Leakage-Resilient Key-Dependent Message Secure Encryption Schemes

Dhairya Gupta[1], Mahesh Sreekumar Rajasree[1], and Harihar Swaminathan[1]

Indian Institute of Technology, Delhi, India
dhairyagupta.211@gmail.com, srmahesh1994@gmail.com,
hariharswaminathan@gmail.com

Abstract. We introduce a new security notion called *leakage-resilient key-dependent message (LR-KDM)* security, which combines both leakage-resilience (LR) and key-dependent message (KDM) security in a unified framework. Leakage-resilient cryptography protects against side-channel attacks that leak partial information about secret keys, while KDM security ensures encryption remains secure even when messages are functions of the secret key. Although both notions have been studied extensively, their combination has not been explored. In this paper,

1. We demonstrate that there exist encryption schemes that are LR secure and KDM secure but do not achieve LR-KDM security.
2. We propose a construction for LR-KDM encryption based on a novel primitive called *leakage-resilient homomorphic* hash proof system, which could be of independent interest.
3. We prove that the encryption scheme proposed by Brakerski *et al.* (EUROCRYPT 2018) satisfies LR-KDM security.
4. We show that the KDM amplification techniques proposed by Waters and Wichs (CRYPTO 2023) and Applebaum (EUROCRYPT 2011) can be adapted in the LR-KDM setting.

Keywords: leakage · key-dependent message · encryption.

1 Introduction

Cryptographic security has traditionally relied on the assumption that secret keys remain hidden from adversaries and that the messages being encrypted are independent of the keys. However, with the advancement of side-channel attacks, these assumptions are often unrealistic in practice. Side-channel attacks exploit physical leakage, such as power consumption [34], electromagnetic radiation [1], or timing information [35], to extract information about the secret keys. This has motivated the development of leakage-resilient (LR) cryptography, where the goal is to ensure security even when some information about the secret key is leaked.

Canetti *et al.* [16] and Dodis, Sahai and Smith [21] were among the first to study leakage-resilient encryption scheme. However, their results were limited to leakage that consisted of subsets of bits of the stored secret, rather than more

general functions of it. The works of Dziembowski [22] and Di Crescenzo, Lipton and Walfish [17] focused on arbitrary leakage functions where adversary can only obtain a fixed number of bits. Akavia, Goldwasser and Vaikuntanathan [2] considered arbitrary leakage in which the amount of leakage is not an absolute value but rather is expressed as a function of the secret-key size. This was followed by several works [39, 24, 19, 14, 25, 23, 29, 40, 28, 8] that prioritised the leakage-rate, i.e., the ratio of the bits that is allowed to be leaked to the size of the secret key. Brakerski *et al.* [12] and Dodis *et al.* [18] expanded this model by considering continual leakage models, which allow adversaries to repeatedly learn information about the secret key in a bounded manner over time. This opened the door for more sophisticated constructions, such as the continual-leakage resilient encryption schemes proposed by Brakerski *et al.* [12] and Dodis *et al.* [18], and the works by Lewko, Lewko and Waters [37], which allowed for encryption in the face of continuous side-channel attacks. Recent efforts in the quantum setting include [15, 3].

While leakage-resilient cryptography addresses the issue of key leakage, another major challenge arises when adversaries can influence the messages being encrypted, particularly when these messages are dependent on the secret key itself. This scenario is captured by the notion of key-dependent message (KDM) security. KDM security ensures that even if the adversary can encrypt messages that are functions of the secret key (for example, encrypting the key itself), they gain no useful information about the key or the underlying plaintexts.

The first formalization of KDM security was introduced by Black, Rogaway, and Shrimpton [9] in the context of public-key encryption, where the challenge was to provide security against adversaries who can choose messages dependent on the secret key. The difficulty of this problem stems from the fact that most encryption schemes break down when such correlations exist between the key and the message. Boneh, Halevi, Hamburg, and Ostrovsky [10] developed the first KDM-secure public-key encryption technique using the decisional Diffie-Hellman (DDH) assumption for affine functions. Their study revealed that KDM security may be achieved using ordinary cryptographic assumptions.

Following this, several constructions were introduced to achieve KDM security for various classes of functions. For instance, Applebaum *et al.* [5] explored KDM security based on the Learning with Errors (LWE) assumption, while the work of Barak, Haitner, Hofheinz and Ishai [7] achieved KDM security for bounded sized circuit under DDH assumption. There are several works in this area [6, 4, 38, 42, 32, 30, 31, 33, 41, 36]. The importance of KDM security cannot be overstated, especially in scenarios where circular encryption or key-dependent operations are unavoidable, such as in encrypted file systems and certain network protocols.

