the security game so that n secret keys $s^1, \ldots, s^n$ of $\mathsf{PKE}_{\mathsf{kdm}}$ are generated by first generating a single source key s and n key shifts $(\Delta^i)_{i \in [n]}$ and then setting $s^i := s \oplus \Delta^i$ for every $i \in [n]$. This does not at all change the distribution of the keys due to the requirement on SKE that a secret key is distributed uniformly in the secret key space $\{0,1\}^{\ell_{\mathsf{s}}}$. We next change the security game so that for every $i^* \in [n]$, an encryption of $f_b(s^1 \| \ldots \| s^n)$ under the i^*-th key can be simulated from f_b and n key shifts $(\Delta^i)_{i \in [n]}$ and *not* the source key s, where (i^*, f_0, f_1) is a KDM-encryption query made by an adversary. This is possible by garbling a circuit into which f_b, i^*, and $(\Delta^i)_{i \in [n]}$ are hardwired,[8] while we just directly garble f_b in the proof for the single user security. Then, we can complete the rest of the security proof in the

[8] To make this change possible, in the formal proof, we need to pad a circuit garbled in the encryption algorithm to some appropriate size depending on n.

same way as the proof of the single user security except that we use the (passive) RKA-KDM security instead of one-time KDM security. For the details of the proof, see Section 6.

Differently from the single user case, it is critical that $\mathsf{sk}_{\mathsf{dv}}$ and sk' are encrypted into $\mathsf{ct}_{\mathsf{ske}}$, and SK consists only of s. If SK is of the form $(s, \mathsf{sk}_{\mathsf{dv}}, \mathsf{sk}')$, it is not clear how we control the multiple secret keys even if SKE is RKA-KDM secure.

KDM-CCA Secure PKE from New Assumptions. An SKE scheme satisfying our definition of RKA-KDM security can be constructed based on the LPN assumption [App13]. Moreover, we show how to construct an SKE scheme satisfying our RKA-KDM security definition based on hash encryption [DGHM18, BLSV18] which in turn can be based on the CDH assumption. The construction is an extension of that of a KDM-CPA secure PKE scheme based on batch encryption proposed by Brakerski et al. [BLSV18]. For the details of the construction and its security proof, see Section 7.

In addition to RKA-KDM secure SKE schemes, all other building blocks of our construction can be obtained based on the LPN and CDH assumptions via KDM-CPA secure PKE schemes. Through our generic construction, we obtain the first PKE schemes that are KDM-CCA secure in the multi-user setting based on the LPN and CDH assumptions. Previously to our work, KDM-CCA secure PKE schemes even in the single user setting based on these assumptions were not known.

3 Preliminaries

In this section, we review basic notation and the definitions of cryptographic primitives used in the paper.

3.1 Notations

$\mathbb{N}$ denotes the set of natural numbers, and for $n \in \mathbb{N}$, we define $[n] := \{1, \ldots, n\}$. 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). λ always denotes a security parameter. PPT stands for *probabilistic polynomial time*. A function $f(\lambda)$ is said to be negligible if $f(\lambda)$ tends to 0 faster than λ^{-c} for every constant $c > 0$. We write $f(\lambda) = \mathsf{negl}(\lambda)$ to mean that $f(\lambda)$ is a negligible function.

3.2 Public-Key Encryption

Here, we review the definitions for public-key encryption (PKE).

Definition 1 (Public-Key Encryption) *A PKE scheme* PKE *is a three tuple* $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *of PPT algorithms.*

- KG *is the key generation algorithm that takes a security parameter* 1^λ *as input, and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$.
- Enc *is the encryption algorithm that takes a public key* pk *and a message* m *as input, and outputs a ciphertext* ct.

- Dec *is the (deterministic) decryption algorithm that takes a public key* pk*, a secret key* sk*, and a ciphertext* ct *as input, and outputs a message* m *which could be the special symbol* $\perp$ *indicating that* ct *is invalid.*

Correctness *We require* $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m$ *for all* $\lambda \in \mathbb{N}$*, all key pairs* $(\mathsf{pk}, \mathsf{sk})$ *output by* $\mathsf{KG}(1^\lambda)$*, and all messages* m*.*

Security Notions for PKE. Next, we review the definitions of key-dependent message security against chosen ciphertext attacks/chosen plaintext attacks (KDM-CPA/CCA security). Note that IND-CPA/CCA security are covered as their special cases.

Definition 2 (KDM-CCA/KDM-CPA Security) *Let* PKE *be a PKE scheme whose secret key and message spaces are* $\mathcal{SK}$ *and* $\mathcal{M}$*, respectively. Let* $n \in \mathbb{N}$*, and let* $\mathcal{F}$ *be a function family with domain* $\mathcal{SK}^n$ *and range* $\mathcal{M}$*. Consider the following* $\mathcal{F}$*-KDM*$^{(n)}$*-CCA game between a challenger and an adversary* $\mathcal{A}$*.*

1. *First, the challenger chooses a challenge bit* $b \xleftarrow{\mathsf{r}} \{0,1\}$*. Next, the challenger generates* n *key pairs* $\left(\mathsf{pk}^i, \mathsf{sk}^i\right) \leftarrow \mathsf{KG}(1^\lambda)\,(i \in [n])$*. Then, the challenger sets* $\mathbf{sk} := \left(\mathsf{sk}^1, \ldots, \mathsf{sk}^n\right)$ *and sends* $\left(\mathsf{pk}^1, \ldots, \mathsf{pk}^n\right)$ *to* $\mathcal{A}$*. Finally, the challenger prepares an empty list* L_{kdm}*.*
2. $\mathcal{A}$ *may adaptively make the following queries.*

   KDM-encryption queries: $\mathcal{A}$ *sends* $(j, f_0, f_1) \in [n] \times \mathcal{F}^2$ *to the challenger. The challenger returns* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}^j, f_b(\mathbf{sk}))$ *to* $\mathcal{A}$*. Finally, the challenger adds* (j, ct) *to* L_{kdm}*.*

   Decryption queries: $\mathcal{A}$ *sends* (j, ct) *to the challenger. If* $(j, \mathsf{ct}) \in L_{\mathsf{kdm}}$*, then the challenger returns* $\perp$ *to* $\mathcal{A}$*. Otherwise, the challenger returns* $m \leftarrow \mathsf{Dec}(\mathsf{pk}^j, \mathsf{sk}^j, \mathsf{ct})$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$*.*

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

$\mathcal{F}$-KDM$^{(n)}$-CPA security is defined similarly, using the $\mathcal{F}$-KDM$^{(n)}$-CPA game where an adversary $\mathcal{A}$ is not allowed to make decryption queries.

The above definition is slightly different from the standard definition where an adversary is required to distinguish encryptions of $f(\mathsf{sk}^1, \ldots, \mathsf{sk}^n)$ from encryptions of some fixed message. However, the two definitions are equivalent if the function class $\mathcal{F}$ contains a constant function, which is the case for the function families used in this paper (see below). This formalization is easier to work with for security proofs.

Function Families. In this paper, we will deal with the following function families for KDM security of PKE:

$\mathcal{P}$ **(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.

$\mathcal{B}_{\mathsf{size}}$ **(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 such that each member in $\mathcal{B}_{\mathsf{size}}$ can be described by a circuit of size size.

$\mathcal{C}$ **(Constant functions):** We denote by $\mathcal{C}$ the set of all constant functions. Note that $\mathcal{C}$-KDM-CCA (resp. $\mathcal{C}$-KDM-CPA) security is equivalent to IND-CCA (resp. IND-CPA) security.

3.3 Secret-Key Encryption

Here, we review the definitions for secret-key encryption (SKE).

Definition 3 (Secret-Key Encryption) *An SKE scheme* SKE *is a three tuple* $(\mathsf{K}, \mathsf{E}, \mathsf{D})$ *of PPT algorithms.*

- K *is the key generation algorithm that takes a security parameter* 1^λ *as input, and outputs a key* s*.*
- E *is the encryption algorithm that takes a secret key* s *and a message* m *as input, and outputs a ciphertext* ct*.*
- D *is the (deterministic) decryption algorithm that takes a secret key* s *and a ciphertext* ct *as input, and outputs a message* m *which could be the special symbol* $\perp$ *indicating that* ct *is invalid.*

Correctness *We require* $\mathsf{D}(s, \mathsf{E}(s, m)) = m$ *for all* $\lambda \in \mathbb{N}$*, all keys* s *output by* $\mathsf{K}(1^\lambda)$*, and all messages* m*.*

Security Notions for SKE. In this paper, we will deal with two types of security notions for SKE: *one-time KDM security* and *passive RKA-KDM security*. We review the definitions below.

One-time KDM security is a weak form of KDM-CPA security in which an adversary is allowed to make only a single KDM-encryption query.

Definition 4 (One-Time KDM Security) *Let* $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *be an SKE scheme whose key and message spaces are* $\mathcal{K}$ *and* $\mathcal{M}$*, respectively. Let* $\mathcal{F}$ *be a function family with domain* $\mathcal{K}$ *and range* $\mathcal{M}$*. Consider the following one-time* $\mathcal{F}$*-KDM game between a challenger and an adversary* $\mathcal{A}$*.*

1. *First, the challenger chooses a challenge bit* $b \xleftarrow{\mathsf{r}} \{0,1\}$*. Next, the challenger generates a secret key* $s \leftarrow \mathsf{K}(1^\lambda)$ *and sends* 1^λ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends a function* $f \in \mathcal{F}$ *as a single KDM-encryption query to the challenger. If* $b = 1$*, the challenger returns* $\mathsf{ct} \leftarrow \mathsf{E}(s, f(s))$ *to* $\mathcal{A}$*; Otherwise, the challenger returns* $\mathsf{ct} \leftarrow \mathsf{E}(s, 0^{|f(\cdot)|})$ *to* $\mathcal{A}$*. (Note that this step is done only once.)*
3. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$*.*

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

Remark 1 (On the Message Space of One-Time KDM Secure SKE) When talking about the one-time KDM security of an SKE scheme, the size of the message space is an important factor. As shown by Hofheinz and Unruh [HU08], SKE satisfying one-time KDM security with respect to *all* functions can be achieved *unconditionally* if its message space is sufficiently smaller than its secret key space.

Unlike ordinary IND-CPA secure encryption schemes, extending the message space of KDM secure encryption schemes is in general not easy. Fortunately, however, things are easy for $\mathcal{P}$-KDM security. We can extend the message space of a one-time $\mathcal{P}$-KDM secure SKE scheme as much as we want, if the size of the message space of the SKE scheme is already sufficiently large. Specifically, we can show that if there exists a one-time $\mathcal{P}$-KDM secure SKE scheme whose secret key and message spaces are $\{0,1\}^\ell$ and $\{0,1\}^\mu$, respectively, for some polynomials $\ell = \ell(\lambda)$ and $\mu = \mu(\lambda)$ satisfying $\mu = \Omega(\ell \cdot \lambda)$, then for any polynomial $\mu' = \mu'(\lambda)$, there also exists a one-time $\mathcal{P}$-KDM secure SKE scheme that can encrypt messages of length μ'.

To see this, we observe that the KDM-CPA secure construction $\mathsf{PKE}^*_{\mathsf{kdm}}$ that we described in Section 2.3, works also in the secret-key setting. Namely, if we replace the building block IND-CPA secure PKE schemes with IND-CPA secure SKE schemes, then the resulting SKE scheme[9] is (multi-time) $\mathcal{B}_{\mathsf{size}}$-KDM secure where $\mathsf{size} = \mathsf{size}(\lambda)$ is some polynomial that depends on the size of a constant circuit (in which a message is hardwired). In fact, we can make the message space of this construction arbitrarily large since by setting size appropriately, we can hardwire a message of arbitrary length into a circuit to be garbled without compromising the security. Moreover, we only need to assume that the underlying one-time $\mathcal{P}$-KDM secure SKE scheme can encrypt messages of length $\mu = \Omega(\ell \cdot \lambda)$ since it is only required to encrypt $\ell + 1$ secret keys of IND-CPA secure SKE schemes, each of which can be assumed to be λ-bit without loss of generality. This means that, using this construction, we can extend the message space of a one-time $\mathcal{P}$-KDM secure SKE scheme as much as we want if the scheme can already encrypt a message of length $\mu = \Omega(\ell \cdot \lambda)$.

Next, we give a formalization of passive RKA-KDM security, which is a weaker variant of RKA-KDM security formalized by Applebaum [App13]. Recall that the original RKA-KDM security of [App13] is a slightly stronger form of standard KDM-CPA security (albeit in the presence of a single challenge key) where we consider an adversary that is allowed to ask encryptions of key-dependent messages, encrypted under "related" keys. In this paper, we only consider "XOR by a constant" as related-key deriving functions, and hence give a definition specialized to this setting. On the other hand, however, we only need a weaker "passive" variant of RKA-KDM security where the security game is changed as follows: (1) not the adversary but the challenger randomly chooses the related-key deriving functions (i.e. constants for XORing in our setting), and (2) an adversary has to make its RKA-KDM-encryption queries in one shot.

Definition 5 (Passive RKA-KDM Security) *Let SKE be an SKE scheme whose key space is $\{0,1\}^\ell$ for some polynomial $\ell = \ell(\lambda)$ and whose message space is $\mathcal{M}$. Let $\mathcal{F}$ be a function family with domain $\{0,1\}^\ell$ and range $\mathcal{M}$. Let $n \in \mathbb{N}$ be an a-priori bounded polynomial. Consider the following passive $\mathcal{F}$-RKA-KDM$^{(n)}$ game between a challenger and an adversary $\mathcal{A}$.*

1. *First, the challenger chooses a challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$ and generates $s \leftarrow \mathsf{K}(\lambda)$ and $\Delta^i \xleftarrow{\mathsf{r}} \{0,1\}^\ell$ for every $i \in [n]$. Then, the challenger sends $(\Delta^i)_{i\in[n]}$ to $\mathcal{A}$.*
2. *$\mathcal{A}$ sends n functions $f^1, \ldots, f^n \in \mathcal{F}$ to the challenger. If $b = 1$, the challenger computes $\mathsf{ct}^i \leftarrow \mathsf{E}(s \oplus \Delta^i, f^i(s))$ for every $i \in [n]$. Otherwise, the challenger computes $\mathsf{ct}^i \leftarrow \mathsf{E}(s \oplus \Delta^i, 0^{|f^i(\cdot)|})$ for every $i \in [n]$. Finally, the challenger sends $\left(\mathsf{ct}^i\right)_{i\in[n]}$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

[9] If we are only interested in one-time KDM security of the resulting scheme, the SKE-ciphertext $\mathsf{ct}_{\mathsf{ske}}$ that is originally put in a public key of $\mathsf{PKE}^*_{\mathsf{kdm}}$ can be sent as part of a ciphertext.

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

3.4 Designated-Verifier Non-interactive Zero-Knowledge Arguments

Here, we review the definitions for (reusable) designated-verifier non-interactive zero-knowledge (DV-NIZK) argument systems.

Definition 6 (DV-NIZK) *Let* L *be an NP language associated with the corresponding NP relation* R*. A DV-NIZK argument system* DVNIZK *for* L *is a three tuple* $(\mathsf{DVKG}, \mathsf{P}, \mathsf{V})$ *of PPT algorithms.*[10]

- DVKG *is the key generation algorithm that takes a security parameter* 1^λ *as input, and outputs a public proving key* pk *and a secret verification key* sk*.*
- P *is the proving algorithm that takes a public proving key* pk*, a statement* x*, and a witness* w *as input, and outputs a proof* π*.*
- V *is the (deterministic) verification algorithm that takes a secret verification key* sk*, statement* x*, and a proof* π *as input, outputs either* accept *or* reject*.*

We require that DVNIZK *satisfy the three requirements: Correctness, (adaptive) soundness, and zero-knowledge. In particular, we consider a version of soundness which holds against adversaries that make multiple verification queries, and a version of zero-knowledge which holds against adversaries that make multiple challenge proving queries. A DV-NIZK argument system that satisfies these versions of soundness and zero-knowledge is called* reusable*.*

Formally, these requirements are defined as follows.

Correctness *We say that* DVNIZK *is correct if we have* $\mathsf{V}(\mathsf{sk}, \mathsf{P}(\mathsf{pk}, x, w)) = \mathsf{accept}$ *for all* $\lambda \in \mathbb{N}$*, all key pairs* $(\mathsf{pk}, \mathsf{sk})$ *output by* $\mathsf{DVKG}(1^\lambda)$*, and all valid statement/witness pairs* $(x, w) \in R$*.*

Soundness *Consider the following soundness game between a challenger and an adversary* $\mathcal{A}$*.*

1. *First, the challenger generates* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{DVKG}(1^\lambda)$ *and sends* pk *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *may adaptively make verification queries. When* $\mathcal{A}$ *makes a verification query* (x, π)*, the challenger responds with* $\mathsf{V}(\mathsf{sk}, x, \pi)$*.*
3. $\mathcal{A}$ *outputs* (x^*, π^*)*.*

We say that DVNIZK *is sound if for any PPT adversary* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{DVNIZK},\mathcal{A}}(\lambda) := \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{DVNIZK},\mathcal{A}}(\lambda) = \Pr[x^* \notin L \wedge \mathsf{V}(\mathsf{sk}, x^*, \pi^*) = \mathsf{accept}] = \mathsf{negl}(\lambda)$.

Zero-Knowledge *Let* $\mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2)$ *be a pair of PPT "simulator" algorithms whose syntax is as follows.*

- S_1 *takes a security parameter* 1^λ *as input, and outputs a fake public key* pk*, a fake secret key* sk*, and a trapdoor* td*.*
- S_2 *takes a trapdoor* td *and a statement* x *as input, and outputs a fake proof* π*.*

[10] Lombardi et al. [LQR+19a] adopted the syntax of a DV-NIZK argument system in which there is a setup algorithm that generates a CRS. This is necessary for considering zero-knowledge property against malicious verifiers that may generate its public proving key maliciously. Since we only consider the standard zero-knowledge property with honestly generated keys, we adopt a simpler syntax without a separate setup algorithm.

Consider the following zero-knowledge game between a challenger and an adversary $\mathcal{A}$.

1. *First, the challenger chooses the challenge bit $b \xleftarrow{r} \{0,1\}$. If $b = 1$, then the challenger generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{DVKG}(1^\lambda)$; Otherwise the challenger generates $(\mathsf{pk}, \mathsf{sk}, \mathsf{td}) \leftarrow \mathsf{S}_1(1^\lambda)$. Then, the challenger sends $(\mathsf{pk}, \mathsf{sk})$ to $\mathcal{A}$.*
2. *$\mathcal{A}$ may adaptively make proving queries. When $\mathcal{A}$ submits a proving query (x, w), if $(x, w) \notin R$, then the challenger returns $\perp$ to $\mathcal{A}$. Then, if $b = 1$, the challenger computes $\pi \leftarrow \mathsf{P}(\mathsf{pk}, x, w)$; Otherwise, the challenger computes $\pi \leftarrow \mathsf{S}_2(\mathsf{td}, x)$. Finally, the challenger returns π to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

We say that DVNIZK *is zero-knowledge if there exists a PPT simulator $\mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2)$ such that for all PPT adversaries $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{zk}}_{\mathsf{DVNIZK},\mathcal{A},\mathsf{S}}(\lambda) := 2 \cdot |\Pr[b = b'] - 1/2| = \mathsf{negl}(\lambda)$.*

3.5 Garbled Circuits

Here, we recall the definitions of a garbling scheme in the form we use in this paper. We can realize a garbling scheme for all efficiently computable circuits based on one-way functions [Yao86].

Definition 7 (Garbled Circuits) *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 garbling scheme* GC *is a three tuple* $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ *of PPT algorithms.*

- Garble *is the garbling algorithm that takes as input a security parameter 1^λ and a circuit $C \in \mathcal{C}_n$, where $n = n(\lambda)$ is a polynomial. Then, it outputs a garbled circuit $\widetilde{C}$ and $2n$ labels $(\mathsf{lab}_{j,\alpha})_{j\in[n],\alpha\in\{0,1\}}$. For simplicity and without loss of generality, we assume that the length of each $\mathsf{lab}_{j,\alpha}$ is λ.*
- Eval *is the evaluation algorithm that takes a garbled circuit $\widetilde{C}$ and n labels $(\mathsf{lab}_j)_{j\in[n]}$ as input, and outputs an evaluation result y.*
- Sim *is the simulator algorithm that takes a security parameter 1^λ, the size parameter* size *(where $\mathsf{size} = \mathsf{size}(\lambda)$ is a polynomial), and a string y as input, and outputs a simulated garbled circuit $\widetilde{C}$ and n simulated labels $(\mathsf{lab}_j)_{j\in[n]}$.*

For a garbling scheme, we require the following correctness and security properties.

Correctness *For all $\lambda, n \in \mathbb{N}$, all $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, and all $C \in \mathcal{C}_n$, we require that the following two equalities hold.*[11]

- $\mathsf{Eval}(\widetilde{C}, (\mathsf{lab}_{j,x_j})_{j\in[n]}) = C(x)$ *for all* $(\widetilde{C}, (\mathsf{lab}_{j,\alpha})_{j\in[n],\alpha\in\{0,1\}})$ *output by* $\mathsf{Garble}(1^\lambda, C)$.
- $\mathsf{Eval}(\widetilde{C}, (\mathsf{lab}_j)_{j\in[n]}) = C(x)$ *for all* $(\widetilde{C}, (\mathsf{lab}_j)_{j\in[n]})$ *output by* $\mathsf{Sim}(1^\lambda, |C|, C(x))$.

Security *Consider the following security game between a challenger and an adversary $\mathcal{A}$.*

1. *First, the challenger chooses a bit $b \xleftarrow{r} \{0,1\}$ and sends a security parameter 1^λ to $\mathcal{A}$.*

[11] Requiring correctness for the output of the simulator may be somewhat non-standard. However, it is satisfied by Yao's garbling scheme based on an IND-CPA secure SKE scheme.

2. *$\mathcal{A}$ sends a circuit $C \in \mathcal{C}_n$ and an input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ to the challenger. Then, if $b = 1$, the challenger computes $(\widetilde{C}, (\mathsf{lab}_{j,\alpha})_{j\in[n],\alpha\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)$ and returns $(\widetilde{C}, (\mathsf{lab}_{j,x_j})_{j\in[n]})$ to $\mathcal{A}$; Otherwise, the challenger returns $(\widetilde{C}, (\mathsf{lab}_j)_{j\in[n]}) \leftarrow \mathsf{Sim}(1^\lambda, |C|, C(x))$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

We say that GC is secure if for all PPT adversaries $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{gc}}_{\mathsf{GC},\mathcal{A},\mathsf{Sim}}(\lambda) := 2 \cdot |\Pr[b = b'] - 1/2| = \mathsf{negl}(\lambda)$.