Dodis *et al.* [20] combined circular security with leakage resilience to build updatable public key encryption from standard assumptions. Beyond this, to the best of our knowledge, no other study has systematically combined LR and KDM security into a single cryptographic scheme, despite various designs that achieve both independently [10, 11, 27, 13]. In this work, we introduce leakage-

resilient key-dependent message security and begin the investigation into this security model.

1.1 Our Results

We outline the various results in this paper.

- **Separation Results:** We show that there exists a PKE scheme which is leakage-resilient and secure against KDM attacks separately, but is not LR-KDM secure.
- **LR-KDM Secure Schemes from Standard Assumptions:** We formally define leakage-resilient version of *homomorphic hash proof system* and use it to construct a 1-ciphertext LR-KDM scheme. We also show that the construction given by Brakerski *et al.* [13] that is leakage-resilient and key-dependent message secure is also LR-KDM secure for affine functions with leakage-rate 1.
- **LR-KDM Amplification:** We show that the amplification given by Waters and Wichs [41] and Applebaum [4] can also be adapted in the LR-KDM setting to give a construction for LR-KDM secure public key encryption scheme for the class of circuits assuming the existence of LR-KDM SKE for the class for projection functions. Therefore, amplifying the LR-KDM schemes from batch encryption gives LR-KDM secure for bounded-sized circuit functions with leakage-rate 1 from the hardness of $\mathsf{CDH}, \mathsf{LWE}, \mathsf{LPN}$.

1.2 Related Works

We first highlight a few prominent works that have given constructions that are separately leakage-resilient and key-dependent message secure. Naor and Segev [39] observed that the circular-KDM secure scheme of Boneh *et al.* [10] is already leakage-resilient under the DDH assumption. Brakerski and Goldwasser [11] demonstrated how to construct schemes with $1 - o(1)$ leakage-rate and affine-KDM security based on the Quadratic Residuosity (QR) and Decisional Composite Residuosity (DCR). Hajiabadi, Kapron and Srinivasan [27] developed $1 - o(1)$ leakage-rate and projective-KDM secure schemes using homomorphic hash proof systems. Brakerski *et al.* [13] used a novel tool called batch encryption to design $1 - o(1)$ leakage-rate and affine-KDM secure schemes based on DDH, Learning with Parity (LPN), and other standard assumptions.

To the best of our knowledge, the only work that combines key-dependent message (KDM) security with leakage-resilience is by Dodis *et al.* [20], which constructs updatable public key encryption from standard assumptions. However, we emphasize that the security definition in their work differs from the one presented in this paper. For more details, see Remark 1.

1.3 Open Problem

As this is the first work to address leakage-resilience and key-dependent message (KDM) security simultaneously, it opens up several directions for further research. We outline a few key open problems:

1. **Multi-Key LR-KDM Security:** In this paper, we focus on the *single-key* version of LR-KDM security, where the security game involves a single pair of public-secret keys. A natural extension is the *multi-key* version, where the adversary interacts with multiple pairs of public-secret keys $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[k]}$. In the challenge phase, the adversary could select functions from the class $f : \mathcal{SK}^k \to \mathcal{M}$ and receive either an encryption of $\mathbf{0}$ or $f(\{\mathsf{sk}_i\}_i)$. In the leakage phase, two variants are possible: the adversary could send either a single leakage function $h : \mathcal{SK}^k \to \{0,1\}^\ell$ and receive $h(\{\mathsf{sk}_i\}_i)$, or k separate leakage functions $h_i : \mathcal{SK} \to \{0,1\}^\ell$ and receive $\{h_j(\mathsf{sk}_i)_i\}_j$. These variations define different levels of security in the multi-key LR-KDM setting, which remain unexplored.
2. **LR-KDM Security under Chosen-Ciphertext Attacks (CCA):** Another interesting direction is to investigate LR-KDM security in the presence of chosen-ciphertext attacks (CCA). While there has been significant work on constructing CCA-LR secure and CCA-KDM secure encryption schemes, as well as transformations from CPA-secure to CCA-secure KDM schemes, the landscape of CCA-secure LR-KDM encryption remains largely unexplored. This offers rich opportunities for exploration and further development in this area.

2 Technical Overview

In this section, we present an overview of the results presented in this paper.

2.1 Separation Results

We begin by demonstrating that the notion of leakage-resilient key-dependent message (LR-KDM) security is inherently *non-trivial*. Specifically, an encryption scheme that is independently leakage-resilient (LR) and key-dependent message (KDM) secure does not immediately imply that it is LR-KDM secure. Whether or not such a separation between these security notions must exist is not immediately clear, and it requires careful analysis to establish.

To illustrate this separation, we construct a symmetric key encryption (SKE) scheme that is independently LR secure and KDM secure but fails to achieve LR-KDM security. Let SKE' represent an SKE scheme that is both LR secure and circular-KDM secure. Circular-KDM security means that the scheme remains secure even if the ciphertext encrypts the secret key itself. From this base scheme, we construct a new SKE scheme SKE by combining SKE' with a pseudorandom function (PRF).

In the construction of SKE, the secret key sk consists of two components: k, which is the key for the PRF, and $\mathsf{ske.sk}$, which is a randomly generated key for the encryption scheme SKE'. The encryption for a message m proceeds as follows: first, it checks if m is equal to $\mathsf{ske.sk}$. If so, it computes $c_0 = \mathsf{PRF.Eval}(k, \mathbf{1})$; otherwise, it computes $c_0 = \mathsf{PRF.Eval}(k, \mathbf{0})$. It then generates $c_1 \leftarrow \mathsf{SKE}'.\mathsf{Enc}(\mathsf{ske.sk}, m)$ and outputs the ciphertext $\mathsf{ct} = (c_0, c_1)$. Decryption is

straightforward: the scheme returns $\mathsf{SKE}'.\mathsf{Dec}(\mathsf{ske.sk}, c_1)$, while the first component c_0 is ignored.

We first argue that SKE retains leakage resilience. Suppose there is an adversary $\mathcal{A}$ capable of breaking the LR security of SKE. We can then construct another adversary $\mathcal{B}$ that breaks the LR security of the underlying scheme SKE'. Adversary $\mathcal{B}$ generates the PRF key k independently and can simulate the LR security game for SKE'. For any leakage function $f(k, \mathsf{ske.sk})$ that $\mathcal{A}$ expects, $\mathcal{B}$ hardcodes the value of k into f and queries its own challenger to receive the appropriate leakage. When $\mathcal{A}$ sends messages m_0 and m_1, $\mathcal{B}$ forwards them to its challenger and receives the challenge ciphertext c_1^*. It then computes $c_0 = \mathsf{PRF}.\mathsf{Eval}(k, \mathbf{0})$ and sends the ciphertext $\mathsf{ct}^* = (c_0, c_1^*)$ to $\mathcal{A}$. Since, with very high probability, neither m_0 nor m_1 equals $\mathsf{ske.sk}$, the ciphertext ct^* is correctly formed. Therefore, $\mathcal{B}$'s probability of winning is negligibly close to $\mathcal{A}$'s probability of success, ensuring that SKE is leakage-resilient.

Next, we argue that SKE is KDM secure with respect to the function $f(k, \mathsf{ske.sk}) = \mathsf{ske.sk}$, meaning that the scheme should remain secure when encrypting the secret key. The key observation here is that the second part of the KDM ciphertext, $c_1 \leftarrow \mathsf{SKE}.\mathsf{Enc}(\mathsf{ske.sk}, f(k, \mathsf{ske.sk}))$, depends only on $\mathsf{ske.sk}$ and not on the PRF key k. By leveraging the security of the underlying PRF (see Section 3.1), we can replace the value of c_0 with a truly random value without the adversary detecting the change. It follows that any adversary $\mathcal{A}$ that breaks the KDM security of SKE using this modified ciphertext could be transformed into an adversary that breaks the KDM security of the base scheme SKE'.

Finally, we demonstrate that despite being both LR secure and KDM secure, the scheme SKE fails to achieve LR-KDM security. We parameterize the scheme so that the size of the PRF key k is smaller than the allowed leakage but still larger than the security parameter. In fact, using the GGM construction [26], we can construct PRFs where the size of the key is equivalent to the security parameter. The adversary can exploit this by leaking the entire PRF key k. Once the adversary receives the challenge ciphertext $\mathsf{ct}^* = (c_0, c_1)$, it checks whether $c_0 = \mathsf{PRF}.\mathsf{Eval}(k, \mathbf{0})$. This allows the adversary to distinguish between an encryption of a zero message and an encryption of $f(\mathsf{ske.sk})$, effectively breaking the LR-KDM security. Further details on this separation are provided in Section 4.1.

2.2 LR-KDM Secure Schemes from Standard Assumptions

To construct a leakage-resilient key-dependent message (LR-KDM) secure public key encryption (PKE) scheme, we introduce the concept of a leakage-resilient homomorphic hash proof system (HPS). Wee [42] originally defined homomorphic hash proof systems and demonstrated their use in constructing KDM-secure public key encryption. By enhancing this primitive to be leakage-resilient, we aim to achieve 1-ciphertext LR-KDM security (i.e., the adversary obtains only a single challenge ciphertext) for public key encryption schemes.