4 DV-NIZK via KDM Security

In this section, we explain how to construct a reusable DV-NIZK argument system from the combination of an IND-CPA secure PKE scheme and a one-time $\mathcal{P}$-KDM secure SKE scheme. Specifically, we explain how the following statement can be derived.

Theorem 2 *Assume that there exist an IND-CPA secure PKE scheme and a one-time $\mathcal{P}$-KDM secure SKE scheme that can encrypt messages of length $\Omega(\ell \cdot \lambda)$, where $\ell = \ell(\lambda)$ is the secret key length of the SKE scheme. Then, there exists a reusable DV-NIZK argument system for all NP languages.*

As mentioned in the introduction, this almost immediately follows by combining the results and techniques from the recent works by Lombardi et al. [LQR+19a] and by Kitagawa et al. [KMT19]. To see this, we first briefly review Lombardi et al.'s work [LQR+19a].

Lombardi et al. showed how to construct a reusable DV-NIZK argument system for all NP languages from the combination of an IND-CPA secure PKE scheme and a hinting PRG introduced by Koppula and Waters [KW18]. The main intermediate technical tool for their construction is what they call *attribute-based secure function evaluation (AB-SFE)*, which can be seen as a generalization (and simplification) of a single-key attribute-based encryption (ABE) scheme (i.e., an ABE scheme secure in the presence of a single secret key). Lombardi et al. formalized two kinds of security notions for AB-SFE: *key-hiding* and *message-hiding*, each notion with strong and weak variants, resulting in total four security notions.[12] Using the notion of AB-SFE, they achieved their result in a modular manner by showing the following steps:

- **(DV-NIZK-from-AB-SFE:)** A reusable DV-NIZK argument system can be constructed from an AB-SFE scheme satisfying *strong key-hiding* and weak message-hiding.
- **(Key-Hiding Enhancement:)** An AB-SFE scheme satisfying strong key-hiding and weak message-hiding can be constructed from an AB-SFE scheme satisfying *weak key-hiding* and weak message-hiding, by additionally assuming a hinting PRG. This step directly uses the CPA-to-CCA security transformation for ABE using a hinting PRG by Koppula and Waters [KW18].
- **(AB-SFE-from-PKE:)** An AB-SFE scheme satisfying weak key-hiding and weak message-hiding can be constructed from an IND-CPA secure PKE scheme.

On the other hand, Kitagawa et al. [KMT19] showed that an IND-CCA secure PKE scheme can be constructed from the combination of an IND-CPA secure PKE scheme and a one-time

[12] We recall the formal definitions for AB-SFE in Appendix A.2. Among the four security notions for AB-SFE, strong message-hiding is not directly relevant to our result on KDM-CCA secure PKE, and we do not mention the results related to it.

$\mathcal{P}$-KDM secure SKE scheme which can encrypt messages of length $\Omega(\ell \cdot \lambda)$, where ℓ denotes the secret key length of the SKE scheme, based on the Koppula-Waters construction [KW18].

Kitagawa et al.'s result can be understood as showing a technique for replacing a hinting PRG in the Koppula-Waters construction (and its variants) with a one-time $\mathcal{P}$-KDM secure SKE scheme. Hence, we can apply Kitagawa et al.'s technique to the "key-hiding enhancement" step of Lombardi et al. to replace the hinting PRG with a one-time $\mathcal{P}$-KDM secure SKE scheme. This can be formally stated as follows.

Theorem 3 (Key-Hiding Enhancement via KDM Security) *Assume that there exists an AB-SFE scheme that satisfies weak key-hiding and weak message-hiding, and a one-time $\mathcal{P}$-KDM secure SKE scheme that can encrypt messages of length $\Omega(\ell \cdot \lambda)$, where $\ell = \ell(\lambda)$ is the secret key length of the SKE scheme. Then, there exists an AB-SFE scheme that satisfies strong key-hiding and weak message-hiding.*

Then, Theorem 2 follows from the combination of the "DV-NIZK-from-AB-SFE" and "AB-SFE-from-PKE" steps of Lombardi et al. [LQR+19a] and Theorem 3.

For completeness, we give the formal proof of Theorem 3 in Appendix B.

5 Generic Construction of KDM-CCA Secure PKE

In this section, we show our main result: a CPA-to-CCA transformation for KDM security.

More specifically, we show how to construct a PKE scheme that is KDM-CCA secure with respect to circuits whose size is bounded by an a-priori determined polynomial $\mathsf{size} = \mathsf{size}(\lambda)$ and in the single user setting (i.e. $\mathcal{B}_{\mathsf{size}}$-KDM$^{(1)}$-CCA), from the combination of the five building block primitives: (1) an IND-CPA secure PKE scheme, (2) an IND-CCA secure PKE scheme, (3) a reusable DV-NIZK argument system for an NP language, (4) a garbling scheme, and (5) a one-time $\mathcal{P}$-KDM secure SKE scheme.

We have seen in Section 4 that a reusable DV-NIZK argument system can be constructed from the combination of an IND-CPA secure PKE scheme and a one-time $\mathcal{P}$-KDM secure SKE scheme. Furthermore, the recent work by Kitagawa et al. [KMT19] showed that an IND-CCA secure PKE scheme can also be constructed from the same building blocks. Moreover, a garbling scheme can be constructed only from a one-way function [Yao86], which is in turn implied by an IND-CPA secure PKE or a one-time $\mathcal{P}$-KDM secure SKE scheme. Hence, our result in this section implies that a $\mathcal{B}_{\mathsf{size}}$-KDM$^{(1)}$-CCA secure PKE scheme can be constructed only from an IND-CPA secure PKE scheme and a one-time $\mathcal{P}$-KDM secure SKE scheme.

Looking ahead, in the next section, we will show that the same construction can be shown to be secure in the n-user setting (i.e. $\mathcal{B}_{\mathsf{size}}$-KDM$^{(n)}$-CCA secure) if we additionally require the SKE scheme to be passively $\mathcal{P}$-RKA-KDM$^{(n)}$ secure.

Construction. Let $\ell_{\mathsf{m}} = \ell_{\mathsf{m}}(\lambda)$ be a polynomial that denotes the length of messages to be encrypted by our constructed PKE scheme. Let $\mathsf{size} = \mathsf{size}(\lambda)$ be a polynomial and let $n \in \mathbb{N}$ be the number of users for which we wish to achieve $\mathcal{B}_{\mathsf{size}}$-KDM$^{(n)}$-CCA security.[13]

We use the following building blocks.

- Let $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme whose message space is $\{0,1\}^\lambda$. We denote the randomness space of Enc by $\mathcal{R}$, and the secret key length by $\ell_{\mathsf{sk}} = \ell_{\mathsf{sk}}(\lambda)$.

[13] As noted earlier, in this section we aim at achieving the security for $n = 1$, and in the next section we will consider more general $n \geq 1$.

- Let $\mathsf{PKE}' = (\mathsf{KG}_{\mathsf{cca}}, \mathsf{Enc}_{\mathsf{cca}}, \mathsf{Dec}_{\mathsf{cca}})$ be a PKE scheme whose message space is $\{0,1\}^*$. We denote its secret key length by $\ell'_{\mathsf{sk}} = \ell'_{\mathsf{sk}}(\lambda)$.
- Let $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be an SKE scheme whose plaintext space is $\{0,1\}^{\mu}$ for a polynomial $\mu = \mu(\lambda)$ to be determined below and whose secret key space is $\{0,1\}^{\ell_s}$ for some polynomial $\ell_s = \ell_s(\lambda)$.
- Let $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ be a garbling scheme.
- Let $\mathsf{DVNIZK} = (\mathsf{DVKG}, \mathsf{P}, \mathsf{V})$ be a DV-NIZK argument system for the following NP language[14]

$$L = \left\{ (\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j\in[\ell_s],\alpha\in\{0,1\}} \;\middle|\; \exists(\mathsf{lab}_j, r_{j,0}, r_{j,1})_{j\in[\ell_s]} \text{ s.t. } \begin{array}{l} \forall(j,\alpha) \in [\ell_s]\times\{0,1\} : \\ \mathsf{ct}_{j,\alpha} = \mathsf{Enc}(\mathsf{pk}_{j,\alpha}, \mathsf{lab}_j; r_{j,\alpha}) \end{array} \right\}.$$

  We denote the verification key length of DVNIZK by $\ell_{\mathsf{sk}_{\mathsf{dv}}} = \ell_{\mathsf{sk}_{\mathsf{dv}}}(\lambda)$.

We require the message length μ of the underlying SKE scheme SKE to satisfy $\mu = \ell_s \cdot \ell_{\mathsf{sk}} + \ell'_{\mathsf{sk}} + \ell_{\mathsf{sk}_{\mathsf{dv}}}$. Finally, let $\mathsf{pad} = \mathsf{pad}(\lambda, n) \geq \mathsf{size}$ be a polynomial that is used as the size parameter for the underlying garbling scheme, and is specified differently in Theorem 4 in this section and in Theorem 5 in Section 6.

Using these ingredients, we construct our proposed PKE scheme $\mathsf{PKE}_{\mathsf{kdm}} = (\mathsf{KG}_{\mathsf{kdm}}, \mathsf{Enc}_{\mathsf{kdm}}, \mathsf{Dec}_{\mathsf{kdm}})$ whose message space is $\{0,1\}^{\ell_{\mathsf{m}}}$, as described in Figure 1.

Correctness. The correctness of $\mathsf{PKE}_{\mathsf{kdm}}$ follows from that of the building blocks. Specifically, let $(\mathsf{PK}, \mathsf{SK}) = (((\mathsf{pk}_{j,\alpha})_{j,\alpha}, \mathsf{pk}_{\mathsf{cca}}, \mathsf{pk}_{\mathsf{dv}}, \mathsf{ct}_{\mathsf{ske}}), s)$ be a key pair output by $\mathsf{KG}_{\mathsf{kdm}}$, let $m \in \{0,1\}^{\ell_{\mathsf{m}}}$ be any message, and let $\mathsf{CT} \leftarrow \mathsf{Enc}_{\mathsf{kdm}}(\mathsf{PK}, m)$ be an honestly generated ciphertext. Due to the correctness of PKE, PKE', SKE, and DVNIZK, each decryption/verification done in the execution of $\mathsf{Dec}_{\mathsf{kdm}}(\mathsf{PK}, \mathsf{SK}, \mathsf{CT})$ never fails, and just before the final step of $\mathsf{Dec}_{\mathsf{kdm}}$, the decryptor can recover a garbled circuit $\widetilde{\mathsf{Q}}$ and the labels $(\mathsf{lab}_j)_j$, which must have been generated as $(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j) \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{pad}, m)$. Hence, by the correctness of GC (in particular, correctness of the evaluation of a simulated garbled circuit and labels), we have $\mathsf{Eval}(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j) = m$.

Security. The following theorem guarantees the $\mathcal{B}_{\mathsf{size}}$-KDM$^{(1)}$-CCA security of the PKE scheme $\mathsf{PKE}_{\mathsf{kdm}}$.

Theorem 4 *Let $\ell_{\mathsf{m}} = \ell_{\mathsf{m}}(\lambda)$ and $\mathsf{size} = \mathsf{size}(\lambda) \geq \max\{\ell_s, \ell_{\mathsf{m}}\}$ be any polynomials, and let $\mathsf{pad} := \mathsf{size}$. Assume that PKE is IND-CPA secure, PKE' is IND-CCA secure, SKE is one-time $\mathcal{P}$-KDM secure, GC is a secure garbling scheme, and DVNIZK is a reusable DV-NIZK argument system for the NP language L. Then, $\mathsf{PKE}_{\mathsf{kdm}}$ is $\mathcal{B}_{\mathsf{size}}$-KDM$^{(1)}$-CCA secure.*

Proof of Theorem 4. Let $\mathcal{A}$ be an arbitrary PPT adversary that attacks the $\mathcal{B}_{\mathsf{size}}$-KDM$^{(1)}$-CCA security of $\mathsf{PKE}_{\mathsf{kdm}}$. We proceed the proof via a sequence of games argument using eight games. For every $t \in [7]$, let $\mathtt{SUC}_t$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit b in Game t. (Game 8 will be used only to bound the probability of a bad event introduced later.)

Game 1: This is the original $\mathcal{B}_{\mathsf{size}}$-KDM$^{(1)}$-CCA game regarding $\mathsf{PKE}_{\mathsf{kdm}}$. By definition, we have $\mathsf{Adv}^{\mathsf{kdmcca}}_{\mathsf{PKE}_{\mathsf{kdm}},\mathcal{B}_{\mathsf{size}},\mathcal{A},1}(\lambda) = 2 \cdot |\Pr[\mathtt{SUC}_1] - 1/2|$.

The detailed description of the game is as follows.

[14] Intuitively, a statement $(\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j\in[\ell_s],\alpha\in\{0,1\}}$ of the language L constitutes a $(\ell_s \times 2)$-matrix of public key/ciphertext pairs, and it is in L if the ciphertexts $\mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}$ in the j-th row encrypt the same plaintext lab_j for each $j \in [\ell_s]$.

$\mathsf{KG}_{\mathsf{kdm}}(1^\lambda)$:
$\forall(j,\alpha) \in [\ell_s] \times \{0,1\} : (\mathsf{pk}_{j,\alpha}, \mathsf{sk}_{j,\alpha}) \leftarrow \mathsf{KG}(1^\lambda)$
$(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}}) \leftarrow \mathsf{KG}_{\mathsf{cca}}(1^\lambda)$
$(\mathsf{pk}_{\mathsf{dv}}, \mathsf{sk}_{\mathsf{dv}}) \leftarrow \mathsf{DVKG}(1^\lambda)$
$s = (s_1, \ldots, s_{\ell_s}) \leftarrow \mathsf{K}(1^\lambda)$
$\mathsf{ct}_{\mathsf{ske}} \leftarrow \mathsf{E}(s, ((\mathsf{sk}_{j,s_j})_j, \mathsf{sk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{dv}}))$
$\mathsf{PK} \leftarrow ((\mathsf{pk}_{j,\alpha})_{j,\alpha}, \mathsf{pk}_{\mathsf{cca}}, \mathsf{pk}_{\mathsf{dv}}, \mathsf{ct}_{\mathsf{ske}})$; $\mathsf{SK} \leftarrow s$
Return $(\mathsf{PK}, \mathsf{SK})$.

$\mathsf{Enc}_{\mathsf{kdm}}(\mathsf{PK}, m)$:
$((\mathsf{pk}_{j,\alpha})_{j,\alpha}, \mathsf{pk}_{\mathsf{cca}}, \mathsf{pk}_{\mathsf{dv}}, \mathsf{ct}_{\mathsf{ske}}) \leftarrow \mathsf{PK}$
$(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j) \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{pad}, m)$ (†)
$\forall(j,\alpha) \in [\ell_s] \times \{0,1\}$:
$r_{j,\alpha} \xleftarrow{\mathsf{r}} \mathcal{R}$
$\mathsf{ct}_{j,\alpha} \leftarrow \mathsf{Enc}(\mathsf{pk}_{j,\alpha}, \mathsf{lab}_j; r_{j,\alpha})$
$x \leftarrow (\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$
$w \leftarrow (\mathsf{lab}_j, r_{j,0}, r_{j,1})_j$
$\pi \leftarrow \mathsf{P}(\mathsf{pk}_{\mathsf{dv}}, x, w)$
$\mathsf{CT} \leftarrow \mathsf{Enc}_{\mathsf{cca}}(\mathsf{pk}_{\mathsf{cca}}, (\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi))$
Return CT.

$\mathsf{Dec}_{\mathsf{kdm}}(\mathsf{PK}, \mathsf{SK}, \mathsf{CT})$: (⋆)
$((\mathsf{pk}_{j,\alpha})_{j,\alpha}, \mathsf{pk}_{\mathsf{cca}}, \mathsf{pk}_{\mathsf{dv}}, \mathsf{ct}_{\mathsf{ske}}) \leftarrow \mathsf{PK}$
$s = (s_1, \ldots, s_{\ell_s}) \leftarrow \mathsf{SK}$
$((\mathsf{sk}_{j,s_j})_j, \mathsf{sk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{dv}}) \leftarrow \mathsf{D}(s, \mathsf{ct}_{\mathsf{ske}})$
$(\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) \leftarrow \mathsf{Dec}_{\mathsf{cca}}(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}}, \mathsf{CT})$
$x \leftarrow (\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$
If $\mathsf{V}(\mathsf{sk}_{\mathsf{dv}}, x, \pi) = \mathsf{reject}$ then return $\perp$.
$\forall j \in [\ell_s] : \mathsf{lab}_j \leftarrow \mathsf{Dec}(\mathsf{pk}_{j,s_j}, \mathsf{sk}_{j,s_j}, \mathsf{ct}_{j,s_j})$
Return $m \leftarrow \mathsf{Eval}(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j)$.

Figure 1: The proposed PKE scheme $\mathsf{PKE}_{\mathsf{kdm}}$. The notations like $(X_{j,\alpha})_{j,\alpha}$ and $(X_j)_j$ are abbreviations for $(X_{j,\alpha})_{j\in[\ell_s],\alpha\in\{0,1\}}$ and $(X_j)_{j\in[\ell_s]}$, respectively. (⋆) If D, Dec, or $\mathsf{Dec}_{\mathsf{cca}}$ returns $\perp$, then we make $\mathsf{Dec}_{\mathsf{kdm}}$ return $\perp$ and terminate. (†) $\mathsf{pad} = \mathsf{pad}(\lambda, n)$ denotes the size parameter that is specified differently in each of Theorems 4 and 5.

1. The challenger chooses the challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$, and generates a key pair $(\mathsf{PK}, \mathsf{SK})$ of $\mathsf{PKE}_{\mathsf{kdm}}$ as follows.
   (a) For every $j \in [\ell_s]$ and $\alpha \in \{0,1\}$, generate $(\mathsf{pk}_{j,\alpha}, \mathsf{sk}_{j,\alpha}) \leftarrow \mathsf{KG}(1^\lambda)$.
   (b) Generate $(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}}) \leftarrow \mathsf{KG}_{\mathsf{cca}}(1^\lambda)$, $(\mathsf{pk}_{\mathsf{dv}}, \mathsf{sk}_{\mathsf{dv}}) \leftarrow \mathsf{DVKG}(1^\lambda)$, and $s = (s_1, \ldots, s_{\ell_s}) \leftarrow \mathsf{K}(1^\lambda)$.
   (c) Compute $\mathsf{ct}_{\mathsf{ske}} \leftarrow \mathsf{E}(s, ((\mathsf{sk}_{j,s_j})_j, \mathsf{sk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{dv}}))$.
   (d) Set $\mathsf{PK} := ((\mathsf{pk}_{j,\alpha})_{j,\alpha}, \mathsf{pk}_{\mathsf{cca}}, \mathsf{pk}_{\mathsf{dv}}, \mathsf{ct}_{\mathsf{ske}})$ and $\mathsf{SK} := s$.

   The challenger sends PK to $\mathcal{A}$, and also prepares an empty list L_{kdm}.
2. $\mathcal{A}$ may adaptively make the following queries.

   KDM-encryption queries: $\mathcal{A}$ sends $(f_0, f_1) \in \mathcal{B}^2_{\mathsf{size}}$ to the challenger. The challenger responds as follows.
   (a) Compute $(\widetilde{\mathsf{Q}}, \{\mathsf{lab}_j\}_j) \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{pad} = \mathsf{size}, f_b(s))$.
   (b) For every $j \in [\ell_s]$ and $\alpha \in \{0,1\}$, pick $r_{j,\alpha} \xleftarrow{\mathsf{r}} \mathcal{R}$ and compute $\mathsf{ct}_{j,\alpha} \leftarrow \mathsf{Enc}(\mathsf{pk}_{j,\alpha}, \mathsf{lab}_j; r_{j,\alpha})$.
   (c) Set $x := (\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$ and $w := (\mathsf{lab}_j, r_{j,0}, r_{j,1})_j$, and compute $\pi \leftarrow \mathsf{P}(\mathsf{pk}_{\mathsf{dv}}, x, w)$.
   (d) Return $\mathsf{CT} \leftarrow \mathsf{Enc}_{\mathsf{cca}}(\mathsf{pk}_{\mathsf{cca}}, (\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi))$ to $\mathcal{A}$, and add CT to the list L_{kdm}.

   Decryption queries: $\mathcal{A}$ sends CT to the challenger. The challenger returns $\perp$ to $\mathcal{A}$ if $\mathsf{CT} \in L_{\mathsf{kdm}}$, and otherwise responds as follows.
   (a) Compute $(\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) \leftarrow \mathsf{Dec}_{\mathsf{cca}}(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}}, \mathsf{CT})$, and set $x := (\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$.
   (b) If $\mathsf{V}(\mathsf{sk}_{\mathsf{dv}}, x, \pi) = \mathsf{reject}$, then return $\perp$ to $\mathcal{A}$.
   (c) For every $j \in [\ell_s]$, compute $\mathsf{lab}_j \leftarrow \mathsf{Dec}(\mathsf{pk}_{j,s_j}, \mathsf{sk}_{j,s_j}, \mathsf{ct}_{j,s_j})$.

(d) Return $m \leftarrow \mathsf{Eval}(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j)$ to $\mathcal{A}$.

Note that the above procedure is not exactly the same as $\mathsf{Dec}_{\mathsf{kdm}}(\mathsf{PK}, \mathsf{SK} = s, \mathsf{CT})$, since the computation of $\mathsf{D}(s, \mathsf{ct}_{\mathsf{ske}})$ for retrieving $((\mathsf{sk}_{j,s_j})_j, \mathsf{sk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{dv}})$ is omitted. However, the answer to a decryption query computed by the above procedure is exactly the same as that computed by $\mathsf{Dec}_{\mathsf{kdm}}$. Therefore, it does not affect $\mathcal{A}$'s view.

3. $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Game 2: Same as Game 1, except that the challenger uses the simulator $\mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2)$ for the zero-knowledge property of DVNIZK for generating $(\mathsf{pk}_{\mathsf{dv}}, \mathsf{sk}_{\mathsf{dv}})$ and a proof π in generating a ciphertext in response to KDM-encryption queries, instead of using DVKG and P. Namely, when generating PK and SK, the challenger generates $(\mathsf{pk}_{\mathsf{dv}}, \mathsf{sk}_{\mathsf{dv}}, \mathsf{td}) \leftarrow \mathsf{S}_1(1^\lambda)$ instead of $(\mathsf{pk}_{\mathsf{dv}}, \mathsf{sk}_{\mathsf{dv}}) \leftarrow \mathsf{DVKG}(1^\lambda)$. In addition, when $\mathcal{A}$ makes a KDM-encryption query (f_0, f_1), the challenger computes $\pi \leftarrow \mathsf{S}_2(\mathsf{td}, x)$ instead of $\pi \leftarrow \mathsf{P}(\mathsf{pk}_{\mathsf{dv}}, x, w)$, where $x = (\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$ and $w = (\mathsf{lab}_j, r_{j,0}, r_{j,1})_j$.