An HPS is typically associated with a language $\mathcal{L}$ that satisfies the property of computational indistinguishability: it should be hard to distinguish whether a given string belongs to the language or not. In addition to this language, an HPS

comprises three algorithms: setup, encapsulation, and decapsulation. The setup algorithm generates a public key pk and a secret key sk. The encapsulation algorithm takes as input the public key pk, an element $x \in \mathcal{L}$, and a corresponding witness w (demonstrating that $x \in \mathcal{L}$). It then outputs a string k. On the other hand, the decapsulation algorithm takes as input the secret key sk and a string x and produces the same string k. The correctness of the system ensures that if the encapsulation and decapsulation algorithms are run on the same input x, they will output the same value k.

Moreover, HPSs must satisfy the *smoothness* property, which guarantees that the decapsulation output is statistically close to a uniform distribution if the input string x does not belong to the language $\mathcal{L}$. Another crucial feature of homomorphic HPSs is that the underlying language supports a group structure, and the decapsulation algorithm exhibits homomorphic properties, meaning that it preserves group operations over the input.

Wee [42] provided a construction of a homomorphic HPS based on the d-Linear (d-Lin) assumption. Let G be a cyclic group of prime order with generator g. The language $\mathcal{L}$ consists of elements of the form $x = g^{r^\top P}$, where $P \leftarrow \mathbb{Z}_p^{d \times k}$ is a random matrix, and $r \leftarrow \mathbb{Z}_q^d$ is a vector that serves as the witness for x. The public key contains the term g^{Ps}, where $s \leftarrow \mathbb{Z}_q^k$, and the secret key is the vector s. On input a public key g^{Ps} and element along with its witness $((g^{r^\top P}), r)$, the encapsulation algorithm outputs $g^{r^\top Ps}$. The decapsulation algorithm outputs the same using the secret key s and element $g^{r^\top P}$. The homomorphic property of this scheme is evident from the group operations over G.

To extend this to leakage-resilient homomorphic hash proof systems, we require that the smoothness property holds even in the presence of partial leakage of the secret key sk. We can show that the above construction remains leakage-resilient by appealing to the Leakage-Hiding Lemma (LHL). This lemma states that for a randomly chosen matrix A, a random vector s, and some leakage function $f(s)$, the distribution of $(A, As, f(s))$ is statistically close to $(A, t, f(s))$, where t is uniformly random. This property ensures that the system's security degrades gracefully even in the presence of bounded leakage. For a more detailed discussion on leakage-resilient HPS and LR-KDM constructions from LR-HPS, refer to Section 5.

Following the definition of leakage-resilient homomorphic HPS, we turn our attention to encryption schemes. Specifically, we show that the encryption scheme proposed by Brakerski *et al.* [13], which leverages *batch encryption*, is both leakage-resilient and KDM secure and can be extended to satisfy LR-KDM security. The core idea of batch encryption is to encrypt multiple messages at once, which allows for efficient handling of key-dependent messages and leakage. Our proof that this scheme is LR-KDM secure closely follows the one provided by the original authors. We selected this particular scheme due to its simplicity, though other candidates for LR and KDM secure encryption could also have been explored. For more detailed information on the LR-KDM secure scheme from batch encryption, see Section 6.

2.3 LR-KDM Amplification

Waters and Wichs [41] introduced a novel construction of a key-dependent message (KDM) secure public key encryption (PKE) scheme for arbitrary circuits, using a combination of two core assumptions. Their approach begins with a public key encryption scheme that is semantically secure under chosen plaintext attacks (CPA-secure) and relies on the existence of a symmetric key encryption (SKE) scheme that is secure against circular-KDM attacks. Essentially, their construction leverages the circular-KDM security of the underlying symmetric encryption scheme to "amplify" the security from circular-KDM to circuit-KDM, thereby achieving KDM security for more complex classes of functions, specifically arbitrary circuits.

We build upon their work by introducing a similar amplification for LR-KDM security. In our approach, we replace the CPA-secure PKE scheme in the Waters-Wichs construction with a leakage-resilient PKE scheme. Simultaneously, we substitute the circular-KDM secure SKE assumption with a leakage-resilient circular-KDM secure SKE scheme. This adjustment allows us to achieve a leakage-resilient KDM secure PKE scheme that is secure for circuits, extending the same amplification technique introduced by Waters and Wichs to the leakage-resilience setting.

However, this amplification does not come without cost. One significant trade-off that arises in our construction is related to the amount of leakage that can be tolerated. Specifically, as we extend the security from circular-KDM to circuit-KDM, the amount of leakage that the system can handle becomes more constrained. For more details, see Section 7.