Due to the zero-knowledge property of DVNIZK, we have $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| = \mathsf{negl}(\lambda)$.

Game 3: Same as Game 2, except that when responding to a KDM-encryption query, the challenger generates a garbled circuit $\widetilde{\mathsf{Q}}$ and labels $(\mathsf{lab}_j)_j$ by garbling f_b. More precisely, when $\mathcal{A}$ makes a KDM-encryption query (f_0, f_1), the challenger computes $(\widetilde{\mathsf{Q}}, (\mathsf{lab}_{j,\alpha})_{j,\alpha}) \leftarrow \mathsf{Garble}(1^\lambda, f_b)$, instead of $(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j) \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{pad}, f_b(s))$. Moreover, for every $j \in [\ell_s]$ and $\alpha \in \{0,1\}$, the challenger computes $\mathsf{ct}_{j,\alpha} \leftarrow \mathsf{Enc}(\mathsf{pk}_{j,\alpha}, \mathsf{lab}_{j,s_j})$.[15]

By definition, the circuit size of f_b is $\mathsf{pad} = \mathsf{size}$. Hence, by the security of GC, we have $|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]| = \mathsf{negl}(\lambda)$.

Game 4: Same as Game 3, except that when responding to a KDM-encryption query (f_0, f_1), the challenger computes $\mathsf{ct}_{j,1\oplus s_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{j,1\oplus s_j}, \mathsf{lab}_{j,1\oplus s_j})$ for every $j \in [\ell_s]$. Due to the change made in this game, the challenger now computes $\mathsf{ct}_{j,\alpha} \leftarrow \mathsf{Enc}(\mathsf{pk}_{j,\alpha}, \mathsf{lab}_{j,\alpha})$ for every $j \in [\ell_s]$ and $\alpha \in \{0,1\}$.

In Games 3 and 4, we do not need the secret keys $(\mathsf{sk}_{j,1\oplus s_j})_j$ of PKE that do not correspond to $s = (s_1, \ldots, s_{\ell_s})$ (though we need $(\mathsf{sk}_{j,s_j})_j$ for computing $\mathsf{ct}_{\mathsf{ske}}$ and responding to decryption queries). Therefore, by the IND-CPA security of PKE under the keys $(\mathsf{pk}_{j,1\oplus s_j})_j$, we have $|\Pr[\mathtt{SUC}_3] - \Pr[\mathtt{SUC}_4]| = \mathsf{negl}(\lambda)$.

At this point, the challenger need not use s to respond to KDM-encryption queries. In the next game, we will ensure that the challenger does not use s to respond to decryption queries.

Game 5: Same as Game 4, except that when responding to a decryption query, the challenger computes the labels $(\mathsf{lab}_j)_j$ of a garbled circuit by decrypting $\mathsf{ct}_{j,0}$, instead of ct_{j,s_j}, for every $j \in [\ell_s]$. More precisely, the challenger responds to decryption queries as follows.

Decryption queries: $\mathcal{A}$ sends CT to the challenger. The challenger returns $\perp$ to $\mathcal{A}$ if $\mathsf{CT} \in L_{\mathsf{kdm}}$, and otherwise responds as follows. (The change from the previous game is underlined.)

1. Compute $(\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) \leftarrow \mathsf{Dec}_{\mathsf{cca}}(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}}, \mathsf{CT})$, and set $x := (\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$.
2. If $\mathsf{V}(\mathsf{sk}_{\mathsf{dv}}, x, \pi) = \mathsf{reject}$, then return $\perp$ to $\mathcal{A}$.

[15] Note that in Game 3, the labels of the "opposite" positions, namely $(\mathsf{lab}_{j,1\oplus s_j})_j$, are not used. They will be used in the subsequent games.

3. For every $j \in [\ell_s]$, compute $\underline{\mathsf{lab}_j \leftarrow \mathsf{Dec}(\mathsf{pk}_{j,0}, \mathsf{sk}_{j,0}, \mathsf{ct}_{j,0})}$.
4. Return $m \leftarrow \mathsf{Eval}(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j)$ to $\mathcal{A}$.

(By the change made in this game, s is not needed for responding to decryption queries.)

We define the following events in Game $t \in \{4, \ldots, 8\}$.

$\mathtt{BDQ}_t$**:** In Game t, $\mathcal{A}$ makes a decryption query $\mathsf{CT} \notin L_{\mathsf{kdm}}$ that satisfies the following two conditions, where $(\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) \leftarrow \mathsf{Dec}_{\mathsf{cca}}(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}}, \mathsf{CT})$:

1. $\mathsf{V}(\mathsf{sk}_{\mathsf{dv}}, (\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) = \mathsf{accept}$.
2. There exists $j^* \in [\ell_s]$ such that $\mathsf{Dec}(\mathsf{pk}_{j^*,0}, \mathsf{sk}_{j^*,0}, \mathsf{ct}_{j^*,0}) \neq \mathsf{Dec}(\mathsf{pk}_{j^*,1}, \mathsf{sk}_{j^*,1}, \mathsf{ct}_{j^*,1})$.

We call such a decryption query a *bad decryption query*.

Games 4 and 5 are identical unless $\mathcal{A}$ makes a bad decryption query in the corresponding games. Therefore, we have $|\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| \leq \Pr[\mathtt{BDQ}_5]$.

Game 6: Same as Game 5, except that when generating PK, the challenger generates $\mathsf{ct}_{\mathsf{ske}} \leftarrow \mathsf{E}(s, 0^\mu)$, instead of $\mathsf{ct}_{\mathsf{ske}} \leftarrow \mathsf{E}(s, ((\mathsf{sk}_{j,s_j})_j, \mathsf{sk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{dv}}))$.

In Games 5 and 6, when generating PK, the challenger does not need the secret key s of SKE except for the step of computing $\mathsf{ct}_{\mathsf{ske}}$. Furthermore, the "message" $((\mathsf{sk}_{j,s_j})_j, \mathsf{sk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{dv}})$ encrypted in $\mathsf{ct}_{\mathsf{ske}}$ in Game 5 can be described by a projection function of s. Thus, by the one-time $\mathcal{P}$-KDM security of SKE, we have $|\Pr[\mathtt{SUC}_5] - \Pr[\mathtt{SUC}_6]| = \mathsf{negl}(\lambda)$. In addition, whether $\mathcal{A}$ has submitted a bad decryption query can be detected by using $\mathsf{sk}_{\mathsf{cca}}$, $\mathsf{sk}_{\mathsf{dv}}$, and $(\mathsf{sk}_{j,\alpha})_{j,\alpha}$, without using s. Thus, again by the one-time $\mathcal{P}$-KDM security of SKE, we have $|\Pr[\mathtt{BDQ}_5] - \Pr[\mathtt{BDQ}_6]| = \mathsf{negl}(\lambda)$.

Game 7: Same as Game 6, except that when responding to a KDM-encryption query, the challenger computes $\mathsf{CT} \leftarrow \mathsf{Enc}_{\mathsf{cca}}(\mathsf{pk}_{\mathsf{cca}}, 0^{\ell'})$, where $\ell' = |\widetilde{\mathsf{Q}}| + 2\ell_s \cdot |\mathsf{ct}_{j,\alpha}| + |\pi|$.

Recall that in the previous game, we have eliminated the information of $\mathsf{sk}_{\mathsf{cca}}$ from $\mathsf{ct}_{\mathsf{ske}}$. Thus, we can rely on the IND-CCA security of PKE' at this point, and straightforwardly derive $|\Pr[\mathtt{SUC}_6] - \Pr[\mathtt{SUC}_7]| = \mathsf{negl}(\lambda)$. Moreover, a reduction algorithm (attacking the IND-CCA security of PKE') can detect whether $\mathcal{A}$'s decryption query is bad by using $(\mathsf{sk}_{j,\alpha})_{j,\alpha}$, $\mathsf{sk}_{\mathsf{dv}}$, and the reduction algorithm's own decryption queries. Thus, again by the IND-CCA security of PKE', we have $|\Pr[\mathtt{BDQ}_6] - \Pr[\mathtt{BDQ}_7]| = \mathsf{negl}(\lambda)$.

We see that in Game 7, the challenge bit b is information-theoretically hidden from $\mathcal{A}$'s view. Thus, we have $\Pr[\mathtt{SUC}_7] = 1/2$.

We need one more game to bound $\Pr[\mathtt{BDQ}_7]$.

Game 8: Same as Game 7, except that when generating PK, the challenger uses DVKG to generate $(\mathsf{pk}_{\mathsf{dv}}, \mathsf{sk}_{\mathsf{dv}})$, instead of using S_1. Namely, we undo the change made between Games 1 and 2 for generating $(\mathsf{pk}_{\mathsf{dv}}, \mathsf{sk}_{\mathsf{dv}})$.[16]

Due to the zero-knowledge property of DVNIZK, we have $|\Pr[\mathtt{BDQ}_7] - \Pr[\mathtt{BDQ}_8]| = \mathsf{negl}(\lambda)$.

Finally, we argue that the soundness of DVNIZK implies $\Pr[\mathtt{BDQ}_8] = \mathsf{negl}(\lambda)$. To see this, notice that in Game 8, $(\mathsf{pk}_{\mathsf{dv}}, \mathsf{sk}_{\mathsf{dv}})$ is now generated by DVKG. Also, if $\mathcal{A}$ submits a bad decryption query CT such that (1) $\mathsf{V}(\mathsf{sk}_{\mathsf{dv}}, (\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) = \mathsf{accept}$ and (2) $\mathsf{Dec}(\mathsf{pk}_{j^*,0}, \mathsf{sk}_{j^*,0}, \mathsf{ct}_{j^*,0}) \neq \mathsf{Dec}(\mathsf{pk}_{j^*,1}, \mathsf{sk}_{j^*,1}, \mathsf{ct}_{j^*,1})$ for some $j^* \in [\ell_s]$, where $(\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) \leftarrow \mathsf{Dec}_{\mathsf{cca}}(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}}, \mathsf{CT})$, then the condition (2) in particular implies $(\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha} \notin L$.

[16] Note that in Games 7 and 8, π is not computed when generating CT, and thus we need not use S_2.

Thus $((\mathsf{pk}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi)$ satisfies the condition of violating the soundness of DVNIZK. Note that a reduction algorithm (attacking the soundness of DVNIZK) is not directly given a secret verification key $\mathsf{sk}_{\mathsf{dv}}$. However, the reduction algorithm is allowed to make verification queries, which is sufficient to perfectly simulate Game 8 for $\mathcal{A}$. The reduction algorithm can also detect whether $\mathcal{A}$ has made a bad decryption query by using $\mathsf{sk}_{\mathsf{cca}}$ and $(\mathsf{sk}_{j,\alpha})_{j,\alpha}$, and verification queries. Hence, by the soundness of DVNIZK, we have $\Pr[\mathtt{BDQ}_8] = \mathsf{negl}(\lambda)$.

From the above arguments, we see that

$$\begin{aligned}
\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{kdmcca}}_{\mathsf{PKE}_{\mathsf{kdm}},\mathcal{B}_{\mathsf{size}},\mathcal{A},1}(\lambda) &= \left|\Pr[\mathtt{SUC}_1] - \frac{1}{2}\right| \\
&\leq \sum_{t\in[6]} |\Pr[\mathtt{SUC}_t] - \Pr[\mathtt{SUC}_{t+1}]| + \left|\Pr[\mathtt{SUC}_7] - \frac{1}{2}\right| \\
&= \sum_{t\in[6]\setminus\{4\}} |\Pr[\mathtt{SUC}_t] - \Pr[\mathtt{SUC}_{t+1}]| + |\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| \\
&\leq \sum_{t\in[6]\setminus\{4\}} |\Pr[\mathtt{SUC}_t] - \Pr[\mathtt{SUC}_{t+1}]| + \sum_{t\in\{5,6,7\}} |\Pr[\mathtt{BDQ}_t] - \Pr[\mathtt{BDQ}_{t+1}]| + \Pr[\mathtt{BDQ}_8] \\
&= \mathsf{negl}(\lambda).
\end{aligned}$$

Since the choice of $\mathcal{A}$ was arbitrary, we can conclude that $\mathsf{PKE}_{\mathsf{kdm}}$ is $\mathcal{B}_{\mathsf{size}}$-KDM$^{(1)}$-CCA secure. □ (**Theorem 4**)

6 Multi-User KDM-CCA Security from RKA-KDM Security

In this section, we show that for any polynomial $n = n(\lambda)$, our proposed PKE scheme $\mathsf{PKE}_{\mathsf{kdm}}$ presented in Section 5 can be shown to be $\mathcal{B}_{\mathsf{size}}$-KDM$^{(n)}$-CCA secure, by choosing a suitable parameter for $\mathsf{pad} = \mathsf{pad}(\lambda, n)$ and additionally requiring the underlying SKE scheme SKE satisfies $\mathcal{P}$-RKA-KDM$^{(n)}$ security, and its key generation algorithm outputs a uniformly random string in the secret key space. Formally, we prove the following theorem.

Theorem 5 *Let $n = n(\lambda)$, $\ell_{\mathsf{m}} = \ell_{\mathsf{m}}(\lambda)$, and $\mathsf{size} = \mathsf{size}(\lambda) \geq \max\{\ell_s, \ell_{\mathsf{m}}\}$ be any polynomials, and let $\mathsf{pad} := \mathsf{size} + O(\ell_s \cdot n)$.*[17] *Assume that PKE is IND-CPA secure, PKE' is IND-CCA secure, SKE is passively $\mathcal{P}$-RKA-KDM$^{(n)}$ secure and its key generation algorithm outputs a string that is distributed uniformly over $\{0,1\}^{\ell_s}$, GC is a secure garbling scheme, and DVNIZK is a reusable DV-NIZK argument system for the NP language L. Then, $\mathsf{PKE}_{\mathsf{kdm}}$ is $\mathcal{B}_{\mathsf{size}}$-KDM$^{(n)}$-CCA secure.*

A high-level structure of the sequence of the games used in the proof of Theorem 5 is similar to that of Theorem 4. The main differences are as follows.

- Before the game-hop for switching the simulator Sim of the garbling scheme GC to the ordinary algorithm Garble, we introduce a game in which every user's secret key s^i is derived by using a randomly chosen single "main" key $s \in \{0,1\}^{\ell_s}$ and a randomly chosen "shift" $\Delta^i \in \{0,1\}^{\ell_s}$, so that $s^i := s \oplus \Delta^i$. This does not at all change the distribution of the keys due to the requirement on SKE that a secret key is distributed uniformly in the secret key space $\{0,1\}^{\ell_s}$. This enables us to conduct the remaining game-hops as if $s \in \{0,1\}^{\ell_s}$ is the single "main" secret key such that we need to care only its leakage to an adversary via KDM-encryption and decryption queries.

[17] Looking ahead, this choice of pad corresponds to the size of the circuit Q specified in Figure 2.

- In the game-hop for switching the simulator Sim of GC to the ordinary garbling algorithm Garble, instead of directly garbling a KDM-function f_b (which is a function of all users' secret keys $S := s^1 \| \dots \| s^\ell_s$ in the n-user setting) appearing in an adversary's KDM-encryption query (i^*, f_0, f_1), we garble some appropriately designed circuit Q with input length ℓ_s. More specifically, we garble a circuit Q that has the index i^*, the KDM-function f_b, and the shifts $(\Delta^i)_{i\in[n]}$ hard-wired, and satisfies $f_b(S) = \mathsf{Q}(s^{i^*})$.
- In the game-hop for erasing the information of $((\mathsf{sk}^i_{j,s_j})_j, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{dv}})$ from $\mathsf{ct}^i_{\mathsf{ske}}$ for every $i \in [n]$, we rely on the passive $\mathcal{P}$-RKA-KDM$^{(n)}$ security of SKE (as opposed to its one-time $\mathcal{P}$-KDM security). Intuitively, passive $\mathcal{P}$-RKA-KDM$^{(n)}$ security suffices here because each user's secret key s^i is computed as $s^i = s \oplus \Delta^i$ where s and each Δ^i are chosen randomly by the challenger, due to the change made in the first item above.

Proof of Theorem 5. Let $n = n(\lambda)$ be an arbitrary polynomial that denotes the number of key pairs. Let $\mathcal{A}$ be an arbitrary PPT adversary that attacks the $\mathcal{B}_{\mathsf{size}}$-KDM$^{(n)}$-CCA security of $\mathsf{PKE}_{\mathsf{kdm}}$. We proceed the proof via a sequence of games argument using nine games. For every $t \in [8]$, let $\mathtt{SUC}_t$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit b in Game t. As in the proof of Theorem 4, the final game (Game 9) is used only to bound the probability of a bad event introduced later.

Game 1: This is the original $\mathcal{B}_{\mathsf{size}}$-KDM$^{(n)}$-CCA game regarding $\mathsf{PKE}_{\mathsf{kdm}}$. Then, we have $\mathsf{Adv}^{\mathsf{kdmcca}}_{\mathsf{PKE}_{\mathsf{kdm}},\mathcal{B}_{\mathsf{size}},\mathcal{A},n}(\lambda) = 2 \cdot |\Pr[\mathtt{SUC}_1] - 1/2|$.

The detailed description of the game is as follows.

1. The challenger chooses the challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$, and generates a key pair $(\mathsf{PK}^i, \mathsf{SK}^i)$ of $\mathsf{PKE}_{\mathsf{kdm}}$ for every $i \in [n]$ as follows.
   (a) For every $j \in [\ell_s]$ and $\alpha \in \{0,1\}$, generate $(\mathsf{pk}^i_{j,\alpha}, \mathsf{sk}^i_{j,\alpha}) \leftarrow \mathsf{KG}(1^\lambda)$.
   (b) Generate $(\mathsf{pk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{cca}}) \leftarrow \mathsf{KG}_{\mathsf{cca}}(1^\lambda)$, $(\mathsf{pk}^i_{\mathsf{dv}}, \mathsf{sk}^i_{\mathsf{dv}}) \leftarrow \mathsf{DVKG}(1^\lambda)$, and $s^i := (s^i_1, \dots, s^i_{\ell_s}) \xleftarrow{\mathsf{r}} \{0,1\}^{\ell_s}$.
   (c) Compute $\mathsf{ct}^i_{\mathsf{ske}} \leftarrow \mathsf{E}(s^i, ((\mathsf{sk}^i_{j,s_j})_j, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{dv}}))$.
   (d) Set $\mathsf{PK}^i := ((\mathsf{pk}^i_{j,\alpha})_{j,\alpha}, \mathsf{pk}^i_{\mathsf{cca}}, \mathsf{pk}^i_{\mathsf{dv}}, \mathsf{ct}^i_{\mathsf{ske}})$ and $\mathsf{SK}^i := s^i$.

   The challenger sends $(\mathsf{PK}^i)_{i\in[n]}$ to $\mathcal{A}$, and prepares an empty list L_{kdm}.
2. $\mathcal{A}$ may adaptively make the following queries. Below, let $S := s^1 \| \dots \| s^n$.
   KDM-encryption queries: $\mathcal{A}$ sends $(i^*, f_0, f_1) \in [n] \times \mathcal{B}^2_{\mathsf{size}}$ to the challenger. The challenger responds as follows.
   (a) Compute $(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j) \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{pad}, f_b(S))$.
   (b) For every $j \in [\ell_s]$ and $\alpha \in \{0,1\}$, pick $r_{j,\alpha} \xleftarrow{\mathsf{r}} \mathcal{R}$ and compute $\mathsf{ct}_{j,\alpha} \leftarrow \mathsf{Enc}(\mathsf{pk}^{i^*}_{j,\alpha}, \mathsf{lab}_j; r_{j,\alpha})$.
   (c) Set $x := (\mathsf{pk}^{i^*}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$ and $w := (\mathsf{lab}_j, r_{j,0}, r_{j,1})_j$, and compute $\pi \leftarrow \mathsf{P}(\mathsf{pk}^{i^*}_{\mathsf{dv}}, x, w)$.
   (d) Return $\mathsf{CT} \leftarrow \mathsf{Enc}_{\mathsf{cca}}(\mathsf{pk}^{i^*}_{\mathsf{cca}}, (\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi))$ to $\mathcal{A}$ and add (i^*, CT) to the list L_{kdm}.

   Decryption queries: $\mathcal{A}$ sends (i, CT) to the challenger. The challenger returns $\perp$ to $\mathcal{A}$ if $(i, \mathsf{CT}) \in L_{\mathsf{kdm}}$, and otherwise responds as follows.
   (a) Compute $(\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) \leftarrow \mathsf{Dec}_{\mathsf{cca}}(\mathsf{pk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{CT})$, and set $x := (\mathsf{pk}^i_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$.
   (b) If $\mathsf{V}(\mathsf{sk}^i_{\mathsf{dv}}, x, \pi) = \mathsf{reject}$, then return $\perp$ to $\mathcal{A}$.

Circuit $\mathsf{Q}[i^*, f, (\Delta^i)_{i\in[n]}](\cdot)$:

Hardwired: An index $i^* \in [n]$, a circuit $f : \{0,1\}^{\ell_s \cdot n} \to \{0,1\}^{\ell_m}$, and strings $(\Delta^i)_{i\in[n]} \in (\{0,1\}^{\ell_s})^n$.
Input: A string $z \in \{0,1\}^{\ell_s}$.

1. For every $i \in [n]$, compute $z^i \leftarrow z \oplus \Delta^{i^*} \oplus \Delta^i$.
2. Return $m \leftarrow f(z^1 \| \dots \| z^n)$.

Figure 2: Description of the circuit Q.

(c) For every $j \in [\ell_s]$, compute $\mathsf{lab}_j \leftarrow \mathsf{Dec}(\mathsf{pk}^i_{j,s^i_j}, \mathsf{sk}^i_{j,s^i_j}, \mathsf{ct}_{j,s^i_j})$.

(d) Return $m \leftarrow \mathsf{Eval}(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j)$ to $\mathcal{A}$.

Note that the above procedure is not exactly the same as $\mathsf{Dec}_{\mathsf{kdm}}(\mathsf{PK}^i, \mathsf{SK}^i = s^i, \mathsf{CT})$, because the computations of $\mathsf{D}(s^i, \mathsf{ct}^i_{\mathsf{ske}})$ for retrieving $((\mathsf{sk}^i_{j,s^i_j})_j, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{dv}})$ is omitted. However, the answer to a decryption query computed by the above procedure is exactly the same as that computed by $\mathsf{Dec}_{\mathsf{kdm}}$. Therefore, it does not affect $\mathcal{A}$'s view.

3. $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Game 2: Same as Game 1, except that the challenger uses the simulator $\mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2)$ for the zero-knowledge property of DVNIZK for generating $(\mathsf{pk}^i_{\mathsf{dv}}, \mathsf{sk}^i_{\mathsf{dv}})$ for every $i \in [n]$ and a proof π in generating a ciphertext in response to KDM-encryption queries, instead of using DVKG and P. Namely, when generating PK^i and SK^i, the challenger generates $(\mathsf{pk}^i_{\mathsf{dv}}, \mathsf{sk}^i_{\mathsf{dv}}, \mathsf{td}^i) \leftarrow \mathsf{S}_1(1^\lambda)$ instead of $(\mathsf{pk}^i_{\mathsf{dv}}, \mathsf{sk}^i_{\mathsf{dv}}) \leftarrow \mathsf{DVKG}(1^\lambda)$. In addition, when $\mathcal{A}$ makes a KDM-encryption query (i^*, f_0, f_1), the challenger computes $\pi \leftarrow \mathsf{S}_2(\mathsf{td}^{i^*}, x)$ instead of $\pi \leftarrow \mathsf{P}(\mathsf{pk}^{i^*}_{\mathsf{dv}}, x, w)$, where $x = (\mathsf{pk}^{i^*}_{j,\alpha}, \mathsf{ct}_{i,\alpha})_{j,\alpha}$ and $w = (\mathsf{lab}_j, r_{j,0}, r_{j,1})_j$.

Due to the zero-knowledge property of DVNIZK, we have $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| = \mathsf{negl}(\lambda)$.

Game 3: Same as Game 2, except for how the secret keys $(s^i)_{i\in[n]}$ are generated. Specifically, in this game, the challenger first generates a single "main" key $s \xleftarrow{r} \{0,1\}^{\ell_s}$. Then, for every $i \in [n]$, the challenger generates the "shift" $\Delta^i \xleftarrow{r} \{0,1\}^{\ell_s}$ and sets $s^i := s \oplus \Delta^i$.

For every $i \in [n]$, the distribution of s^i in Game 3 is identical to that in Game 2. Thus, we have $\Pr[\mathtt{SUC}_2] = \Pr[\mathtt{SUC}_3]$.

Game 4: Same as Game 3, except that when responding to a KDM-encryption query from $\mathcal{A}$, the challenger generates a garbled circuit by garbling the circuit Q shown in Figure 2. More precisely, when $\mathcal{A}$ makes a KDM-encryption query (i^*, f_0, f_1), the challenger computes $(\widetilde{\mathsf{Q}}, (\mathsf{lab}_{j,\alpha})_{j,\alpha}) \leftarrow \mathsf{Garble}(1^\lambda, \mathsf{Q}[i^*, f_b, (\Delta^i)_{i\in[n]}])$. Moreover, for every $j \in [\ell_s]$ and $\alpha \in \{0,1\}$, the challenger computes $\mathsf{ct}_{j,\alpha} \leftarrow \mathsf{Enc}(\mathsf{pk}^{i^*}_{j,\alpha}, \mathsf{lab}_{j,s^{i^*}_j})$.[18]

Note that $s^{i^*} \oplus \Delta^{i^*} \oplus \Delta^i = s \oplus \Delta^i = s^i$ holds for all $i \in [n]$. Thus, we have

$$\mathsf{Q}[i^*, f_b, (\Delta^i)_{i\in[n]}](s^{i^*}) = f_b(s^1 \| \dots \| s^n) = f_b(S).$$

Note also that the size of Q generated in this way is $\mathsf{pad} = \mathsf{size} + O(\ell_s \cdot n)$. Thus, by the security of GC, we have $|\Pr[\mathtt{SUC}_3] - \Pr[\mathtt{SUC}_4]| = \mathsf{negl}(\lambda)$.

[18] Note that in Game 4, the labels of the "opposite" positions, namely $(\mathsf{lab}_{j,1\oplus s^{i^*}_j})_j$, are not used. They will be used in the subsequent games.

Game 5: Same as Game 4, except that when responding to a KDM-encryption query (i^*, f_0, f_1), the challenger computes $\mathsf{ct}_{j,1\oplus s_j^{i^*}} \leftarrow \mathsf{Enc}(\mathsf{pk}^{i^*}_{j,1\oplus s_j^{i^*}}, \mathsf{lab}_{j,1\oplus s_j^{i^*}})$ for every $j \in [\ell_s]$. Due to the change made in this game, the challenger now computes $\mathsf{ct}_{j,\alpha} \leftarrow \mathsf{Enc}(\mathsf{pk}^{i^*}_{j,\alpha}, \mathsf{lab}_{j,\alpha})$ for every $j \in [\ell_s]$ and $\alpha \in \{0,1\}$.

In Games 4 and 5, we do not need the secret keys $(\mathsf{sk}^{i^*}_{j,1\oplus s_j^{i^*}})_j$ of PKE that do not correspond to s^{i^*} (though we need $(\mathsf{sk}^{i^*}_{j,s_j^{i^*}})_j$ for computing $\mathsf{ct}^{i^*}_{\mathsf{ske}}$ and responding to decryption queries). Therefore, by the IND-CPA security of PKE under the keys $(\mathsf{pk}^{i^*}_{j,1\oplus s_j^{i^*}})_j$, we have $|\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| = \mathsf{negl}(\lambda)$.

At this points, the challenger need not use the main key s to respond to KDM-encryption queries. In the next game, we will ensure that the challenger does not use the main key s to respond to decryption queries.

Game 6: Same as Game 5, except that when responding to a decryption query, the challenger computes labels $(\mathsf{lab}_j)_j$ of a garbled circuit by decrypting $\mathsf{ct}_{j,0}$, instead of ct_{j,s_j^i}, for every $j \in [\ell_s]$. More precisely, the challenger responds to decryption queries as follows.

Decryption queries: $\mathcal{A}$ sends (i, CT) to the challenger. The challenger returns $\perp$ to $\mathcal{A}$ if $(i, \mathsf{CT}) \in L_{\mathsf{kdm}}$, and otherwise responds as follows. (The change from the previous game is underlined.)

1. Compute $(\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) \leftarrow \mathsf{Dec}_{\mathsf{cca}}(\mathsf{pk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{CT})$, and set $x := (\mathsf{pk}^i_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$.
2. If $\mathsf{V}(\mathsf{sk}^i_{\mathsf{dv}}, x, \pi) = \mathsf{reject}$, then return $\perp$ to $\mathcal{A}$.
3. For every $j \in [\ell_s]$, compute $\underline{\mathsf{lab}_j \leftarrow \mathsf{Dec}(\mathsf{pk}^i_{j,0}, \mathsf{sk}^i_{j,0}, \mathsf{ct}_{j,0})}$.
4. Return $m \leftarrow \mathsf{Eval}(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j)$ to $\mathcal{A}$.

(By the change made in this game, the main key s is not needed for responding to decryption queries.)

We define the following events in Game $t \in \{5, \ldots, 9\}$.

$\mathtt{BDQ}_t$**:** In Game t, $\mathcal{A}$ makes a decryption query $(i, \mathsf{CT}) \notin L_{\mathsf{kdm}}$ that satisfies the following two conditions, where $(\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) \leftarrow \mathsf{Dec}_{\mathsf{cca}}(\mathsf{pk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{CT})$:

1. $\mathsf{V}(\mathsf{sk}^i_{\mathsf{dv}}, (\mathsf{pk}^i_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) = \mathsf{accept}$.
2. There exists $j^* \in [\ell_s]$ such that $\mathsf{Dec}(\mathsf{pk}^i_{j^*,0}, \mathsf{sk}^i_{j^*,0}, \mathsf{ct}_{j^*,0}) \neq \mathsf{Dec}(\mathsf{pk}^i_{j^*,1}, \mathsf{sk}^i_{j^*,1}, \mathsf{ct}_{j^*,1})$.

We call such a decryption query a *bad decryption query.*

Games 5 and 6 are identical unless $\mathcal{A}$ makes a bad decryption query in the corresponding games. Therefore, we have $|\Pr[\mathtt{SUC}_5] - \Pr[\mathtt{SUC}_6]| \leq \Pr[\mathtt{BDQ}_6]$.

Game 7: Same as Game 6, except that for every $i \in [n]$, when generating PK^i, the challenger computes $\mathsf{ct}^i_{\mathsf{ske}} \leftarrow \mathsf{E}(s^i, 0^\mu)$, instead of $\mathsf{ct}^i_{\mathsf{ske}} \leftarrow \mathsf{E}(s^i, ((\mathsf{sk}^i_{j,s_j^i})_j, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{dv}}))$.

We argue that the passive $\mathcal{P}$-RKA-KDM$^{(n)}$ security of SKE implies $|\Pr[\mathtt{SUC}_6] - \Pr[\mathtt{SUC}_7]| = \mathsf{negl}(\lambda)$. To see this, consider the following PPT adversary $\mathcal{A}_{\mathsf{rka}}$ that uses $\mathcal{A}$ as a subroutine and attacks the passive $\mathcal{P}$-RKA-KDM$^{(n)}$ security of SKE.

1. Given 1^λ and the random "shifts" $(\Delta^i)_{i\in[n]}$ from $\mathcal{A}_{\mathsf{rka}}$'s challenger, $\mathcal{A}_{\mathsf{rka}}$ generates $b \xleftarrow{r} \{0,1\}$ and execute the following steps for every $i \in [n]$:

Circuit $g[\Delta^i, (\mathsf{sk}^i_{j,\alpha})_{j,\alpha}, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{dv}}](\cdot)$:

Hardwired: Strings Δ^i, $(\mathsf{sk}^i_{j,\alpha})_{j,\alpha}$, $\mathsf{sk}^i_{\mathsf{cca}}$, $\mathsf{sk}^i_{\mathsf{dv}}$.
Input: A string $z \in \{0,1\}^{\ell_s}$.

1. Set $z^i = (z^i_1, \ldots, z^i_{\ell_s}) := z \oplus \Delta^i$.
2. Return $((\mathsf{sk}^i_{j,z^i_j})_j, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{dv}})$.

Figure 3: Description of the circuit g.

(a) For every $j \in [\ell_s]$ and $\alpha \in \{0,1\}$, generate $(\mathsf{pk}^i_{j,\alpha}, \mathsf{sk}^i_{j,\alpha}) \leftarrow \mathsf{KG}(1^\lambda)$.
(b) Generate $(\mathsf{pk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{cca}}) \leftarrow \mathsf{KG}_{\mathsf{cca}}(1^\lambda)$ and $(\mathsf{pk}^i_{\mathsf{dv}}, \mathsf{sk}^i_{\mathsf{dv}}, \mathsf{td}^i) \leftarrow \mathsf{S}_1(1^\lambda)$.
(c) Prepare the following circuit:

$$g^i(\cdot) := g[\Delta^i, (\mathsf{sk}^i_{j,\alpha})_{j,\alpha}, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{dv}}](\cdot),$$

where g is described in Figure 3. Note that $g^i(s) = ((\mathsf{sk}^i_{j,s^i_j})_j, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{dv}})$.

Then, $\mathcal{A}_{\mathsf{rka}}$ submit n circuits $g^1, \ldots, g^n$ as a (one-shot) RKA-KDM-encryption query, and receives the challenge ciphertexts $(\mathsf{ct}^i_{\mathsf{ske}})_{i\in[n]}$ from the challenger. $\mathcal{A}_{\mathsf{rka}}$ then sets $\mathsf{PK}^i := ((\mathsf{pk}^i_{j,\alpha})_{j,\alpha}, \mathsf{pk}^i_{\mathsf{cca}}, \mathsf{pk}^i_{\mathsf{dv}}, \mathsf{ct}^i_{\mathsf{ske}})$ for every $i \in [n]$. Finally, $\mathcal{A}_{\mathsf{rka}}$ sends $(\mathsf{PK}^i)_{i\in[n]}$ to $\mathcal{A}$, and prepares an empty list L_{kdm}.

2. $\mathcal{A}_{\mathsf{rka}}$ responds to KDM-encryption and decryption queries made by $\mathcal{A}$ as follows.
KDM-encryption queries: $\mathcal{A}$ sends $(i^*, f_0, f_1) \in [n] \times \mathcal{B}^2_{\mathsf{size}}$. $\mathcal{A}_{\mathsf{rka}}$ responds as follows.
(a) Compute $(\widetilde{\mathsf{Q}}, (\mathsf{lab}_{j,\alpha})_{j,\alpha}) \leftarrow \mathsf{Garble}(1^\lambda, \mathsf{Q}[i^*, f_b, (\Delta^i)_{i\in[n]}])$.
(b) For every $j \in [\ell_s]$ and $\alpha \in \{0,1\}$, compute $\mathsf{ct}_{j,\alpha} \leftarrow \mathsf{Enc}(\mathsf{pk}^{i^*}_{j,\alpha}, \mathsf{lab}_{j,\alpha})$.
(c) Set $x := (\mathsf{pk}^{i^*}_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$ and compute $\pi \leftarrow \mathsf{S}_2(\mathsf{td}^{i^*}, x)$.
(d) Return $\mathsf{CT} \leftarrow \mathsf{Enc}_{\mathsf{cca}}(\mathsf{pk}^{i^*}_{\mathsf{cca}}, (\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi))$ to $\mathcal{A}$ and add (i^*, CT) to the list L_{kdm}.
Decryption queries: $\mathcal{A}$ sends (i, CT). $\mathcal{A}_{\mathsf{rka}}$ returns $\perp$ to $\mathcal{A}$ if $(i, \mathsf{CT}) \in L_{\mathsf{kdm}}$, and otherwise responds as follows.
(a) Compute $(\widetilde{\mathsf{Q}}, (\mathsf{ct}_{j,\alpha})_{j,\alpha}, \pi) \leftarrow \mathsf{Dec}_{\mathsf{cca}}(\mathsf{pk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{CT})$, and set $x := (\mathsf{pk}^i_{j,\alpha}, \mathsf{ct}_{j,\alpha})_{j,\alpha}$.
(b) If $\mathsf{V}(\mathsf{sk}^i_{\mathsf{dv}}, x, \pi) = \mathsf{reject}$, then return $\perp$ to $\mathcal{A}$.
(c) For every $j \in [\ell_s]$, compute $\mathsf{lab}_j \leftarrow \mathsf{Dec}(\mathsf{pk}^i_{j,0}, \mathsf{sk}^i_{j,0}, \mathsf{ct}_{j,0})$.
(d) Return $m \leftarrow \mathsf{Eval}(\widetilde{\mathsf{Q}}, (\mathsf{lab}_j)_j)$ to $\mathcal{A}$.

3. When $\mathcal{A}$ terminates with output $b' \in \{0,1\}$, $\mathcal{A}_{\mathsf{rka}}$ sets $\beta' := 1$ if $b = b'$, and otherwise sets $\beta' := 0$. Finally, $\mathcal{A}_{\mathsf{rka}}$ terminates with output β'.

We see that the function g in Figure 3 is a projection function since each bit of sk^i_{j,s^i_j} depends only on the j-th bit $s_j \in \{0,1\}$ of s for every $j \in [\ell_s]$, and $\mathsf{sk}^i_{\mathsf{cca}}$ and $\mathsf{sk}^i_{\mathsf{dv}}$ do not depend on the input s. Let $\beta \in \{0,1\}$ be the challenge bit in $\mathcal{A}_{\mathsf{rka}}$'s passive $\mathcal{P}$-RKA-KDM$^{(n)}$ game. Then, we have $\mathsf{Adv}^{\mathsf{prkakdm}}_{\mathsf{SKE},\mathcal{P},\mathcal{A}_{\mathsf{rka}},n}(\lambda) = |\Pr[\beta' = 1|\beta = 1] - \Pr[\beta' = 1|\beta = 0]|$.

If $\beta = 1$, then $\mathcal{A}_{\mathsf{rka}}$ perfectly simulates Game 6 for $\mathcal{A}$ so that the secret key in $\mathcal{A}_{\mathsf{rka}}$'s passive $\mathcal{P}$-RKA-KDM$^{(n)}$ game is the "main" key s in the game for $\mathcal{A}$ simulated by $\mathcal{A}_{\mathsf{rka}}$.

In particular, each $\mathsf{ct}^i_{\mathsf{ske}}$ is an encryption of $((\mathsf{sk}^i_{j,s^i_j})_j, \mathsf{sk}^i_{\mathsf{cca}}, \mathsf{sk}^i_{\mathsf{dv}})$ under the secret key $s^i = s \oplus \Delta^i$, which is exactly how it is generated in Game 6.

On the other hand, if $\beta = 0$, then $\mathcal{A}_{\mathsf{rka}}$ perfectly simulates Game 7 for $\mathcal{A}$. In particular, each $\mathsf{ct}^i_{\mathsf{ske}}$ is an encryption of 0^μ under the secret key $s^i = s \oplus \Delta^i$, which is exactly how it is generated in Game 7.

Furthermore, $\mathcal{A}_{\mathsf{rka}}$ outputs $\beta' = 1$ if and only if $\mathcal{A}$ succeeds in guessing the challenge bit b, i.e. $b = b'$ occurs. Thus, we have $\Pr[\beta' = 1|\beta = 0] = \Pr[\mathtt{SUC}_6]$ and $\Pr[\beta' = 1|\beta = 1] = \Pr[\mathtt{SUC}_7]$, and consequently we have $\mathsf{Adv}^{\mathsf{prkakdm}}_{\mathsf{SKE},\mathcal{P},\mathcal{A}_{\mathsf{rka}},n}(\lambda) = |\Pr[\mathtt{SUC}_6] - \Pr[\mathtt{SUC}_7]|$. Since SKE is assumed to be passively $\mathcal{P}$-RKA-KDM$^{(n)}$ secure, we have $|\Pr[\mathtt{SUC}_6] - \Pr[\mathtt{SUC}_7]| = \mathsf{negl}(\lambda)$.

In addition, note that $\mathcal{A}_{\mathsf{rka}}$ can detect whether a decryption query made by $\mathcal{A}$ is bad without using s, by using $(\mathsf{sk}^i_{j,\alpha})_{j,\alpha}$, $\mathsf{sk}^i_{\mathsf{cca}}$, and $\mathsf{sk}^i_{\mathsf{dv}}$, all of which are generated by $\mathcal{A}_{\mathsf{rka}}$ itself. Thus, by considering a slight variant of $\mathcal{A}_{\mathsf{rka}}$ that outputs $\beta' = 1$ if and only if $\mathcal{A}$ has submitted a bad decryption query, we can also derive $|\Pr[\mathtt{BDQ}_6] - \Pr[\mathtt{BDQ}_7]| = \mathsf{negl}(\lambda)$.

Game 8: Same as Game 7, except that when responding to a KDM-encryption query (i^*, f_0, f_1), the challenger computes $\mathsf{CT} \leftarrow \mathsf{Enc}_{\mathsf{cca}}(\mathsf{pk}^{i^*}_{\mathsf{cca}}, 0^{\ell'})$, where $\ell' = |\widetilde{\mathsf{Q}}| + 2\ell_s \cdot |\mathsf{ct}_{j,\alpha}| + |\pi|$.

Recall that in the previous game, we have eliminated the information of $\mathsf{sk}^i_{\mathsf{cca}}$ from $\mathsf{ct}^i_{\mathsf{ske}}$ for every $i \in [n]$. Thus, we can rely on the IND-CCA security of PKE' at this point, and straightforwardly derive $|\Pr[\mathtt{SUC}_7] - \Pr[\mathtt{SUC}_8]| = \mathsf{negl}(\lambda)$. Moreover, a reduction algorithm (attacking the IND-CCA security of PKE') can detect whether $\mathcal{A}$'s decryption query (i, CT) is bad by using $(\mathsf{sk}^i_{j,\alpha})_{j,\alpha}$, $\mathsf{sk}^i_{\mathsf{dv}}$, and the reduction algorithm's own decryption queries. Thus, again by the IND-CCA security of PKE', we have $|\Pr[\mathtt{BDQ}_7] - \Pr[\mathtt{BDQ}_8]| = \mathsf{negl}(\lambda)$.

We see that in Game 8, the challenge bit b is information-theoretically hidden from $\mathcal{A}$'s view. Thus, we have $\Pr[\mathtt{SUC}_8] = 1/2$.

We need one more game to bound $\Pr[\mathtt{BDQ}_8]$.

Game 9: Same as Game 8, except that for every $i \in [n]$, when generating PK^i, the challenger uses DVKG to generate $(\mathsf{pk}^i_{\mathsf{dv}}, \mathsf{sk}^i_{\mathsf{dv}})$, instead of using S_1. Namely, we undo the change made between Games 1 and 2 for generating $(\mathsf{pk}^i_{\mathsf{dv}}, \mathsf{sk}^i_{\mathsf{dv}})$ for every $i \in [n]$.

Due to the zero-knowledge property of DVNIZK, we have $|\Pr[\mathtt{BDQ}_8] - \Pr[\mathtt{BDQ}_9]| = \mathsf{negl}(\lambda)$. In addition, due to the soundness of DVNIZK, we also obtain $\Pr[\mathtt{BDQ}_9] = \mathsf{negl}(\lambda)$. (The reasoning is essentially identical to the corresponding step in the proof of Theorem 4.)

From the above arguments, we see that

$$\begin{aligned}
\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{kdmcca}}_{\mathsf{PKE},\mathcal{B}_{\mathsf{size}},\mathcal{A},n}(\lambda) &= |\Pr[\mathtt{SUC}_1] - \frac{1}{2}| \\
&\leq \sum_{t\in[7]} |\Pr[\mathtt{SUC}_t] - \Pr[\mathtt{SUC}_{t+1}]| + \left|\Pr[\mathtt{SUC}_8] - \frac{1}{2}\right| \\
&= \sum_{t\in[7]\setminus\{5\}} |\Pr[\mathtt{SUC}_t] - \Pr[\mathtt{SUC}_{t+1}]| + |\Pr[\mathtt{SUC}_5] - \Pr[\mathtt{SUC}_6]| \\
&\leq \sum_{t\in[7]\setminus\{5\}} |\Pr[\mathtt{SUC}_t] - \Pr[\mathtt{SUC}_{t+1}]| + \sum_{t\in\{6,7,8\}} |\Pr[\mathtt{BDQ}_t] - \Pr[\mathtt{BDQ}_{t+1}]| + \Pr[\mathtt{BDQ}_9] \\
&= \mathsf{negl}(\lambda).
\end{aligned}$$

Since the choice of $\mathcal{A}$ and n was arbitrary, we can conclude that $\mathsf{PKE}_{\mathsf{kdm}}$ is $\mathcal{B}_{\mathsf{size}}$-KDM$^{(n)}$-CCA secure. □ (**Theorem 5**)

7 Passively RKA-KDM Secure SKE from Hash Encryption

In this section, we show how to construct an RKA-KDM secure SKE scheme based on a hash encryption scheme.