To address the disadvantage of leakage degradation, we apply Applebaum's amplification [4], which starts with a stronger assumption: projection-KDM secure PKE. As we show in Section 8, by starting with a projection-LR-KDM secure PKE scheme, we can amplify it to circuit-LR-KDM security without any loss in the leakage rate. This approach allows us to achieve strong security without the trade-off of reduced leakage tolerance.

3 Preliminaries

Throughout this paper, we will use λ to denote the security parameter and $\mathrm{negl}(\cdot)$ to denote a negligible function in the input. We will use the short-hand notation PPT for "probabilistic polynomial time". For any finite set X, $x \leftarrow X$ denotes the process of picking an element x from X uniformly at random. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from the distribution $\mathcal{D}$. For any natural number $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For any language $\mathcal{L} \subseteq \{0,1\}^*$, we use the notation $\overline{\mathcal{L}}$ to denote the set of element not in the language $\mathcal{L}$.

3.1 Pseudorandom Functions

A family of pseudorandom functions $\mathsf{PRF} = (\mathsf{Setup}, \mathsf{Eval})$ with key space $\{\mathcal{K}_\lambda\}_\lambda$, input space $\{\mathcal{X}_\lambda\}_\lambda$ and output space $\{\mathcal{Y}_\lambda\}_\lambda$ consists of the following algorithms.

- $\mathsf{Setup}(1^\lambda)$: The key generation algorithm is a randomized algorithm that takes as input the security parameter 1^λ and outputs a key $k \in \mathcal{K}_\lambda$.
- $\mathsf{Eval}(k, x)$: The evaluation algorithm is a deterministic algorithm that takes as input a key $k \in \mathcal{K}_\lambda$ and $x \in \mathcal{X}_\lambda$ and outputs $y \in \mathcal{Y}_\lambda$.

Definition 1. *A PRF scheme* PRF *is secure if for all PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,*

$$|\Pr[\mathcal{A}^{\mathsf{Eval}(k,\cdot)}(1^\lambda) = 1 \ : \ k \leftarrow \mathsf{Setup}(1^\lambda)] - \Pr[\mathcal{A}^{R(\cdot)}(1^\lambda) = 1 \ : \ R \leftarrow \mathcal{U}_\lambda]| \leq \mathrm{negl}(\lambda)$$

where $\mathcal{U}_\lambda$ *is the set of all functions from* $\mathcal{X}_\lambda$ *to* $\mathcal{Y}_\lambda$*.*

3.2 Public Key Encryption (PKE)

A public key encryption scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ consists of the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter λ and outputs a pair of public and secret key $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: The encryption algorithm is a randomized algorithm that takes as input a public key pk and message $m \in \mathcal{M}$ and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs either a message $m \in \mathcal{M}$ or $\perp$.

Correctness. For correctness, we require for all $\lambda \in \mathbb{N}, (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $m \in \mathcal{M}$,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m] = 1,$$

where the probability is over the random bits used in the encryption algorithm.

CPA security. A public key encryption scheme PKE is said to satisfy CPA security if no PPT adversary $\mathcal{A}$ can win the following security game with probability greater than $\frac{1}{2} + \mathrm{negl}(\lambda)$.

- **Initialization Phase:** The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and sends pk to $\mathcal{A}$. The challenger also samples a uniformly random bit b.
- **Challenge Phase:** The adversary sends a query (m_0, m_1). The challenger replies with $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m_b)$.
- **Response Phase:** $\mathcal{A}$ outputs a bit b' and wins if $b = b'$.

KDM security. A public key encryption scheme PKE is said to be KDM secure over a class of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ if no PPT adversary $\mathcal{A}$ can win the following security game with probability greater than $\frac{1}{2} + \mathrm{negl}(\lambda)$.

- **Initialization Phase:** The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and sends pk to $\mathcal{A}$. The challenger also samples a uniformly random bit b.

- **Challenge Phase:** The adversary sends adaptive queries $f_1, , \cdots, f_Q$ with each $f_i \in \mathcal{F}_\lambda$. The challenger replies to the i^{th} query with $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m_b)$ where $m_0 = \mathbf{0}$ and $m_1 = f_i(\mathsf{sk})$.
- **Response Phase:** $\mathcal{A}$ outputs a bit b' and wins if $b = b'$.

We will work with the following class of functions for KDM security in this paper.

- **Circular:** A function is circular[1] if it is of the form $f_i(x_1, \ldots, x_n) = x_i$.
- **Projection:** A function is a projection function if each of its output bits depends on at most a single input bit.
- **Affine:** A function is an affine function if the function can be represented as $f(x) = Ax + b$ where A is a matrix and b is a vector.
- **Circuits of a-priori bounded size s:** A function in this class can be described by a circuit of size s.