This section is organized as follows. In Section 7.1, we first recall the definition of hash encryption. Then, in Section 7.2, we present our proposed SKE scheme and proves its passive RKA-KDM security.

7.1 Definition of Hash Encryption

Here, we review the definition of hash encryption introduced in [DGHM18].

Definition 8 (Hash Encryption) *A hash encryption scheme* HE *is a four tuple* $(\mathsf{HKG}, \mathsf{H}, \mathsf{HEnc}, \mathsf{HDec})$ *of PPT algorithms.*

- HKG *is the key generation algorithm that takes as input a security parameter* 1^λ *and the input-length* ℓ_{inp} *(where* $\ell_{\mathsf{inp}} = \ell_{\mathsf{inp}}(\lambda)$ *is a polynomial). Then, it outputs a hash key* hk*.*
- H *is the (deterministic) hashing algorithm that takes a hash key* hk *and a string* $x \in \{0,1\}^{\ell_{\mathsf{inp}}}$ *as input, and outputs a hash value* $h \in \{0,1\}^\lambda$*.*
- HEnc *is the encryption algorithm that takes a hash key* hk*, a triple* $(h, j, \alpha) \in \{0,1\}^\lambda \times [\ell_{\mathsf{inp}}] \times \{0,1\}$*, and a message* $m \in \{0,1\}^*$ *as input, and outputs a ciphertext* ct*.*
- HDec *is the (deterministic) decryption algorithm that takes a hash key* hk*, a string* $x \in \{0,1\}^{\ell_{\mathsf{inp}}}$*, and a ciphertext* ct *as input, and outputs a message* m *which could be the special invalid symbol* $\perp$*.*

We require the following properties.

Correctness *We require* $\mathsf{HDec}(\mathsf{hk}, x, \mathsf{HEnc}(\mathsf{hk}, (\mathsf{H}(\mathsf{hk}, x), j, x_j), m)) = m$ *for all* $\lambda \in \mathbb{N}$*, all polynomials* $\ell_{\mathsf{inp}} = \ell_{\mathsf{inp}}(\lambda)$*, all strings* $x = (x_1, \ldots, x_{\ell_{\mathsf{inp}}}) \in \{0,1\}^{\ell_{\mathsf{inp}}}$*, all positions* $j \in [\ell_{\mathsf{inp}}]$*, all hash keys* hk *output by* $\mathsf{HKG}(1^\lambda, \ell_{\mathsf{inp}})$*, and all messages* m*.*

Security *Consider the following security game between a challenger and an adversary* $\mathcal{A}$*.*

1. *First, the challenger chooses a challenge bit* $b \xleftarrow{\mathsf{r}} \{0,1\}$*. Next, the challenger generates* $\mathsf{hk} \leftarrow \mathsf{HKG}(1^\lambda, \ell_{\mathsf{inp}})$ *and sends* hk *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends* $x = (x_1, \ldots, x_{\ell_{\mathsf{inp}}}) \in \{0,1\}^{\ell_{\mathsf{inp}}}$*, a position* $j \in [\ell_{\mathsf{inp}}]$*, and a pair of messages* (m_0, m_1) *of the same length to the challenger. The challenger computes* $h \leftarrow \mathsf{H}(\mathsf{hk}, x)$ *and* $\mathsf{ct} \leftarrow \mathsf{E}(\mathsf{hk}, (h, j, 1 \oplus x_j), m_b)$*, and returns* ct *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$*.*

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

Hash encryption can be constructed under the CDH assumption [DGHM18, BLSV18].

$\mathsf{K}(1^\lambda)$:	
Return $s = (s_1, \ldots, s_\ell) \xleftarrow{\mathsf{r}} \{0,1\}^\ell$.	
$\mathsf{E}(s, m)$:	$\mathsf{D}(s, \mathsf{CT})$:
$\mathsf{hk} \leftarrow \mathsf{HKG}(1^\lambda, \ell)$	$(\mathsf{hk}, H, \mathsf{ct}) \leftarrow \mathsf{CT}$
$h \leftarrow \mathsf{H}(\mathsf{hk}, s)$	$(\mathsf{ct}_{j,\alpha})_{j,\alpha} \leftarrow \mathsf{ct} \oplus \mathsf{PRG}(H(s))$
// Generate random shares $r_1, \ldots, r_\ell$ of m:	$\forall j \in [\ell] : r_j \leftarrow \mathsf{HDec}(\mathsf{hk}, s, \mathsf{ct}_{j,s_j})$
$\forall j \in [\ell - 1] : r_j \xleftarrow{\mathsf{r}} \{0,1\}^\mu$	Return $m \leftarrow \bigoplus_{j \in [\ell]} r_j$.
$r_\ell \leftarrow (\bigoplus_{j \in [\ell-1]} r_j) \oplus m$	
$\forall (j, \alpha) \in [\ell] \times \{0,1\} : \mathsf{ct}_{j,\alpha} \leftarrow \mathsf{HEnc}(\mathsf{hk}, (h, j, \alpha), r_j)$	
$H \xleftarrow{\mathsf{r}} \mathcal{H}$	
$\mathsf{ct} \leftarrow (\mathsf{ct}_{j,\alpha})_{j,\alpha} \oplus \mathsf{PRG}(H(s))$	
Return $\mathsf{CT} \leftarrow (\mathsf{hk}, H, \mathsf{ct})$.	

Figure 4: The proposed SKE scheme SKE. The notation $(\mathsf{ct}_{j,\alpha})_{j,\alpha}$ is the abbreviation for $(\mathsf{ct}_{j,\alpha})_{j \in [\ell], \alpha \in \{0,1\}}$.

7.2 Construction

Let $n = n(\lambda)$ be any polynomial with respect to which we wish our SKE scheme to be passively $\mathcal{P}$-RKA-KDM$^{(n)}$ secure. Let $\mathsf{HE} = (\mathsf{HKG}, \mathsf{H}, \mathsf{HEnc}, \mathsf{HDec})$ be a hash encryption scheme whose plaintext space is $\{0,1\}^\mu$ for some polynomial $\mu = \mu(\lambda)$ and whose ciphertext size is $\ell_c = \ell_c(\lambda)$. Let $\ell = \ell(\lambda)$ be any polynomial satisfying $\ell = 2n\lambda + \omega(\log \lambda)$, and let $L = L(\lambda) = 2\ell \cdot \ell_c$. Finally, let $\mathcal{H} = \{H : \{0,1\}^\ell \to \{0,1\}^\lambda\}$ be a universal hash family, and let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^L$ be a PRG. (The formal definition of a PRG is recalled in Appendix A.4.)

Using these ingredients, we construct our SKE scheme $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ whose secret-key space and message space are $\{0,1\}^\ell$ and $\{0,1\}^\mu$, respectively, as described in Figure 4. Note that the key generation algorithm K of SKE just outputs a secret key $s \in \{0,1\}^\ell$ uniformly at random.

Correctness. The correctness of SKE follows from those of building blocks.

Security. The following theorem guarantees that SKE satisfies passive $\mathcal{P}$-RKA-KDM security.

Theorem 6 *Let $n = n(\lambda)$ and $\ell = \ell(\lambda)$ be any polynomials satisfying $\ell = 2n\lambda + \omega(\log \lambda)$. Assume HE is a secure hash encryption scheme, $\mathcal{H}$ is a universal hash family, and PRG is a secure PRG. Then, SKE is passively $\mathcal{P}$-RKA-KDM$^{(n)}$ secure.*

Outline of the Proof. Before showing the formal proof of Theorem 6, we briefly explain its outline. Letting $\mathcal{A}$ be an adversary attacking the passive $\mathcal{P}$-RKA-KDM$^{(n)}$ security of SKE, we start with the security game in which $\mathcal{A}$ is given a ciphertext CT^i in which a key-dependent message $f^i(s)$ is encrypted under the shifted key $s \oplus \Delta^i$, namely $\mathsf{CT}^i \leftarrow \mathsf{E}(s \oplus \Delta^i, f^i(s))$, for every $i \in [n]$, where f^i and Δ^i are the i-th projection function queried by $\mathcal{A}$ and i-th key shift randomly chosen by the challenger, respectively. Our goal is to gradually change this game and reach the game where $\mathcal{A}$ is given CT^i that encrypts the constant message 0^μ under the shifted key $s \oplus \Delta^i$ for every $i \in [n]$, in such a way that the changes are not noticed by $\mathcal{A}$.

We first change the security game so that an encryption of $f^i(s)$ under the key $s \oplus \Delta^i$ can be simulated without using s except for computing $h^i = \mathsf{H}(\mathsf{hk}^i, s \oplus \Delta^i)$ and $H^i(s \oplus \Delta^i)$, where hk^i is a hash key of HE and H^i is a randomly chosen universal hash function for the i-th ciphertext CT^i. We perform this change by considering how each output bit of f^i is determined. Since f^i

is a projection function, for each output bit of f^i, one of the following three cases always holds: **(i)** it is a copy of some input bit **(ii)** it is a flip of some input bit **(iii)** it is a constant. Based on this fact, by using some statistical arguments and the security of HE, we can complete the change.

Once we complete the above change, for every $i \in [n]$, we can replace $H^i(s \oplus \Delta^i)$ with a uniformly random string by relying on the leftover hash lemma. (A version of the leftover hash lemma we use in the proof is reviewed in Appendix A.5.) This change is justified by the fact that now s has enough entropy from $\mathcal{A}$'s viewpoint since except for $(H^i(s \oplus \Delta^i))_{i\in[n]}$, s is now used to compute only "short" values $(h^i)_{i\in[n]}$, and we set the secret key length ℓ large enough to enjoy the statistical indistinguishability guaranteed by the leftover hash lemma.

Then, we can use the security of PRG since its inputs are now uniformly at random by the previous change. The security of PRG ensures that the third component ct^i of CT^i is distributed independently and uniformly at random, and ct^i can be now viewed as a one-time-pad ciphertext encrypting the HE-ciphertexts $(\mathsf{ct}^i_{j,\alpha})_{j,\alpha}$. Finally, after changing how the third component ct^i in CT^i is generated appropriately, and then undoing the changes made by using the security of PRG and the leftover hash lemma, we reach the final game in which every CT^i encrypts 0^μ under the key $s \oplus \Delta^i$, and complete the security proof.

Proof of Theorem 6. Let $\mathcal{A}$ be any adversary that attacks the passive $\mathcal{P}$-RKA-KDM$^{(n)}$ security of SKE. We proceed the proof via a sequence of games argument using eight games. For every $t \in [8]$, let $\mathtt{T}_t$ be the event that $\mathcal{A}$ outputs 1 in Game t.

In the following, the notations like $(X_{j,\alpha})_{j,\alpha}$ and $(X_j)_j$ are the abbreviations for $(X_{j,\alpha})_{j\in[\ell],\alpha\in\{0,1\}}$ and $(X_j)_{j\in[\ell]}$, respectively.

Game 1: This is the original passive $\mathcal{P}$-RKA-KDM$^{(n)}$ game regarding SKE in which the challenge bit is 1. The detailed description of the game is as follows.

1. The challenger samples $s = (s_1, \ldots, s_\ell) \xleftarrow{\mathsf{r}} \{0,1\}^\ell$ and $\Delta^i \xleftarrow{\mathsf{r}} \{0,1\}^\ell$ for every $i \in [n]$, and sends $(\Delta^i)_{i\in[n]}$ to $\mathcal{A}$.
2. $\mathcal{A}$ sends to the challenger n projection functions $f^1, \ldots, f^n$ whose domain and range are $\{0,1\}^\ell$ and $\{0,1\}^\mu$, respectively. For every $i \in [n]$, the challenger computes CT^i as follows.[19]
   (a) Generate $\mathsf{hk}^i \leftarrow \mathsf{HKG}(1^\lambda, \ell)$.
   (b) Compute $h^i \leftarrow \mathsf{H}(\mathsf{hk}^i, s \oplus \Delta^i)$.
   (c) Pick $r^i_1, \ldots, r^i_{\ell-1} \xleftarrow{\mathsf{r}} \{0,1\}^\mu$ and set $r^i_\ell \leftarrow (\bigoplus_{j\in[\ell-1]} r^i_j) \oplus f^i(s)$.
   (d) For every $j \in [\ell]$ and $\alpha \in \{0,1\}$, compute $\mathsf{ct}^i_{j,\alpha} \leftarrow \mathsf{HEnc}(\mathsf{hk}^i, (h^i, j, \alpha), r^i_j)$.
   (e) Generate $H^i \xleftarrow{\mathsf{r}} \mathcal{H}$.
   (f) Compute $\mathsf{ct}^i \leftarrow (\mathsf{ct}^i_{j,\alpha})_{j,\alpha} \oplus \mathsf{PRG}(H^i(s \oplus \Delta^i))$.
   (g) Set $\mathsf{CT}^i \leftarrow (\mathsf{hk}^i, H^i, \mathsf{ct}^i)$.

   Then, the challenger sends the challenge ciphertexts $(\mathsf{CT}^i)_{i\in[n]}$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Game 2: Same as Game 1, except for how the challenger generates $(r^i_j)_j$ for every $i \in [n]$. In this game, each of the strings $(r^i_j)_j$ is generated in a "bit-by-bit" fashion. To explain it more specifically, let us introduce some notation. For every $k \in [\mu]$, we denote the k-th bit of r^i_j by $r^i_j[k]$, and let $f^i[k] : \{0,1\}^\ell \to \{0,1\}$ denote the function corresponding to

[19] Note that the procedure below is exactly $\mathsf{CT}^i \leftarrow \mathsf{E}(s \oplus \Delta^i, f^i(s))$.

the k-th output bit of $f^i(\cdot)$. Furthermore, we denote the j-th bit of Δ^i by Δ^i_j, for every $j \in [\ell]$.

Then, since each f^i is a projection function, one of the followings holds for every $k \in [\mu]$.

(i) $f^i[k]$ is a copy of the j^*-th input bit for some $j^* \in [\ell]$. (Thus, $f^i[k](s) = s_{j^*}$.)

(ii) $f^i[k]$ is a flip of the j^*-th input bit for some $j^* \in [\ell]$. (Thus, $f^i[k](s) = 1 \oplus s_{j^*}$.)

(iii) $f^i[k]$ is a constant value $\gamma \in \{0,1\}$. (Thus, $f^i[k](s) = \gamma$.)

For every $k \in [\mu]$, depending on which one of the above three options holds, the challenger generates $(r^i_j[k])_j$ as follows.

- In Case (i), the challenger first generates random shares of $\Delta^i_{j^*}$ by picking $w^i_j[k] \xleftarrow{\mathsf{r}} \{0,1\}$ for every $j \in [\ell] \setminus \{j^*\}$ and then setting $w^i_{j^*}[k] := (\bigoplus_{j \in [\ell]\setminus\{j^*\}} w^i_j[k]) \oplus \Delta^i_{j^*}$. Then, for every $j \in [\ell]$, the challenger sets

$$r^i_j[k] \leftarrow \begin{cases} w^i_{j^*}[k] \oplus s_{j^*} \oplus \Delta^i_{j^*} & \text{if } j = j^* \\ w^i_j[k] & \text{otherwise} \end{cases}.$$

  Note that $r^i_{j^*}[k] = (\bigoplus_{j \in [\ell]\setminus\{j^*\}} r^i_j[k]) \oplus s_{j^*}$. Hence, $(r^i_j[k])_j$ is distributed identically to random shares of $s_{j^*} = f^i[k](s)$, which is exactly as in Game 1.

- In Case (ii), the challenger first generates random shares of $1 \oplus \Delta^i_{j^*}$ by picking $w^i_j[k] \xleftarrow{\mathsf{r}} \{0,1\}$ for every $j \in [\ell] \setminus \{j^*\}$ and then setting $w^i_{j^*}[k] := (\bigoplus_{j \in [\ell]\setminus\{j^*\}} w^i_j[k]) \oplus 1 \oplus \Delta^i_{j^*}$. Then, for every $j \in [\ell]$, the challenger sets

$$r^i_j[k] \leftarrow \begin{cases} w^i_{j^*}[k] \oplus s_{j^*} \oplus \Delta^i_{j^*} & \text{if } j = j^* \\ w^i_j[k] & \text{otherwise} \end{cases}.$$

  Note that $r^i_{j^*}[k] = (\bigoplus_{j \in [\ell]\setminus\{j^*\}} r^i_j[k]) \oplus 1 \oplus s_{j^*}$. Hence, $(r^i_j[k])_j$ is distributed identically to random shares of $1 \oplus s_{j^*} = f^i[k](s)$, which is exactly as in Game 1.

- In Case (iii), the challenger picks $r^i_j[k] \xleftarrow{\mathsf{r}} \{0,1\}$ for every $j \in [\ell - 1]$, and then sets $r^i_\ell[k] := (\bigoplus_{j \in [\ell-1]} r^i_j[k]) \oplus \gamma$. Obviously, $(r^i_j[k])_j$ is distributed identically to random shares of $\gamma = f^i[k](s)$, which is exactly as in Game 1.

Then, for every $j \in [\ell]$, r^i_j is defined as $r^i_j := r^i_j[1] \| \dots \| r^i_j[\mu]$.

The above completes how we generate $(r^i_j)_j$ in Game 2. For every $i \in [n]$, the distribution of $(r^i_j)_j$ in Game 2 is exactly the same as that in Game 1, since so is each of their bits, as seen above. Thus, we have $\Pr[\mathsf{T}_1] = \Pr[\mathsf{T}_2]$.

Game 3: Same as Game 2, except for how $(\mathsf{ct}^i_{j,\alpha})_{j,\alpha}$ is generated for every $i \in [n]$. Specifically, in Game 3, each $\mathsf{ct}^i_{j,\alpha}$ is generated by $\mathsf{ct}^i_{j,\alpha} \leftarrow \mathsf{HEnc}(\mathsf{hk}^i, (h^i, j, \alpha), r^i_{j,\alpha})$,[20] where each $r^i_{j,\alpha}$ is defined bit by bit based on which one of Cases (i), (ii), and (iii) above holds for each of the output bits of f^i.

Similarly to Game 2, we denote the k-th bit of $r^i_{j,\alpha}$ by $r^i_{j,\alpha}[k]$, for every $k \in [\mu]$. Then, for every $k \in [\mu]$, $(r^i_{j,\alpha}[k])_{j,\alpha}$ is generated as follows.

[20] In contrast, in Game 2, both $\mathsf{ct}^i_{j,0}$ and $\mathsf{ct}^i_{j,1}$ always encrypt the same value r^i_j for every $j \in [\ell]$.

- In Case (i), the challenger first generates random shares $(w^i_j[k])_j$ of $\Delta^i_{j^*}$ satisfying $\bigoplus_{j\in[\ell]} w^i_j[k] = \Delta^i_{j^*}$ as in Game 2. Then, every $j \in [\ell]$ and $\alpha \in \{0,1\}$, the challenger sets
$$r^i_{j,\alpha}[k] \leftarrow \begin{cases} w^i_{j^*}[k] \oplus \alpha & \text{if } j = j^* \\ w^i_j[k] & \text{otherwise} \end{cases}.$$
Note that for every $(j,\alpha) \neq (j^*, 1\oplus s_{j^*}\oplus\Delta^i_{j^*})$, $r^i_{j,\alpha}[k]$ in Game 3 is generated in exactly the same way as $r^i_j[k]$ in Game 2. The only value that is generated differently from Game 2 is $r^i_{j^*,1\oplus s_{j^*}\oplus\Delta^i_{j^*}}[k]$, which is a flip of $r^i_j[k]$ in Game 2.
- In Case (ii), the challenger first generates random shares $(w^i_j[k])_j$ of $1\oplus\Delta^i_{j^*}$ satisfying $\bigoplus_{j\in[\ell]} w^i_j[k] = 1 \oplus \Delta^i_{j^*}$ as in Game 2. Then, for every $j \in [\ell]$ and $\alpha \in \{0,1\}$, the challenger sets
$$r^i_{j,\alpha}[k] \leftarrow \begin{cases} w^i_{j^*}[k] \oplus \alpha & \text{if } j = j^* \\ w^i_j[k] & \text{otherwise} \end{cases}.$$
As in Case (i), for every $(j,\alpha) \neq (j^*, 1 \oplus s_{j^*} \oplus \Delta^i_{j^*})$, $r^i_{j,\alpha}[k]$ in Game 3 is generated in exactly the same way as $r^i_j[k]$ in Game 2. The only value that is generated differently from Game 2 is $r^i_{j^*,1\oplus s_{j^*}\oplus\Delta^i_{j^*}}[k]$, which is a flip of $r^i_j[k]$ in Game 2.
- In Case (iii), the challenger generates $(r^i_j[k])_j$ as random shares of γ satisfying $\bigoplus_{j\in[\ell]} r^i_j[k] = \gamma$ as in Game 2, and sets $r^i_{j,0}[k] := r^i_j[k]$ and $r^i_{j,1}[k] := r^i_j[k]$. That is, in this case, both $(r^i_{j,0}[k])_j$ and $(r^i_{j,1}[k])_j$ are distributed identically to $(r^i_j[k])_j$ in Game 2.

Then, for every $j \in [\ell]$ and $\alpha \in \{0,1\}$, $r^i_{j,\alpha}$ is defined as $r^i_{j,\alpha} := r^i_{j,\alpha}[1]\| \ldots \|r^i_{j,\alpha}[\mu]$.

The above completes how we generate $(r^i_{j,\alpha})_{j,\alpha}$ in Game 3. In the transition from Game 2 to Game 3, we have changed the distribution of only $\mathsf{ct}^i_{j,1\oplus s_j\oplus\Delta^i_j}$ (but not $\mathsf{ct}^i_{j,s_j\oplus\Delta^i_j}$), for every $j \in [\ell]$ and $i \in [n]$. Moreover, recall that for each $(j,\alpha) \in [\ell] \times \{0,1\}$, $\mathsf{ct}^i_{j,\alpha}$ is generated as $\mathsf{ct}^i_{j,\alpha} \leftarrow \mathsf{HEnc}(\mathsf{hk}^i, (h^i, j, \alpha), r^i_{j,\alpha})$, where h^i is the hash value computed as $h^i = \mathsf{H}(\mathsf{hk}^i, s \oplus \Delta^i)$. Note also that $s_j \oplus \Delta^i_j$ is the j-th bit of $(s \oplus \Delta^i)$, and thus $1 \oplus s_j \oplus \Delta^i_j$ is its flip. Therefore, by the security of HE, we have that the ciphertexts $(\mathsf{ct}^i_{j,1\oplus s_j\oplus\Delta^i_j})_j$ generated in Game 3 are indistinguishable from those generated in Game 2, and thus we can derive $|\Pr[\mathtt{T}_2] - \Pr[\mathtt{T}_3]| = \mathsf{negl}(\lambda)$.

We note that in Game 3, for every $i \in [n]$, $(r^i_{j,\alpha})_{j,\alpha}$ is generated independently of s. Furthermore, $(\mathsf{ct}^i_{j,\alpha})_{j,\alpha}$ can be generated by only using the hash value $h^i = \mathsf{H}(\mathsf{hk}^i, s \oplus \Delta^i)$ (and no more information on s is needed). We are now ready for using the leftover hash lemma to replace each $H^i(s \oplus \Delta^i)$ with a random value, which we do next.

Game 4**:** Same as Game 3, except that for every $i \in [n]$, the challenger replaces $H^i(s \oplus \Delta^i)$ with a uniformly random string $R^i \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$.

As seen above, in Game 3, except for computing $(H^i(s \oplus \Delta^i))_{i\in[n]}$, s is used to generate only $(h^i \leftarrow \mathsf{H}(\mathsf{hk}^i, s \oplus \Delta^i))_{i\in[n]}$. This fact allows us to show $|\Pr[\mathtt{T}_3] - \Pr[\mathtt{T}_4]| = \mathsf{negl}(\lambda)$ based on the leftover hash lemma.

Specifically, we consider the following adversary $\mathcal{A}_{\mathsf{lhl}}$ playing the LHL game (see Lemma 1 in Section A.5 for its definition). $\mathcal{A}_{\mathsf{lhl}}$ picks $\Delta^i \xleftarrow{\mathsf{r}} \{0,1\}^\ell$ and generates $\mathsf{hk}^i \leftarrow \mathsf{HKG}(1^\lambda, \ell)$

for every $i \in [n]$. Next, $\mathcal{A}_{\mathsf{lhl}}$ defines a hash family $\mathcal{H}' = \{H' : \{0,1\}^{\ell} \to (\{0,1\}^{\lambda})^n\}$ and a function $\mathsf{leak} : \{0,1\}^{\ell} \to (\{0,1\}^{\lambda})^n$ as follows:

$$\mathcal{H}' := \left\{ H'(\cdot) := \Big(H^1(\cdot \oplus \Delta^1), \ldots, H^n(\cdot \oplus \Delta^n)\Big) \;\middle|\; H^1, \ldots, H^n \in \mathcal{H} \right\},$$
$$\mathsf{leak}(\cdot) := \Big(\mathsf{H}(\mathsf{hk}^1, \cdot \oplus \Delta^1), \ldots, \mathsf{H}(\mathsf{hk}^n, \cdot \oplus \Delta^n) \Big).$$

Note that $\mathcal{H}'$ is a universal hash family if so is $\mathcal{H}$, and we identify its description H' with n descriptions $(H^i)_{i\in[n]}$. Then, $\mathcal{A}_{\mathsf{lhl}}$ submits the universal hash family $\mathcal{H}'$ and the function leak to the challenger of the LHL game, and receives a tuple $(H', R, \mathsf{leak}(s))$, where $H' = (H^i)_{i\in[n]}$, $R = (R^i)_{i\in[n]}$, and $\mathsf{leak}(s) = (h^i)_{i\in[n]} = (\mathsf{H}(\mathsf{hk}^i, s \oplus \Delta^i))_{i\in[n]}$. (Here, each R^i is either $H^i(s \oplus \Delta^i)$ or a random string in $\{0,1\}^{\lambda}$.) Using these values, $\mathcal{A}_{\mathsf{lhl}}$ executes the remaining procedure for generating $\mathsf{CT}^i = (\mathsf{hk}^i, H^i, \mathsf{ct}^i)$ for every $i \in [n]$, in exactly the same way as done in Game 4, and sends the challenge ciphertexts $(\mathsf{CT}^i)_{i\in[n]}$ to $\mathcal{A}$. (This is possible for $\mathcal{A}_{\mathsf{lhl}}$ since the remaining procedure does not directly use s.) Then, $\mathcal{A}_{\mathsf{lhl}}$ outputs whatever $\mathcal{A}$ outputs.

Note that if the challenge bit for $\mathcal{A}_{\mathsf{lhl}}$ is 1 (resp. 0), $\mathcal{A}_{\mathsf{lhl}}$ perfectly simulates Game 3 (resp. Game 4) for $\mathcal{A}$, and thus we have $\mathsf{Adv}^{\mathsf{lhl}}_{\mathcal{A}_{\mathsf{lhl}}}(\lambda) = |\Pr[\mathtt{T}_3] - \Pr[\mathtt{T}_4]|$. Recall that the input length and output length of $\mathcal{H}'$ are $\ell = 2n \cdot \lambda + \omega(\log \lambda)$ and $n \cdot \lambda$, respectively, and the output length of leak is $n \cdot \lambda$. Hence, by Lemma 1, we have $\mathsf{Adv}^{\mathsf{lhl}}_{\mathcal{A}_{\mathsf{lhl}}} \leq 2^{-\frac{\ell - 2n\cdot\lambda}{2}} = 2^{-\omega(\log\lambda)} = \mathsf{negl}(\lambda)$. Consequently, we have $|\Pr[\mathtt{T}_3] - \Pr[\mathtt{T}_4]| = \mathsf{negl}(\lambda)$.

Game 5: Same as Game 4, except that for every $i \in [n]$, the challenger replaces $\mathsf{PRG}(R^i)$ with a uniformly random string $W^i \in \{0,1\}^L$.

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

Game 6: Same as Game 5, except for how CT^i is generated for every $i \in [n]$. Specifically, in this game, the challenger first generates random shares $(r^i_j)_j$ of 0^{μ}. Then, the challenger generates $\mathsf{ct}^i_{j,\alpha} \leftarrow \mathsf{HEnc}(\mathsf{hk}^i, (h^i, j, \alpha), r^i_j)$ for every $j \in [\ell]$ and $\alpha \in \{0,1\}$. Finally, the challenger computes $\mathsf{ct}^i \leftarrow (\mathsf{ct}^i_{j,\alpha})_{i,j,\alpha} \oplus W^i$, and sets $\mathsf{CT}^i \leftarrow (\mathsf{hk}^i, H^i, \mathsf{ct}^i)$.

Since each $W^i \in \{0,1\}^L$ is chosen uniformly at random, independently of any other values, the change made in this game does not affect $\mathcal{A}$'s view. Thus, we have $\Pr[\mathtt{T}_5] = \Pr[\mathtt{T}_6]$.

Game 7: Same as Game 6, except that we undo the change made between Games 4 and 5. Namely, for each $i \in [n]$, W^i is replaced with $\mathsf{PRG}(R^i)$, where $R^i \xleftarrow{\mathsf{r}} \{0,1\}^{\lambda}$.

By the security of PRG, we have $|\Pr[\mathtt{T}_6] - \Pr[\mathtt{T}_7]| = \mathsf{negl}(\lambda)$.

Game 8: Same as Game 7, except that we undo the change made between Games 3 and 4. Namely, for each $i \in [n]$, R^i is generated as $R^i \leftarrow H^i(s \oplus \Delta^i)$. This game is exactly the same as the original passive $\mathcal{P}$-RKA-KDM$^{(n)}$ game regarding SKE in which the challenge bit is 0.

Applying the leftover hash lemma in the same way as before, we have $|\Pr[\mathtt{T}_7] - \Pr[\mathtt{T}_8]| = \mathsf{negl}(\lambda)$.

From the above arguments, we see that

$$\mathsf{Adv}^{\mathsf{prkakdm}}_{\mathsf{SKE},\mathcal{P},\mathcal{A},n}(\lambda) = |\Pr[\mathtt{T}_1] - \Pr[\mathtt{T}_8]| \leq \sum_{t\in[7]} |\Pr[\mathtt{T}_t] - \Pr[\mathtt{T}_{t+1}]| = \mathsf{negl}(\lambda).$$

Since the choice of $\mathcal{A}$ was arbitrary, SKE is passively $\mathcal{P}$-RKA-KDM$^{(n)}$ secure. □ (**Theorem 6**)

8 Putting It All Together

In this section, we summarize our results.

By combining Theorems 2 and 4, for any polynomial $\mathsf{size} = \mathsf{size}(\lambda)$, a $\mathcal{B}_{\mathsf{size}}$-KDM$^{(1)}$-CCA secure PKE scheme can be constructed from an IND-CPA secure PKE scheme, an IND-CCA secure PKE scheme, a one-time $\mathcal{P}$-KDM secure SKE scheme, and a garbling scheme. From the result by Kitagawa et al. [KMT19], we can realize an IND-CCA secure PKE scheme from an IND-CPA secure PKE scheme and a one-time $\mathcal{P}$-KDM secure PKE scheme. Moreover, a garbling scheme is implied by one-way functions [Yao86], which is in turn implied by an IND-CPA secure PKE scheme. From these, we obtain the following theorem.

Theorem 7 *Assume that there exist an IND-CPA secure PKE scheme and a one-time $\mathcal{P}$-KDM secure SKE scheme that can encrypt messages of length $\Omega(\ell \cdot \lambda)$, where $\ell = \ell(\lambda)$ denotes the secret key length of the SKE scheme. Then, for any polynomial $\mathsf{size} = \mathsf{size}(\lambda)$, there exists a $\mathcal{B}_{\mathsf{size}}$-KDM$^{(1)}$-CCA secure PKE scheme.*

Since both an IND-CPA secure PKE scheme and a one-time $\mathcal{P}$-KDM secure SKE scheme are implied by a $\mathcal{P}$-KDM$^{(1)}$-CPA secure PKE scheme, we obtain the following main theorem.

Theorem 8 (CPA-to-CCA Transformation for KDM Security) *Assume that there exists a $\mathcal{P}$-KDM$^{(1)}$-CPA secure PKE scheme. Then, for any polynomial $\mathsf{size} = \mathsf{size}(\lambda)$, there exists a $\mathcal{B}_{\mathsf{size}}$-KDM$^{(1)}$-CCA secure PKE scheme.*

Similarly to Theorem 7, by combining Theorems 2 and 5, and the previous results [KMT19, Yao86], we also obtain the following theorem.

Theorem 9 *Let $n = n(\lambda)$ be a polynomial. Assume that there exist an IND-CPA secure PKE scheme, and a passively $\mathcal{P}$-RKA-KDM$^{(n)}$ secure SKE scheme that can encrypt messages of length $\Omega(\ell \cdot \lambda)$, where $\ell = \ell(\lambda)$ denotes the secret key length of the SKE scheme, and whose secret key generation algorithm outputs a string that is distributed uniformly over $\{0,1\}^{\ell}$. Then, for any polynomial $\mathsf{size} = \mathsf{size}(\lambda)$, there exists a $\mathcal{B}_{\mathsf{size}}$-KDM$^{(n)}$-CCA secure PKE scheme.*

Note that a passively $\mathcal{P}$-RKA-KDM$^{(n)}$ secure SKE scheme is also a one-time $\mathcal{P}$-KDM secure SKE scheme.

For any polynomials n and μ, we can construct a passively $\mathcal{P}$-RKA-KDM$^{(n)}$ secure SKE scheme whose message space is $\{0,1\}^{\mu}$ based on the LPN assumption [App13]. In addition, from Theorem 6 and the result by Döttling et al. [DGHM18] or Brakerski et al. [BLSV18], for any polynomials n and μ, we can construct a $\mathcal{P}$-RKA-KDM$^{(n)}$ secure SKE scheme whose message space is $\{0,1\}^{\mu}$ based on the CDH assumption. The key generation algorithms of the LPN-/CDH-based constructions output a uniformly random string as a secret key. Since an IND-CPA secure PKE scheme can be constructed based on the LPN and CDH assumptions, we obtain the following corollary.

Corollary 1 *Let $n = n(\lambda)$ and $\mathsf{size} = \mathsf{size}(\lambda)$ be any polynomials. There exists a $\mathcal{B}_{\mathsf{size}}$-KDM$^{(n)}$-CCA secure PKE scheme under either the LPN or CDH assumption.*

Acknowledgement A part of this work was supported by JST CREST Grant Number JPMJCR19F6.

References

[ABHS05] Pedro Adão, Gergei Bana, Jonathan Herzog, and Andre Scedrov. Soundness of formal encryption in the presence of key-cycles. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS 2005*, volume 3679 of *LNCS*, pages 374–396. Springer, Heidelberg, September 2005.

[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.

[App13] Benny Applebaum. Garbling XOR gates "for free" in the standard model. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 162–181. Springer, Heidelberg, March 2013.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer, Heidelberg, August 1998.

[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.

[BHK13] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 398–415. Springer, Heidelberg, August 2013.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Heidelberg, April 2009.

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2019, Part II*, LNCS, pages 3–33. Springer, Heidelberg, May 2019.

[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.

[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.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 455–469. Springer, Heidelberg, August 1997.

[CCH+18] Ran Canetti, Yilei Chen, Justi Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. Fiat-shamir from simpler assumptions. *IACR Cryptology ePrint Archive*, 2018:1004, 2018.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 91–122. Springer, Heidelberg, April / May 2018.

[CCS09] Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 351–368. Springer, Heidelberg, April 2009.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, Heidelberg, May 2004.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, Heidelberg, August 2002.

[CLW18] Ran Canetti, Alex Lombardi, and Daniel Wichs. Non-interactive zero knowledge and correlation intractability from circular-secure FHE. *IACR Cryptology ePrint Archive*, 2018:1248, 2018.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

[Dac14] Dana Dachman-Soled. A black-box construction of a CCA2 encryption scheme from a plaintext aware (sPA1) encryption scheme. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 37–55. Springer, Heidelberg, March 2014.

[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.

[DIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *30th ACM STOC*, pages 141–150. ACM Press, May 1998.

[Döt15] Nico Döttling. Low noise LPN: KDM secure public key encryption and sample amplification. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 604–626. Springer, Heidelberg, March / April 2015.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, Heidelberg, May 2004.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GHV12] David Galindo, Javier Herranz, and Jorge L. Villar. Identity-based encryption with master key-dependent message security and leakage-resilience. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS 2012*, volume 7459 of *LNCS*, pages 627–642. Springer, Heidelberg, September 2012.

[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.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[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.

[HLW12] Susan Hohenberger, Allison B. Lewko, and Brent Waters. Detecting dangerous queries: A new approach for chosen ciphertext security. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 663–681. Springer, Heidelberg, April 2012.

[HO13] Brett Hemenway and Rafail Ostrovsky. Building lossy trapdoor functions from lossy encryption. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 241–260. Springer, Heidelberg, December 2013.

[HU08] Dennis Hofheinz and Dominique Unruh. Towards key-dependent message security in the standard model. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 108–126. Springer, Heidelberg, April 2008.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 581–600. Springer, Heidelberg, March 2006.

[KMHT15] Fuyuki Kitagawa, Takahiro Matsuda, Goichiro Hanaoka, and Keisuke Tanaka. Completeness of single-bit projection-KDM security for public key encryption. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 201–219. Springer, Heidelberg, April 2015.

[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.

[KMT19] Fuyuki Kitagawa, Takahiro Matsuda, and Keisuke Tanaka. Cca security and trapdoor functions via key-dependent-message security. *IACR Cryptology ePrint Archive*, 2019:291, 2019. To appear in CRYPTO 2019.

[KT18a] Fuyuki Kitagawa and Keisuke Tanaka. A framework for achieving KDM-CCA secure public-key encryption. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 127–157. Springer, Heidelberg, December 2018.

[KT18b] Fuyuki Kitagawa and Keisuke Tanaka. Key dependent message security and receiver selective opening security for identity-based encryption. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 32–61. Springer, Heidelberg, March 2018.

[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.

[MH14a] Takahiro Matsuda and Goichiro Hanaoka. Chosen ciphertext security via point obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 95–120. Springer, Heidelberg, February 2014.

[MH14b] Takahiro Matsuda and Goichiro Hanaoka. Chosen ciphertext security via UCE. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 56–76. Springer, Heidelberg, March 2014.

[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.

[MH16] Takahiro Matsuda and Goichiro Hanaoka. Trading plaintext-awareness for simulatability to achieve chosen ciphertext security. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 3–34. Springer, Heidelberg, March 2016.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

[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.

[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.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

[Wee10] Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 314–332. Springer, Heidelberg, August 2010.

[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.

[YYHK16] Takashi Yamakawa, Shota Yamada, Goichiro Hanaoka, and Noboru Kunihiro. Adversary-dependent lossy trapdoor function from hardness of factoring semi-smooth RSA subgroup moduli. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 3–32. Springer, Heidelberg, August 2016.

[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 Recovery from Randomness for PKE

Here, we recall a structural property for a PKE scheme called *recovery from randomness* [KW18], which will be required for the building block PKE scheme used in the construction of a strongly

key-hiding AB-SFE scheme presented in Appendix B. It was shown by Koppula and Waters [KW18] that any IND-CPA secure PKE scheme can be turned into one satisfying this property.

Definition 9 (Recovery from Randomness [KW18]) *Let* $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *be a PKE scheme, and let* $\mathcal{R}$ *denote the randomness space of* Enc*. We say that* PKE *satisfies the* recovery-from-randomness *property if there exists a PPT algorithm* Rec *(called the recovery algorithm) with the following property.*

- Rec *takes a public key* pk*, a ciphertext* ct*, and a randomness* $r \in \mathcal{R}$ *as input. It outputs a message* m *satisfying* $\mathsf{Enc}(\mathsf{pk}, m; r) = \mathsf{ct}$ *if such* m *exists; Otherwise, it outputs* $\bot$*.*

A.2 Attribute-Based Secure Function Evaluation

Attribute-based secure function evaluation (AB-SFE) was introduced by Lombardi et al. [LQR+19a] as the main building block of their reusable DV-NIZK argument system. We review its definition here.

Informally, AB-SFE is a generalization (and simplification) of single-key ABE, and is associated with an efficiently computable function $F : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the ciphertext-attribute and key-attribute spaces, respectively. In AB-SFE, each public/secret key pair $(\mathsf{pk}, \mathsf{sk})$ is associated with a key-attribute $y \in \mathcal{Y}$, and each ciphertext ct is associated with a ciphertext-attribute $x \in \mathcal{X}$, and ct (generated by using pk) can be decrypted by sk if $F(x, y) = 1$; On the other hand, if $F(x, y) = 0$, then ct should hide its underlying message even against an adversary that holds sk.

Definition 10 (Attribute-Based Secure Function Evaluation) *An AB-SFE scheme* ABSFE *for a function* $F : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ *is a four tuple* $(\mathsf{ASetup}, \mathsf{AKG}, \mathsf{AEnc}, \mathsf{ADec})$ *of PPT algorithms.*

- ASetup *is the setup algorithm that takes a security parameter* 1^λ *as input, and outputs a CRS* crs*.*
- AKG *is the key generation algorithm that takes a CRS* crs *and a key-attribute* $y \in \mathcal{Y}$ *as input, and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- Enc *is the encryption algorithm that takes a CRS* crs*, a public key* pk*, a ciphertext-attribute* $x \in \mathcal{X}$*, and a message* m *as input, and outputs a ciphertext* ct*.*
- Dec *is the (deterministic) decryption algorithm that takes a CRS* crs*, a secret key* sk*, a ciphertext-attribute* $x \in \mathcal{X}$*, and a ciphertext* ct *as input, and outputs a message* m *which could be the special invalid symbol* $\bot$*.*

Correctness *We require* $\mathsf{ADec}(\mathsf{crs}, \mathsf{sk}, x, \mathsf{AEnc}(\mathsf{crs}, \mathsf{pk}, x, m)) = m$ *for all CRS* crs *output by* $\mathsf{ASetup}(1^\lambda)$*, all attribute pairs* $(x, y) \in \mathcal{X} \times \mathcal{Y}$ *such that* $F(x, y) = 1$*, all key pairs* $(\mathsf{pk}, \mathsf{sk})$ *output by* $\mathsf{AKG}(\mathsf{crs}, y)$*, and all messages* m*.*

Recovery from Randomness. As in the PKE case, we recall the definition of the recovery-from-randomness property of AB-SFE, formalized by Lombardi et al [LQR+19a] analogously to the PKE case. This property will be required for the underlying AB-SFE scheme in the construction of a strongly key-hiding AB-SFE scheme presented in Appendix B.

Definition 11 (Recovery from Randomness ([KW18, LQR+19a])) *Let* $\mathsf{ABSFE} = (\mathsf{ASetup}, \mathsf{AKG}, \mathsf{AEnc}, \mathsf{ADec})$ *be an AB-SFE scheme for a function* $F : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*, and let* $\mathcal{R}$ *denote the randomness space of* AEnc*. We say that* ABSFE *satisfies the* recovery-from-randomness *property if there exists a PPT algorithm* ARec *(called the recovery algorithm) with the following property.*

- ARec *takes a CRS* crs*, a public key* pk*, a ciphertext* ct*, and a randomness* $r \in \mathcal{R}$ *as input. It outputs* (x, m) *satisfying* $\mathsf{AEnc}(\mathsf{crs}, \mathsf{pk}, x, m; r) = \mathsf{ct}$ *if such a pair* (x, m) *exists; Otherwise, it outputs* $\bot$*.*

Security Notions for AB-SFE. Here, we recall the security definitions for AB-SFE, called strong key-hiding, weak key-hiding, and weak message-hiding, as formalized by Lombardi et al. [LQR+19a].[21]

Definition 12 (Strong/Weak Key-Hiding for AB-SFE) *Let* $\mathsf{ABSFE} = (\mathsf{ASetup}, \mathsf{AKG}, \mathsf{AEnc}, \mathsf{ADec})$ *be an AB-SFE scheme for a function* $F : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*. Let* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ *be a pair of PPT "simulator" algorithms whose syntax is as follows.*

- Sim_1 *takes a security parameter* 1^λ *as input, and outputs a fake CRS* $\overline{\mathsf{crs}}$*, a fake public key* $\overline{\mathsf{pk}}$*, and a trapdoor* td*.*
- Sim_2 *takes a trapdoor* td*, a ciphertext-attribute* $x \in \mathcal{X}$*, a ciphertext* ct*, and a bit* $\beta \in \{0,1\}$ *(which is supposed to be* $F(x, y)$*) as input, and outputs a message* m *(which could be the special invalid symbol* $\bot$*).*

Consider the following key-hiding game between a challenger and an adversary $\mathcal{A}$*.*

1. *First,* $\mathcal{A}$ *chooses the challenge key-attribute* $y \in \mathcal{Y}$ *and sends it to the challenger. Then, the challenger chooses the challenge bit* $b \xleftarrow{\mathsf{r}} \{0,1\}$*. If* $b = 1$*, then the challenger generates* $\mathsf{crs} \leftarrow \mathsf{ASetup}(1^\lambda)$ *and* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{AKG}(\mathsf{crs}, y)$*, and sends* $(\mathsf{crs}, \mathsf{pk})$ *to* $\mathcal{A}$*; Otherwise, the challenger generates* $(\overline{\mathsf{crs}}, \overline{\mathsf{pk}}, \mathsf{td}) \leftarrow \mathsf{Sim}_1(1^\lambda)$*, and sends* $(\overline{\mathsf{crs}}, \overline{\mathsf{pk}})$ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *can adaptively make decryption queries. For each of* $\mathcal{A}$*'s decryption queries* (x, ct)*, if* $b = 1$*, then the challenger responds with* $\mathsf{ADec}(\mathsf{crs}, \mathsf{sk}, x, \mathsf{ct})$*; Otherwise, the challenger responds with* $\mathsf{Sim}_2(\mathsf{td}, x, \mathsf{ct}, F(x, y))$*.*
3. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$*.*

We say that ABSFE *satisfies* strong key-hiding *if there exists a PPT simulator* Sim *such that for all PPT adversaries* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{strong-kh}}_{\mathsf{ABSFE},\mathcal{A},\mathsf{Sim}}(\lambda) := 2 \cdot |\Pr[b = b'] - 1/2| = \mathsf{negl}(\lambda)$*.*

Furthermore, we say that ABSFE *satisfies* weak key-hiding *if there exists a PPT simulator* Sim *such that for all PPT adversaries* $\mathcal{A}$ *that make no decryption queries, we have* $\mathsf{Adv}^{\mathsf{strong-kh}}_{\mathsf{ABSFE},\mathcal{A},\mathsf{Sim}}(\lambda) = \mathsf{negl}(\lambda)$*.*[22]

Definition 13 (Weak Message-Hiding for AB-SFE) *Let* $\mathsf{ABSFE} = (\mathsf{ASetup}, \mathsf{AKG}, \mathsf{AEnc}, \mathsf{ADec})$ *be an AB-SFE scheme for a function* $F : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*. Consider the following weak message-hiding game between a challenger and an adversary* $\mathcal{A}$*.*

[21] As mentioned in Section 4, Lombardi et al. also introduced strong message-hiding. We do not recall its formal definition here since it is not directly relevant to our purpose in this paper.