LR security. A public key encryption scheme PKE is said to be ℓ-LR secure if no PPT adversary $\mathcal{A}$ can win the following security game with probability greater than $\frac{1}{2} + \mathrm{negl}(\lambda)$.

- **Initialization Phase:** The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pk to $\mathcal{A}$.
- **Leakage Phase:** $\mathcal{A}$ outputs a function $h : \mathcal{SK}_\lambda \rightarrow \{0,1\}^*$. The challenger sends the leakage $h(\mathsf{sk})$ to $\mathcal{A}$, given $|h(\mathsf{sk})| \leq \ell(\lambda)$.
- **Challenge Phase:** $\mathcal{A}$ sends (m_0, m_1) it to the challenger. The challenger randomly chooses $b \in \{0,1\}$ and computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{pk}, m_b)$. It sends ct^* to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ receives ct^* and outputs b'. $\mathcal{A}$ wins the experiment if $b = b'$.

3.3 Garbling Scheme

A garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ for a class of circuits $\{\mathcal{C}_\lambda\}_\lambda$ consists of the following algorithms.

- $\mathsf{Garble}(1^\lambda, C)$: The garbling algorithm is a randomized algorithm that takes as input the security parameter 1^λ and a circuit $C \in \mathcal{C}_\lambda$ such that $C : \{0,1\}^n \rightarrow \{0,1\}^m$ and outputs a garbled circuit $\hat{C}$ and a set of labels $\{\mathsf{lab}_{i,b}\}_{i\in[n], b\in\{0,1\}}$.
- $\mathsf{Eval}(\hat{C}, \{\mathsf{lab}_i\}_{i\in[n]})$: The evaluation algorithm takes as input a garbled circuit $\hat{C}$ and a set of n labels $\{\mathsf{lab}_i\}_{i\in[n]}$ and outputs $y \in \{0,1\}^m$.

[1] This definition is also used in [5].

Correctness. For correctness of a garbling scheme GC for a class of circuits $\{\mathcal{C}_\lambda\}_\lambda$, we require that for all $\lambda \in \mathbb{N}, C \in \mathcal{C}_\lambda$, $x \in \{0,1\}^n$,

$$\mathsf{Eval}(\hat{C}, \{\mathsf{lab}_{i,x_i}\}_{i\in[n]}) = C(x)$$

where $(\hat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)$.

Definition 2 (Simulation security). *A garbling scheme* $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ *for a class of circuits* $\{\mathcal{C}_\lambda\}_\lambda$ *is said to be secure if there exists a PPT algorithm* Sim *such that for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds.*

$$\left| \Pr\left[\mathcal{A}_2(\hat{C}, \{\mathsf{lab}_{i,x_i}\}_{i\in[n]}, \mathsf{aux}) = 1 : \begin{array}{c}(C, x, \mathsf{aux}) \leftarrow \mathcal{A}_1(1^\lambda),\\ (\hat{C}, \{\mathsf{lab}_{i,x_i}\}_{i\in[n]}) \leftarrow \mathsf{Sim}(1^\lambda, 1^n, 1^{|C|}, C(x))\end{array}\right]\right.$$

$$\left. - \Pr\left[\mathcal{A}_2(\hat{C}, \{\mathsf{lab}_{i,x_i}\}_{i\in[n]}, \mathsf{aux}) = 1 : \begin{array}{c}(C, x, \mathsf{aux}) \leftarrow \mathcal{A}_1(1^\lambda),\\ (\hat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C))\end{array}\right]\right| \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

Using the above security definition, we have the following theorems.

Theorem 1. *Let* $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ *be a secure garbling scheme. Then for all circuits* C_1, C_2*, with the same sizes, input-output spaces and input* x *such that* $C_1(x) = C_2(x)$*,*

$$\left(\hat{C_1}, \{\mathsf{lab}_{i,x_i}\} \mid (\hat{C_1}, \{\mathsf{lab}_{i,b}\}) \leftarrow \mathsf{Garble}(1^\lambda, C_1)\right) \approx_c \left(\hat{C_2}, \{\mathsf{lab}_{i,x_i}\} \mid (\hat{C_2}, \{\mathsf{lab}_{i,b}\}) \leftarrow \mathsf{Garble}(1^\lambda, C_2)\right).$$

Theorem 2. *Let* $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ *be a secure garbling scheme. Then for all circuits* C_1, C_2*, with the same input-output spaces and size,*

$$\left(\hat{C_1} \mid (\hat{C_1}, \{\mathsf{lab}_{i,b}\}) \leftarrow \mathsf{Garble}(1^\lambda, C_1)\right) \approx_c \left(\hat{C_2} \mid (\hat{C_2}, \{\mathsf{lab}_{i,b}\}) \leftarrow \mathsf{Garble}(1^\lambda, C_2)\right).$$

3.4 Hash Proof System (HPS)

A hash proof system $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$ associated with an efficiently sampleable language $\mathcal{L}$ consists of the following algorithms.