[22] Since an adversary for weak key-hiding never makes a decryption query, the corresponding simulator Sim can just consist of a single algorithm that takes 1^λ as input and outputs a fake CRS $\overline{\mathsf{crs}}$ and a fake public key $\overline{\mathsf{pk}}$ (and no trapdoor).

1. *First, $\mathcal{A}$ chooses the challenge key-attribute $y \in \mathcal{Y}$ and sends it to the challenger. Then, the challenger generates* $\mathsf{crs} \leftarrow \mathsf{ASetup}(1^\lambda)$ *and* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{AKG}(\mathsf{crs}, y)$*, and sends* $(\mathsf{crs}, \mathsf{pk}, \mathsf{sk})$ *to* $\mathcal{A}$*.*

2. *$\mathcal{A}$ sends the challenge ciphertext-attribute $x \in \mathcal{X}$ such that $F(x, y) = 0$ and the challenge messages (m_0, m_1) of equal length to the challenger. The challenger chooses the challenge bit $b \xleftarrow{r} \{0,1\}$, and then returns the challenge ciphertext* $\mathsf{ct} \leftarrow \mathsf{ABSFE}(\mathsf{crs}, \mathsf{pk}, x, m_b)$ *to* $\mathcal{A}$*.*

3. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

We say that ABSFE *satisfies* weak message-hiding *if for all PPT adversaries $\mathcal{A}$, we have* $\mathsf{Adv}^{\mathsf{weak-mh}}_{\mathsf{ABSFE},\mathcal{A}}(\lambda) := 2 \cdot |\Pr[b = b'] - 1/2| = \mathsf{negl}(\lambda)$.

A.3 Equivocable Commitment

Here, we review the definition of a non-interactive equivocable (bit) commitment scheme [DIO98] in the CRS model (which we will hereafter simply call an equivocable commitment scheme). We will refer to a CRS in an equivocable commitment scheme as a committing key, and denote it by ck.

Definition 14 (Equivocable Commitment) *An equivocable commitment scheme* EQCom *is a four tuple* $(\mathsf{CSetup}, \mathsf{Com}, \mathsf{CSetupEQ}, \mathsf{Equiv})$ *of PPT algorithms.*

- CSetup *is the setup algorithm that takes a security parameter 1^λ as input, and outputs a committing key* ck*.*
- Com *is the committing algorithm that takes a committing key* ck *and a message $m \in \{0,1\}$ as input, and outputs a commitment* com*.*
- CSetupEQ *is the fake setup algorithm that takes a security parameter 1^λ as input, and outputs a fake committing key* $\overline{\mathsf{ck}}$*, a fake commitment* $\overline{\mathsf{com}}$*, and a trapdoor* td*.*
- Equiv *is the equivocation algorithm that takes a trapdoor* td *and a message $m \in \{0,1\}$ as input, and outputs a randomness r.*

We require the following properties for an equivocable commitment scheme EQCom*. Let $\mathcal{R}$ be the randomness space of* Com*.*

Statistical Binding *It holds that*

$$\Pr_{\mathsf{ck} \leftarrow \mathsf{CSetup}(1^\lambda)}\left[\ \exists r_0, r_1 \in \mathcal{R} : \mathsf{Com}(\mathsf{ck}, 0; r_0) = \mathsf{Com}(\mathsf{ck}, 1; r_1)\ \right] = \mathsf{negl}(\lambda).$$

We call a committing key ck erroneous *if there exists a randomness pair $(r_0, r_1) \in \mathcal{R}^2$ for which* $\mathsf{Com}(\mathsf{ck}, 0; r_0) = \mathsf{Com}(\mathsf{ck}, 1; r_1)$ *holds, and otherwise we call* ck non-erroneous*.*

Security *Consider the following security game between a challenger and an adversary $\mathcal{A}$.*

1. *First, $\mathcal{A}$ sends a message $m \in \{0,1\}$ to the challenger. Then, the challenger chooses the challenge bit $b \xleftarrow{r} \{0,1\}$. If $b = 1$, then the challenger generates* $\mathsf{ck} \leftarrow \mathsf{CSetup}(1^\lambda)$*, picks $r \xleftarrow{r} \mathcal{R}$, computes* $\mathsf{com} \leftarrow \mathsf{Com}(\mathsf{ck}, m; r)$*, and then sends* $(\mathsf{ck}, \mathsf{com}, r)$ *to $\mathcal{A}$; Otherwise, the challenger generates* $(\overline{\mathsf{ck}}, \overline{\mathsf{com}}, \mathsf{td}) \leftarrow \mathsf{CSetupEQ}(1^\lambda)$ *and* $r \leftarrow \mathsf{Equiv}(\mathsf{td}, m)$*, and then sends* $(\overline{\mathsf{ck}}, \overline{\mathsf{com}}, r)$ *to $\mathcal{A}$.*
2. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

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

We can realize an equivocable commitment scheme based on one-way functions [DIO98].

A.4 Pseudorandom Generator

We recall the definition of a pseudorandom generator (PRG).

Definition 15 (Pseudorandom Generator) *Let* $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^\ell$ *be an efficiently computable deterministic function where* $\ell = \ell(\lambda) > \lambda$ *is some polynomial. We say that* PRG *is a secure PRG if for all PPT adversaries* $\mathcal{A}$*, we have*

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

We can realize secure PRGs based on one-way functions [HILL99].

A.5 Leftover Hash Lemma

Recall that a hash family $\mathcal{H} := \left\{H : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}\right\}$ is called *universal* if for all distinct inputs $x, x' \in \{0,1\}^{\ell_{\mathsf{in}}}$, we have $\Pr_{H \xleftarrow{r} \mathcal{H}}[H(x) = H(x')] \leq 2^{-\ell_{\mathsf{out}}}$.

Here, we recall the leftover hash lemma [HILL99, DRS04], which will used in the security proof for our construction of an RKA-KDM secure SKE scheme based on a hash encryption scheme in Section 7. For convenience, we formalize it via a game between a challenger and an adversary.

Lemma 1 (Leftover Hash Lemma) *Consider the following LHL game between a challenger and an adversary* $\mathcal{A}$*.*

1. *First, the challenger chooses a bit* $b \xleftarrow{r} \{0,1\}$ *and sends a security parameter* 1^λ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends the description of a universal hash family* $\mathcal{H} := \left\{H : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}\right\}$ *and a function* $\mathsf{leak} : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{leak}}}$ *to the challenger. Then, the challenger generates* $H \xleftarrow{r} \mathcal{H}$*,* $x \xleftarrow{r} \{0,1\}^{\ell_{\mathsf{in}}}$*, and* $R \xleftarrow{r} \{0,1\}^{\ell_{\mathsf{out}}}$*. If* $b = 1$*, the challenger returns* $(H, H(x), \mathsf{leak}(x))$ *to* $\mathcal{A}$*; Otherwise, the challenger returns* $(H, R, \mathsf{leak}(x))$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$*.*

Then, for all computationally unbounded adversaries $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{lhl}}_{\mathcal{A}}(\lambda) := \left|\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]\right| \leq 2^{-\frac{\ell_{\mathsf{in}} - (\ell_{\mathsf{out}} + \ell_{\mathsf{leak}})}{2}}.$$

B Key-Hiding Enhancement for AB-SFE via KDM Security

In this section, we give the formal proof of Theorem 3 stated in Section 4. That is, we formally show how to convert any AB-SFE scheme with weak key-hiding into one with strong key-hiding, using a one-time $\mathcal{P}$-KDM secure SKE scheme. (The formal definition of ABSFE is given in Appendix A.2.)

The transformation preserves the function F with which the underlying AB-SFE scheme is associated, and the weak message-hiding of the underlying scheme. The transformation uses

an IND-CPA secure PKE scheme and an equivocable bit commitment scheme (whose formal definition is given in Appendix A.3) as additional building blocks, both of which exist if there exists an AB-SFE scheme with weak key-hiding and weak message-hiding.

The main structure of the transformation is based on the one by Lombardi et al. [LQR+19a]. Essentially, we replace the hinting PRG used in their transformation with a one-time $\mathcal{P}$-KDM secure SKE scheme by using the technique of Kitagawa et al. [KMT19].

Construction. Let $\mathsf{ABSFE} = (\mathsf{ASetup}, \mathsf{AKG}, \mathsf{AEnc}, \mathsf{ADec})$ be an AB-SFE scheme for an efficiently computable function $F : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, and $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme. We assume that ABSFE and PKE support the recovery-from-randomness property, and we denote their recovery algorithms by ARec and Rec, respectively. (We recall the recovery-from-randomness property of PKE and ABSFE in Appendices A.1 and A.2, respectively.) Furthermore, for notational simplicity (and without loss of generality), we assume that the message spaces of both ABSFE and PKE contain the special symbol $\perp$, and the randomness space of both AEnc and Enc to be $\{0,1\}^\lambda$.[23] Let $\mathsf{EQCom} = (\mathsf{CSetup}, \mathsf{Com}, \mathsf{CSetupEQ}, \mathsf{Equiv})$ be an equivocable bit commitment scheme, and we denote the randomness space of Com by $\mathcal{R}$. We assume that ABSFE and PKE can encrypt elements in $\mathcal{R}$. Let $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be an SKE scheme whose secret key space is $\{0,1\}^\ell$ for some polynomial $\ell = \ell(\lambda)$, and which can encrypt messages of length $\ell \cdot \lambda + \ell_{\mathsf{m}}$ for some polynomial $\ell_{\mathsf{m}} = \ell_{\mathsf{m}}(\lambda)$.

Using these building blocks, we construct another AB-SFE scheme $\mathsf{ABSFE}' = (\mathsf{ASetup}', \mathsf{AKG}', \mathsf{AEnc}', \mathsf{ADec}')$ whose message space is $\{0,1\}^{\ell_{\mathsf{m}}}$, as described in Figure 5.

Correctness. The correctness of ABSFE' follows from that of ABSFE and PKE and their corresponding recovery algorithms ARec and Rec, respectively.

Security. The following theorems guarantee the key-hiding and message-hiding of ABSFE'.

Theorem 10 *Assume* ABSFE *satisfies weak key-hiding and* EQCom *satisfies statistical binding. Then,* ABSFE' *satisfies strong key-hiding.*

Theorem 11 *Assume* ABSFE *satisfies weak message-hiding,* PKE *is IND-CPA secure,* SKE *is one-time $\mathcal{P}$-KDM secure, and* EQCom *is a secure equivocable commitment scheme. Then,* ABSFE' *satisfies weak message-hiding.*

As mentioned earlier, an IND-CPA secure PKE scheme and an equivocable commitment scheme used as building blocks exist if there exists an AB-SFE scheme with weak key-hiding and weak message-hiding. Hence, Theorem 3 follows from the construction of ABSFE' and the combination of Theorems 10 and 11.

The formal proofs of Theorems 10 and 11 are given in Appendices B.1 and B.2, respectively.

Remark 2 We note that ABSFE' satisfies *strong message-hiding* (as defined in [LQR+19a]) if so does the underlying AB-SFE scheme ABSFE. This can be easily inferred from the proof of Theorem 11 given in Appendix B.2, and the proof of [LQR+19a, Theorem 5.13]. We omit the details since it is not directly relevant to our results for KDM-CCA secure PKE in this paper.

[23] These assumptions are without loss of generality when we only encrypt messages of an a-priori fixed length (say, n). The former property can be achieved by putting a prefix indicator bit, say 1, for an ordinary message, and encoding $\perp$ as $0\|0^n$. The latter property can be achieved by using a pseudorandom generator, because it does not destroy the security properties of an AB-SFE scheme and the IND-CPA security of a PKE scheme.

$\mathsf{ASetup}'(1^\lambda)$:	$\mathsf{AKG}'(\mathsf{CRS}, y \in \mathcal{Y})$:
$\mathsf{crs} \leftarrow \mathsf{ASetup}(1^\lambda)$	$(\mathsf{crs}, \mathsf{pk}, (\mathsf{ck}_j)_{j\in[\ell]}) \leftarrow \mathsf{CRS}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$	$(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{AKG}(\mathsf{crs}, y)$
$\forall j \in [\ell] : \mathsf{ck}_j \leftarrow \mathsf{CSetup}(1^\lambda)$	$\mathsf{APK} \leftarrow \mathsf{apk}$
$\mathsf{CRS} \leftarrow (\mathsf{crs}, \mathsf{pk}, (\mathsf{ck}_j)_{j\in[\ell]})$	$\mathsf{ASK} \leftarrow (y, \mathsf{ask})$
Return CRS.	Return $(\mathsf{APK}, \mathsf{ASK})$.
$\mathsf{AEnc}'(\mathsf{CRS}, \mathsf{APK}, x \in \mathcal{X}, m)$:	$\mathsf{ADec}'(\mathsf{CRS}, \mathsf{ASK}, x \in \mathcal{X}, \mathsf{CT})$:
$(\mathsf{crs}, \mathsf{pk}, (\mathsf{ck}_j)_{j\in[\ell]}) \leftarrow \mathsf{CRS}$	$(\mathsf{crs}, \mathsf{pk}, (\mathsf{ck}_j)_{j\in[\ell]}) \leftarrow \mathsf{CRS}$
$\mathsf{apk} \leftarrow \mathsf{APK}$	$(y, \mathsf{ask}) \leftarrow \mathsf{ASK}$
$s = (s_1, \dots, s_\ell) \leftarrow \mathsf{K}(1^\lambda)$	If $F(x, y) = 0$ then return $\bot$.
$r_1^0, \dots r_\ell^0, r_1^1, \dots, r_\ell^1 \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$	$((\mathsf{ct}_j^0, \mathsf{ct}_j^1, \mathsf{com}_j)_{j\in[\ell]}, \mathsf{ct}_{\mathsf{ske}}) \leftarrow \mathsf{CT}$
$\mathsf{ct}_{\mathsf{ske}} \leftarrow \mathsf{E}(s, (r_j^{s_j})_{j\in[\ell]} \| m)$	$\forall j \in [\ell]$:
$\forall j \in [\ell]$:	$\rho_j \leftarrow \mathsf{ADec}(\mathsf{crs}, \mathsf{ask}, x, \mathsf{ct}_j^0)$
$\rho_j \xleftarrow{\mathsf{r}} \mathcal{R}$	$s_j \leftarrow \begin{cases} 0 & \text{if } \rho_j \neq \bot \text{ and } \mathsf{Com}(\mathsf{ck}_j, 0; \rho_j) = \mathsf{com}_j \\ 1 & \text{otherwise} \end{cases}$
$\mathsf{com}_j \leftarrow \mathsf{Com}(\mathsf{ck}_j, s_j; \rho_j)$	$s \leftarrow (s_1 \dots, s_\ell) \in \{0,1\}^\ell$
$M_j^{s_j} \leftarrow \rho_j$	$M \leftarrow \mathsf{D}(s, \mathsf{ct}_{\mathsf{ske}})$
$M_j^{1\oplus s_j} \leftarrow \bot$	If $M = \bot$ then return $\bot$.
$\mathsf{ct}_j^0 \leftarrow \mathsf{AEnc}(\mathsf{crs}, \mathsf{apk}, x, M_j^0; r_j^0)$	Parse M as $((r_j)_{j\in[\ell]}, m) \in (\{0,1\}^\lambda)^\ell \times \{0,1\}^{\ell_{\mathsf{m}}}$.
$\mathsf{ct}_j^1 \leftarrow \mathsf{Enc}(\mathsf{pk}, M_j^1; r_j^1)$	$\forall j \in [\ell]$: If the following check fails then return $\bot$:
$\mathsf{CT} \leftarrow ((\mathsf{ct}_j^0, \mathsf{ct}_j^1, \mathsf{com}_j)_{j\in[\ell]}, \mathsf{ct}_{\mathsf{ske}})$	Case $s_j = 0$:
Return CT.	Check if $\mathsf{ARec}(\mathsf{crs}, \mathsf{apk}, \mathsf{ct}_j^0, r_j) = (x, \rho_j)$.
	Case $s_j = 1$:
	$\rho'_j \leftarrow \mathsf{Rec}(\mathsf{pk}, \mathsf{ct}_j^1, r_j)$
	Check if $\rho'_j \neq \bot$ and $\mathsf{Com}(\mathsf{ck}_j, 1; \rho'_j) = \mathsf{com}_j$.
	Return m.

Figure 5: A transformation converting a weakly key-hiding AB-SFE scheme into a strongly key-hiding one.

B.1 Proof of Theorem 10: Strong Key-Hiding of ABSFE'

Let Sim be the PPT simulator for the weak key-hiding of the underlying AB-SFE scheme ABSFE.[24] Using Sim, we construct a PPT simulator $\mathsf{Sim}' = (\mathsf{Sim}'_1, \mathsf{Sim}'_2)$ for showing the strong key-hiding of ABSFE' as described in Figure 6.

Let $\mathcal{A}$ be an arbitrary PPT adversary that attacks the strong key-hiding of ABSFE'. We consider a sequence of three games, where the first (resp. last) game is the strong key-hiding game with the challenge bit $b = 1$ (resp. $b = 0$). For $t \in [3]$, let $\mathtt{T}_t$ be the event that $\mathcal{A}$ outputs 1 in Game t. We will show that $\mathsf{Adv}^{\mathsf{strong-kh}}_{\mathsf{ABSFE}',\mathcal{A},\mathsf{Sim}'}(\lambda) = |\Pr[\mathtt{T}_1] - \Pr[\mathtt{T}_3]| = \mathsf{negl}(\lambda)$, which will prove the theorem.

Game 1: This is the strong key-hiding game regarding ABSFE' in which $b = 1$.

The detailed description of the game is as follows.

1. $\mathcal{A}$ submits the challenge key-attribute $y \in \mathcal{Y}$ to the challenger. The challenger generates a CRS CRS, a public key APK, a secret key ASK, and a trapdoor td as

[24] As remarked in the footnote in Definition 12, a simulator for weak key-hiding only outputs a pair of a fake CRS and a fake public key (and no trapdoor).

$\mathsf{Sim}'(1^\lambda)$:	$\mathsf{Sim}'_2(\mathsf{td}, x, \mathsf{CT}, \beta \in \{0,1\})$:
$(\overline{\mathsf{crs}}, \overline{\mathsf{apk}}) \leftarrow \mathsf{Sim}(1^\lambda)$	If $\beta = 0$ then return $\bot$.
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$	$(\overline{\mathsf{CRS}}, \overline{\mathsf{apk}}, \mathsf{sk}) \leftarrow \mathsf{td}$
$\forall j \in [\ell] : \mathsf{ck}_j \leftarrow \mathsf{CSetup}(1^\lambda)$	$(\overline{\mathsf{crs}}, \mathsf{pk}, (\mathsf{ck}_j)_{j\in[\ell]}) \leftarrow \overline{\mathsf{CRS}}$
$\overline{\mathsf{CRS}} \leftarrow (\overline{\mathsf{crs}}, \mathsf{pk}, (\mathsf{ck}_j)_{j\in[\ell]})$	$((\mathsf{ct}_j^0, \mathsf{ct}_j^1, \mathsf{com}_j)_{j\in[\ell]}, \mathsf{ct}_{\mathsf{ske}}) \leftarrow \mathsf{CT}$
$\overline{\mathsf{APK}} \leftarrow \overline{\mathsf{apk}}$	$\forall j \in [\ell]$:
$\mathsf{td} \leftarrow (\overline{\mathsf{CRS}}, \overline{\mathsf{apk}}, \mathsf{sk})$	$\rho_j \leftarrow \mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}_j^1)$
Return $(\overline{\mathsf{CRS}}, \overline{\mathsf{APK}}, \mathsf{td})$.	$s_j \leftarrow \begin{cases} 1 & \text{if } \rho_j \neq \bot \text{ and } \mathsf{Com}(\mathsf{ck}_j, 1; \rho_j) = \mathsf{com}_j \\ 0 & \text{otherwise} \end{cases}$
	$s \leftarrow (s_1 \ldots, s_\ell) \in \{0,1\}^\ell$
	$M \leftarrow \mathsf{D}(s, \mathsf{ct}_{\mathsf{ske}})$
	If $M = \bot$ then return $\bot$.
	Parse M as $((r_j)_{j\in[\ell]}, m) \in (\{0,1\}^\lambda)^\ell \times \{0,1\}^{\ell_{\mathsf{m}}}$.
	$\forall j \in [\ell]$: If the following check fails then return $\bot$:
	Case $s_j = 0$:
	$(x'_j, \rho'_j) \leftarrow \mathsf{ARec}(\overline{\mathsf{crs}}, \overline{\mathsf{apk}}, \mathsf{ct}_j^0, r_j)$
	Check if $x'_j = x$ and $\mathsf{Com}(\mathsf{ck}_j, 0; \rho'_j) = \mathsf{com}_j$.
	Case $s_j = 1$:
	Check if $\mathsf{Rec}(\mathsf{pk}, \mathsf{ct}_j^1, r_j) = \rho_j$.
	Return m.