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter and outputs a pair of public and secret key $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Encaps}(\mathsf{pk}, (x, w))$: The encoding algorithm takes as input a public key pk and a pair of string and witness (x, w) from the language $\mathcal{L}$ and outputs a string $k \in \{0,1\}^\ell$.
- $\mathsf{Decaps}(\mathsf{sk}, x)$: The decoding algorithm takes as input a public key pk and string and outputs a string $k \in \{0,1\}^\ell$.

Correctness. For all $\lambda, (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(x, w) \leftarrow \mathcal{L}$,

$$\mathsf{Encaps}(\mathsf{pk}, (x, w)) = \mathsf{Decaps}(\mathsf{sk}, x)$$

Language Indistinguishability. For large enought λ, the following distributions are computationally close – $\mathcal{L} \approx_c \mathcal{L} \cup \overline{\mathcal{L}}$[2].

Smoothness property. For large enough λ, the following distributions are statistically close,

$$(\mathsf{pk}, x, \mathsf{Decaps}(\mathsf{sk}, x)) \approx_s (\mathsf{pk}, x, t)$$

where the distribution is over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$ and $t \leftarrow \{0,1\}^\ell$.

Definition 3. *A hash proof system is said to be homomorphic if the language $\mathcal{L} \cup \overline{\mathcal{L}}$ has a group structure with the operator $\odot$ and for all $\lambda \in \mathbb{N}, (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $x, y \in \mathcal{L} \cup \overline{\mathcal{L}}$, $\mathsf{Decaps}(\mathsf{sk}, x \odot y) = \mathsf{Decaps}(\mathsf{sk}, x) \oplus \mathsf{Decaps}(\mathsf{sk}, y)$.*

3.5 *d*-Linear Assumption (*d*-LIN)

Let $\mathbb{G}$ be a cyclic group or prime-order q and g be a generator of $\mathbb{G}$. We denote $\mathcal{G} = (\mathbb{G}, q, g)$ and $[x] = g^x$ where $x \in \mathbb{Z}_q$. The notation $[\cdot]$ can be naturally extended to vectors and matrices. We say that $\mathbb{G}$ satisfies d-Linear assumption if

$$(\mathcal{G}, [\mathbf{x}], [\mathbf{A}\mathbf{r}]) \approx_c (\mathcal{G}, [\mathbf{x}], [\mathbf{A}'\mathbf{r}'])$$

where $\mathbf{x}, \mathbf{r}' \leftarrow \mathbb{Z}_q^{d+1}, \mathbf{r} \leftarrow \mathbb{Z}_q^d$, and $\mathbf{A} \in \mathbb{Z}_q^{(d+1)\times d}, \mathbf{A}' \in \mathbb{Z}_q^{(d+1)\times(d+1)}$ such that

$$\mathbf{A} = \begin{bmatrix} x_1 & 0 & 0 & \dots & 0 \\ 0 & x_2 & 0 & \dots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \dots & x_d \\ x_{d+1} & x_{d+1} & x_{d+1} & \dots & x_{d+1} \end{bmatrix}, \mathbf{A}' = \begin{bmatrix} x_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & x_2 & 0 & \dots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \dots & x_d & 0 \\ 0 & 0 & 0 & \dots & 0 & x_{d+1} \end{bmatrix}$$

3.6 Batch Encryption

A batch encryption scheme $\mathsf{BENC} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of the following algorithms

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter λ and outputs a common reference string crs.
- $\mathsf{Gen}(\mathsf{crs}, \mathsf{sk})$: The key generation algorithm takes as input a common reference string crs and a secret key sk. It outputs a public key pk.
- $\mathsf{Enc}(\mathsf{crs}, \mathsf{pk}, \mathbf{M})$: The encryption algorithm takes as input a common reference string crs, public key pk, matrix $\mathbf{M}$ and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{crs}, \mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a common reference string crs, secret key sk, ciphertext ct and outputs a message m.

Correctness. For all $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathsf{sk}), \mathbf{M}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pk}, \mathbf{M})$ and $m' = \mathsf{Dec}(\mathsf{crs}, \mathsf{sk}, \mathsf{ct})$, we must have $m'_i = \mathbf{M}_{i,\mathsf{sk}_i}$.

[2] The above definition is from [42]. There are other works that define language indistinguishability as $\mathcal{L} \approx_c \overline{\mathcal{L}}$.

Security. A batch encryption scheme BENC is secure if no PPT adversary $\mathcal{A}$ can win the following security game with probability greater than $\frac{1}{2} + \mathrm{negl}(\lambda)$.