Figure 6: The simulator algorithms used for showing the strong key-hiding of ABSFE'.

follows. (Although td is not used in Game 1, it is well-defined even in Game 1, and defining it here is useful for describing the subsequent games and our analysis.)

(a) Generate $\mathsf{crs} \leftarrow \mathsf{ASetup}(1^\lambda)$.
(b) Generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$.
(c) Generate $\mathsf{ck}_j \leftarrow \mathsf{CSetup}(1^\lambda)$ for every $j \in [\ell]$.
(d) Generate $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{AKG}(\mathsf{crs}, y)$.
(e) Set $\mathsf{CRS} := (\mathsf{crs}, \mathsf{pk}, (\mathsf{ck}_j)_{j\in[\ell]})$, $\mathsf{APK} := \mathsf{apk}$, $\mathsf{ASK} := (y, \mathsf{ask})$, and $\mathsf{td} := (\mathsf{CRS}, \mathsf{apk}, \mathsf{sk})$

The challenger sends $(\mathsf{CRS}, \mathsf{APK})$ to $\mathcal{A}$.

2. $\mathcal{A}$ may adaptively make decryption queries $(x, \mathsf{CT} = ((\mathsf{ct}_j^0, \mathsf{ct}_j^1, \mathsf{com}_j)_{j\in[\ell]}, \mathsf{ct}_{\mathsf{ske}}))$. The challenger responds to each of the queries as follows.
   (a) If $F(x, y) = 0$ then return $\bot$ to $\mathcal{A}$.
   (b) For each $j \in [\ell]$, do the following: Compute $\rho_j \leftarrow \mathsf{ADec}(\mathsf{crs}, \mathsf{ask}, x, \mathsf{ct}_j^0)$. If $\rho_j \neq \bot$ and $\mathsf{Com}(\mathsf{ck}_j, 0; \rho_j) = \mathsf{com}_j$, then set $s_j \leftarrow 0$. Otherwise, set $s_j \leftarrow 1$.
   (c) Set $s \leftarrow (s_1 \ldots, s_\ell) \in \{0,1\}^\ell$.
   (d) Compute $M \leftarrow \mathsf{D}(s, \mathsf{ct}_{\mathsf{ske}})$.
   (e) If $M = \bot$ then return $\bot$ to $\mathcal{A}$. Otherwise, parse M as $((r_j)_{j\in[\ell]}, m) \in (\{0,1\}^\lambda)^\ell \times \{0,1\}^{\ell_{\mathsf{m}}}$.
   (f) For each $j \in [\ell]$, check the following, and return $\bot$ to $\mathcal{A}$ if any of the checks fails:
      - Case $s_j = 0$: Check if $\mathsf{ARec}(\mathsf{crs}, \mathsf{apk}, \mathsf{ct}_j^0, r_j) = (x, \rho_j)$.
      - Case $s_j = 1$: Compute $\rho'_j \leftarrow \mathsf{Rec}(\mathsf{pk}, \mathsf{ct}_j^1, r_j)$ and check if $\rho'_j \neq \bot$ and $\mathsf{Com}(\mathsf{ck}_j, 1; \rho'_j) = \mathsf{com}_j$.
   (g) Return m to $\mathcal{A}$.

3. $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Game 2: Same as Game 1, except that the challenger uses Sim'_2 to answer $\mathcal{A}$'s decryption queries. Namely, for a decryption query (x, CT) asked by $\mathcal{A}$, the challenger responds with $\mathsf{Sim}'_2(\mathsf{td}, x, \mathsf{CT}, F(x,y))$. (Note that crs and apk in $(\mathsf{CRS}, \mathsf{APK}, \mathsf{td})$ are still generated as in Game 1.)

We will later show in Lemma 2 that the statistical binding of the underlying equivocable commitment scheme EQCom implies $|\Pr[\mathtt{T}_1] - \Pr[\mathtt{T}_2]| = \mathsf{negl}(\lambda)$.

Game 3: Same as Game 2, except that the challenger runs $(\overline{\mathsf{crs}}, \overline{\mathsf{apk}}) \leftarrow \mathsf{Sim}(1^\lambda)$, and uses them as the substitutes for crs and apk in $(\mathsf{CRS}, \mathsf{APK}, \mathsf{td})$. Note that this game is exactly the strong key-hiding game regarding ABSFE' in which $b = 0$.

By the weak key-hiding of the underlying AB-SFE scheme ABSFE, we have $|\Pr[\mathtt{T}_2] - \Pr[\mathtt{T}_3]| = \mathsf{negl}(\lambda)$.

It remains to show the following.

Lemma 2 *If* EQCom *is statistically binding, then we have* $|\Pr[\mathtt{T}_1] - \Pr[\mathtt{T}_2]| = \mathsf{negl}(\lambda)$.

Proof of Lemma 2. Note that Game 1 and Game 2 proceed identically unless $\mathcal{A}$ submits a decryption query (x, CT) satisfying $\mathsf{ADec}'(\mathsf{CRS}, \mathsf{ASK}, x, \mathsf{CT}) \neq \mathsf{Sim}'_2(\mathsf{td}, x, \mathsf{CT}, F(x,y))$. We call such a decryption query *bad*.

By the statistical binding of EQCom and the union bound, the probability that some of $(\mathsf{ck}_j)_{j\in[\ell]}$ is erroneous in Game 1 or Game 2 is bounded by $\mathsf{negl}(\lambda)$. Below, we will show that if none of $(\mathsf{ck}_j)_{j\in[\ell]}$ is erroneous, then there do not exist bad decryption queries in Game 1 or Game 2, which will imply the lemma.

To this end, fix $\mathcal{A}$'s challenge key-attribute y, a CRS $\mathsf{CRS} = (\mathsf{crs}, \mathsf{pk}, (\mathsf{ck}_j)_{j\in[\ell]})$ such that all of $(\mathsf{ck}_j)_{j\in[\ell]}$ are non-erroneous, a key pair $(\mathsf{APK}, \mathsf{ASK}) = (\mathsf{apk}, (y, \mathsf{ask}))$, and a trapdoor $\mathsf{td} = (\mathsf{CRS}, \mathsf{apk}, \mathsf{sk})$. Let $(x, \mathsf{CT} = ((\mathsf{ct}_j^0, \mathsf{ct}_j^1, \mathsf{com}_j)_{j\in[\ell]}, \mathsf{ct}_{\mathsf{ske}}))$ be any decryption query submitted by $\mathcal{A}$. If $F(x,y) = 0$, then both ADec' and Sim'_2 output $\perp$. Thus, below we consider the case $F(x,y) = 1$. Let $s = (s_1, \ldots, s_\ell) \in \{0,1\}^\ell$ and $s' = (s'_1, \ldots, s'_\ell) \in \{0,1\}^\ell$ be the s-values computed in ADec' and Sim'_2, respectively. Namely, for each $j \in [\ell]$, we have

$$s_j = \begin{cases} 0 & \text{if } \mathsf{ADec}(\mathsf{crs}, \mathsf{ask}, x, \mathsf{ct}_j^0) = \rho_j \neq \perp \text{ and } \mathsf{Com}(\mathsf{ck}_j, 0; \rho_j) = \mathsf{com}_j \\ 1 & \text{otherwise} \end{cases},$$

$$s'_j = \begin{cases} 1 & \text{if } \mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}_j^1) = \rho_j \neq \perp \text{ and } \mathsf{Com}(\mathsf{ck}_j, 1; \rho_j) = \mathsf{com}_j \\ 0 & \text{otherwise} \end{cases}.$$

In the following, we will consider two cases and show that in either case, the outputs of $\mathsf{ADec}'(\mathsf{CRS}, \mathsf{ASK}, x, \mathsf{CT})$ and $\mathsf{Sim}'_2(\mathsf{td}, x, \mathsf{CT}, F(x,y))$ agree.

Case: $s_j = s'_j$ **holds for all** $j \in [\ell]$**.** In this case, $s = s'$ holds, and ABSFE' and Sim'_2 perform the same set of validity checks on (x, CT) in their entire processes. Thus, their outputs agree (either the least significant ℓ_{m} bits of $\mathsf{D}(s, \mathsf{ct}_{\mathsf{ske}}) = \mathsf{D}(s', \mathsf{ct}_{\mathsf{ske}})$, or $\perp$).

Case: There exists a position $j^* \in [\ell]$ **for which** $s_{j^*} \neq s'_{j^*}$ **holds.** Let $\rho_{j^*} := \mathsf{ADec}(\mathsf{crs}, \mathsf{ask}, x, \mathsf{ct}_{j^*}^0)$ and $\rho'_{j^*} := \mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}_{j^*}^1)$. Consider the following two sub-cases.

- Subcase $(s_{j^*}, s'_{j^*}) = (0,1)$: The condition of this subcase implies $\mathsf{Com}(\mathsf{ck}_{j^*}, 0; \rho_{j^*}) = \mathsf{Com}(\mathsf{ck}_{j^*}, 1; \rho'_{j^*}) = \mathsf{com}_{j^*}$. However, it contradicts the premise that ck_{j^*} is non-erroneous, and thus this sub-case never occurs.

- Subcase $(s_{j^*}, s'_{j^*}) = (1, 0)$: We will show that both ADec' and Sim'_2 return $\perp$ in this case. We first explain why ADec' returns $\perp$: If $\mathsf{D}(s, \mathsf{ct}_{\mathsf{ske}}) = \perp$, then ADec' returns $\perp$ by design; If $\mathsf{D}(s, \mathsf{ct}_{\mathsf{ske}}) = ((r_j)_{j\in[\ell]}, m) \neq \perp$ (for some m), then $s'_{j^*} = 0$ and the correctness of the recovery algorithm Rec imply[25] either
  - (a) $\mathsf{Rec}(\mathsf{pk}, \mathsf{ct}^1_{j^*}, r_{j^*}) = \perp$, or
  - (b) $\mathsf{Rec}(\mathsf{pk}, \mathsf{ct}^1_{j^*}, r_{j^*}) = \rho'_{j^*}$ and $\mathsf{Com}(\mathsf{ck}_{j^*}, 1; \rho'_{j^*}) \neq \mathsf{com}_{j^*}$.

  Note that $s_{j^*} = 1$ means that the validity of the PKE-ciphertext $\mathsf{ct}^1_{j^*}$ is checked in the last step of ADec'. However, "(a) or (b)" implies that the validity check of $\mathsf{ct}^1_{j^*}$ cannot be satisfied. Hence, regardless of whether $\mathsf{D}(s, \mathsf{ct}_{\mathsf{ske}}) = \perp$ or not, ADec' returns $\perp$.

  The explanation on why Sim'_2 returns $\perp$ is symmetric to the above: If $\mathsf{D}(s', \mathsf{ct}_{\mathsf{ske}}) = \perp$, then Sim'_2 returns $\perp$ by design. If $\mathsf{D}(s', \mathsf{ct}_{\mathsf{ske}}) = ((r_j)_{j\in[\ell]}, m) \neq \perp$ (for some m), then $s_{j^*} = 1$ and the correctness of the recovery algorithm ARec imply either
  - (a') $\mathsf{ARec}(\mathsf{crs}, \mathsf{apk}, \mathsf{ct}^0_{j^*}, r_{j^*}) = \perp$,
  - (b') $\mathsf{ARec}(\mathsf{crs}, \mathsf{apk}, \mathsf{ct}^0_{j^*}, r_{j^*}) = (x', \rho) \neq \perp$ for some $x' \neq x$ and ρ, or
  - (c') $\mathsf{ARec}(\mathsf{crs}, \mathsf{apk}, \mathsf{ct}^0_{j^*}, r_{j^*}) = (x, \rho_{j^*}) \neq \perp$ and $\mathsf{Com}(\mathsf{ck}_{j^*}, 0; \rho_{j^*}) \neq \mathsf{com}_{j^*}$.

  Note that $s'_{j^*} = 0$ means that the validity of the AB-SFE-ciphertext $\mathsf{ct}^0_{j^*}$ is checked in the last step of Sim'_2. However, "(a') or (b') or (c')" implies that the validity check of $\mathsf{ct}^0_{j^*}$ cannot be satisfied. Hence, regardless of whether $\mathsf{D}(s', \mathsf{ct}_{\mathsf{ske}}) = \perp$ or not, Sim'_2 returns $\perp$.

As seen above, if none of $(\mathsf{ck}_j)_{j\in[\ell]}$ in CRS is erroneous, then we always have $\mathsf{ADec}'(\mathsf{CRS}, \mathsf{ASK}, x, \mathsf{CT}) = \mathsf{Sim}'_2(\mathsf{td}, x, \mathsf{CT}, F(x, y))$. □ (**Lemma 2**)

From the above arguments, we see that

$$\mathsf{Adv}^{\mathsf{strong-kh}}_{\mathsf{ABSFE}',\mathcal{A},\mathsf{Sim}'}(\lambda) = |\Pr[\mathtt{T}_1] - \Pr[\mathtt{T}_3]| \leq \sum_{t\in[2]} |\Pr[\mathtt{T}_t] - \Pr[\mathtt{T}_{t+1}]| = \mathsf{negl}(\lambda).$$

Since the choice of $\mathcal{A}$ was arbitrary, we can conclude that ABSFE' satisfies strong key-hiding. □ (**Theorem 10**)

B.2 Proof of Theorem 11: Weak Message-Hiding of ABSFE′

Let $\mathcal{A}$ be an arbitrary PPT adversary that attacks the weak message-hiding of ABSFE'. We proceed the proof via a sequence of games argument using five games. For every $t \in [5]$, let $\mathtt{SUC}_t$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit b in Game t.

Game 1: This is the weak message-hiding game regarding ABSFE'. Thus, we have $\mathsf{Adv}^{\mathsf{weak-mh}}_{\mathsf{ABSFE}',\mathcal{A}}(\lambda) = 2 \cdot |\Pr[\mathtt{SUC}_0] - 1/2|$.

The detailed description of the game is as follows.

1. $\mathcal{A}$ submits the challenge key-attribute $y \in \mathcal{Y}$ to the challenger. The challenger generates a CRS CRS, a public key APK, and a secret key ASK as follows.
   (a) Generate $\mathsf{crs} \leftarrow \mathsf{ASetup}(1^\lambda)$.

[25] Note that the correctness of the recovery algorithm Rec guarantees that for any key pair $(\mathsf{pk}, \mathsf{sk})$ output by KG and any ciphertext ct (not necessarily in the image of $\mathsf{Enc}(\mathsf{pk}, \cdot)$), if $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}) = m$ (which could be $\perp$), then we have $\mathsf{Rec}(\mathsf{pk}, \mathsf{ct}, r) \in \{m, \perp\}$ for any r. Hence, since $\rho'_{j^*} = \mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}^1_{j^*})$, we have $\mathsf{Rec}(\mathsf{pk}, \mathsf{ct}^1_{j^*}, r_{j^*}) \in \{\rho'_{j^*}, \perp\}$.

(b) Generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$.
(c) Generate $\mathsf{ck}_j \leftarrow \mathsf{CSetup}(1^\lambda)$ for every $j \in [\ell]$.
(d) Generate $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{AKG}(\mathsf{crs}, y)$.
(e) Set $\mathsf{CRS} := (\mathsf{crs}, \mathsf{pk}, (\mathsf{ck}_j)_{j\in[\ell]})$, $\mathsf{APK} := \mathsf{apk}$, and $\mathsf{ASK} := (y, \mathsf{ask})$.

The challenger sends $(\mathsf{CRS}, \mathsf{APK}, \mathsf{ASK})$ to $\mathcal{A}$.

2. $\mathcal{A}$ sends the challenge ciphertext-attribute $x \in \mathcal{X}$ such that $F(x, y) = 0$ and the challenge messages $(m_0, m_1) \in (\{0,1\}^{\ell_\mathsf{m}})^2$ to the challenger. The challenger chooses the challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$, and computes the challenge ciphertext CT as follows.
   (a) Generate $s = (s_1, \ldots, s_\ell) \leftarrow \mathsf{K}(1^\lambda)$.
   (b) Pick $r_1^0, \ldots, r_\ell^0, r_1^1, \ldots, r_\ell^1 \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$.
   (c) Compute $\mathsf{ct}_\mathsf{ske} \leftarrow \mathsf{E}(s, (r_j^{s_j})_{j\in[\ell]} \| m_b)$.
   (d) Pick $\rho_j \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and compute $\mathsf{com}_j \leftarrow \mathsf{Com}(\mathsf{ck}_j, s_j; \rho_j)$ for each $j \in [\ell]$.
   (e) For each $j \in [\ell]$, do the following: Set $M_j^{s_j} \leftarrow \rho_j$ and $M_j^{1\oplus s_j} \leftarrow \bot$. Then, compute $\mathsf{ct}_j^0 \leftarrow \mathsf{AEnc}(\mathsf{crs}, \mathsf{apk}, x, M_j^0; r_j^0)$ and $\mathsf{ct}_j^1 \leftarrow \mathsf{Enc}(\mathsf{pk}, M_j^1; r_j^1)$.
   (f) Set $\mathsf{CT} := ((\mathsf{ct}_j^0, \mathsf{ct}_j^1, \mathsf{com}_j)_{j\in[\ell]}, \mathsf{ct}_\mathsf{ske})$.

   The challenger sends CT to $\mathcal{A}$.
3. $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Game 2: Same as Game 1, except that the challenger uses the fake setup and equivocation algorithms of EQCom for generating ck_j, com_j, and ρ_j for all $j \in [\ell]$. That is, for each $j \in [\ell]$, when generating ck_j, the challenger runs $(\overline{\mathsf{ck}}_j, \overline{\mathsf{com}}_j, \mathsf{td}_j) \leftarrow \mathsf{CSetupEQ}(1^\lambda)$, and uses $\overline{\mathsf{ck}}_j$ as the substitute for ck_j. Furthermore, when generating the challenge ciphertext CT, the challenger runs $\rho_j \leftarrow \mathsf{Equiv}(\mathsf{td}, s_j)$, and uses this ρ_j and $\overline{\mathsf{com}}_j$ that it has already generated as ρ_j and com_j, respectively, for all $j \in [\ell]$.

By the security of the underlying equivocable commitment scheme EQCom, we have $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| = \mathsf{negl}(\lambda)$.

Game 3: Same as Game 2, except that when generating the challenge ciphertext, M_j^1's for positions $j \in [\ell]$ with $s_j = 0$, are generated as $M_j^1 \leftarrow \mathsf{Equiv}(\mathsf{td}_j, 1)$, instead of $M_j^1 \leftarrow \bot$.

Note that the randomness for generating ct_j^1 for positions $j \in [\ell]$ with $s_j = 0$ are not needed for generating the remaining components in CT. Hence, by the IND-CPA security of the underlying PKE scheme PKE, we have $|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]| = \mathsf{negl}(\lambda)$.

Game 4: Same as Game 3, except that when generating the challenge ciphertext, M_j^0's for positions $j \in [\ell]$ with $s_j = 1$, are generated as $M_j^0 \leftarrow \mathsf{Equiv}(\mathsf{td}_j, 0)$, instead of $M_j^0 \leftarrow \bot$.

Note that the randomness for generating ct_j^0 for positions $j \in [\ell]$ with $s_j = 1$ are not needed for generating the remaining components in CT, and in the security game for the weak-hiding of the underlying AB-SFE scheme ABSFE, a reduction algorithm is given a secret key ask from the challenger. Hence, by the weak message-hiding of the underlying AB-SFE scheme ABSFE and $F(x, y) = 0$, we have $|\Pr[\mathtt{SUC}_3] - \Pr[\mathtt{SUC}_4]| = \mathsf{negl}(\lambda)$.

Note that in Game 4, we have $\mathsf{ct}_j^0 \leftarrow \mathsf{AEnc}(\mathsf{crs}, \mathsf{apk}, x, M_j^0; r_j^0)$ and $\mathsf{ct}_j^1 \leftarrow \mathsf{Enc}(\mathsf{pk}, M_j^1; r_j^1)$ for all $j \in [\ell]$, where $M_j^\alpha \leftarrow \mathsf{Equiv}(\mathsf{td}_j, \alpha)$ for all $(j, \alpha) \in [\ell] \times \{0,1\}$. Hence, the components $(\mathsf{ct}_j^0, \mathsf{ct}_j^1, \mathsf{com}_j)_{j\in[\ell]}$ in CT become independent of $s = (s_1, \ldots, s_\ell)$. This means that the only component that is still dependent on s in CT is ct_ske, which is generated as $\mathsf{ct}_\mathsf{ske} \leftarrow \mathsf{E}(s, (r_j^{s_j})_{j\in[\ell]} \| m_b)$.

Game 5**:** Same as Game 4, except that the information on m_b is eliminated from $\mathsf{ct}_{\mathsf{ske}}$ by generating it as $\mathsf{ct}_{\mathsf{ske}} \leftarrow \mathsf{E}(s, 0^{\ell \cdot \lambda + \ell_{\mathsf{m}}})$. Since CT becomes truly independent of m_b, we have $\Pr[\mathtt{SUC}_5] = 1/2$.

Finally, we argue that $|\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| = \mathsf{negl}(\lambda)$ holds due to the one-time $\mathcal{P}$-KDM security of the underlying SKE scheme SKE. Specifically, consider the function $f : \{0,1\}^{\ell} \to \{0,1\}^{\ell \cdot \lambda + \ell_{\mathsf{m}}}$ that has $(r_j^{\alpha})_{j \in [\ell], \alpha \in \{0,1\}} \| m_b$ hard-wired, and on input $s = (s_1, \ldots, s_\ell)$ outputs $(r_j^{s_j})_{j \in [\ell]} \| m_b$. Note that f is a projection function. Hence, we can straightforwardly construct a reduction algorithm that simulates Game 4 or Game 5 for $\mathcal{A}$ (depending on the reduction's challenge bit) using the function f as its challenge KDM-encryption query, and outputs 1 if and only if $\mathcal{A}$ succeeds in guessing the challenge bit. Such a reduction algorithm has advantage exactly $|\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]|$, and thus we can derive $|\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| = \mathsf{negl}(\lambda)$.

From the above arguments, we see that

$$\begin{aligned}
\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{weak-mh}}_{\mathsf{ABSFE}',\mathcal{A}}(\lambda) &= \left| \Pr[\mathtt{SUC}_1] - \frac{1}{2} \right| \\
&\leq \sum_{t \in [4]} |\Pr[\mathtt{SUC}_t] - \Pr[\mathtt{SUC}_{t+1}]| + \left| \Pr[\mathtt{SUC}_5] - \frac{1}{2} \right| \\
&= \mathsf{negl}(\lambda).
\end{aligned}$$

Since the choice of $\mathcal{A}$ was arbitrary, we can conclude that ABSFE' satisfies weak message-hiding. □ (**Theorem 11**)