- **Initialization Phase:** The adversary takes 1^λ as input and sends sk to the challenger. The challenger runs $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and sends crs to $\mathcal{A}$. The challenger also samples a uniformly random bit b.
- **Challenge Phase:** The adversary sends a query $(\mathbf{M}^{(0)}, \mathbf{M}^{(1)})$ such that $\mathbf{M}^{(0)}_{i,\mathsf{sk}_i} = \mathbf{M}^{(1)}_{i,\mathsf{sk}_i}$ for all $i \in [n]$. The challenger computes $\mathsf{pk} = \mathsf{Gen}(\mathsf{crs}, \mathsf{sk})$ and replies with $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pk}, \mathbf{M}^{(b)})$.
- **Response Phase:** $\mathcal{A}$ outputs a bit b' and wins if $b = b'$.

Theorem 3. *Assuming the hardness of $X \in \{\mathsf{CDH}, \mathsf{LWE}, \mathsf{LPN}\}$, there exists secure batch encryption schemes.*

4 Leakage-Resilient Key-Dependent Message Secure Encryption Scheme : Definition

In this section, we proceed to formally define leakage-resilient key-dependent secure encryption scheme. Let $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme with secret key space $\{\mathcal{SK}_\lambda\}_\lambda$ and message space $\{\mathcal{M}_\lambda\}_\lambda$. Consider the following experiment with an adversary $\mathcal{A}$ where $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ such that $\mathcal{F}_\lambda$ is a class of functions $f : \mathcal{SK}_\lambda \to \mathcal{M}_\lambda$.

- **Initialization Phase:** The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pk to $\mathcal{A}$. The challenger also samples a uniformly random bit b.
- **Leakage Phase:** $\mathcal{A}$ outputs a function $h : \mathcal{SK}_\lambda \to \{0,1\}^*$. The challenger sends the leakage $h(\mathsf{sk})$ to $\mathcal{A}$.
- **Challenge Phase:** The adversary sends adaptive queries $f_1, , \cdots, f_Q$ with each $f_i \in \mathcal{F}_\lambda$. The challenger replies to the i^{th} query with $\mathsf{ct}_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m_b)$ where $m_0 = \mathbf{0}$ and $m_1 = f_i(\mathsf{sk})$.
- **Response Phase:** $\mathcal{A}$ receives ct^* and outputs b'. $\mathcal{A}$ wins the experiment if $b = b'$.

Definition 4. *A PKE scheme is said to be $(\ell, \mathcal{F})$-LR-KDM secure if for all PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$, $\mathsf{Pr}[\mathcal{A}$ wins in the above experiment$] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$, provided $|h(\mathsf{sk})| \leq \ell(\lambda)$.*

Definition 5. *A PKE scheme is said to be 1-ciphertext $(\ell, \mathcal{F})$-LR-KDM secure if for all PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$, $\mathsf{Pr}[\mathcal{A}$ wins in the above experiment$] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$, provided $|h(\mathsf{sk})| \leq \ell(\lambda)$ and $\mathcal{A}$ queries a single f in the challenge phase.*

Remark 1. We informally present the security game described in [20] to allow the reader to compare it with the security definition provided above. In the security definition from [20], after the adversary receives the public key, it submits two messages m_0, m_1 along with a leakage function h and a key-dependent function f. The adversary then receives an encryption of m_b for a randomly chosen bit b, as well as an encryption of $f(\mathsf{sk})$ and the leakage value $h(\mathsf{sk})$. The adversary's objective is to guess the value of b.

4.1 Non-Triviality of Leakage-Resilient Key-Dependent Message Security

We will show that there exists a scheme that is ℓ-leakage-resilient and $\mathcal{F}$-KDM secure, but is not $(\ell, \mathcal{F})$-LR-KDM secure. Let $\mathsf{SKE}' = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be a SKE scheme that is ℓ-leakage-resilient and circular-KDM secure where $\ell(\lambda) \geq \lambda$[3]. Let $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$ be a secure PRF that has key size, input size and output size equal to the security parameter λ. Consider a SKE scheme SKE with the following description.

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter λ. It generates $k \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ and $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$. It outputs $\mathsf{sk} := (k, \mathsf{ske.sk})$.
- $\mathsf{Enc}(\mathsf{sk}, m)$: The encryption algorithm takes as input a secret key $\mathsf{sk} = (k, \mathsf{ske.sk})$ and a message m. If $m \neq \mathsf{ske.sk}$, it sets $c_0 := \mathsf{PRF.Eval}(k, \mathbf{0})$. Otherwise, it sets $c_0 = \mathsf{PRF.Eval}(k, \mathbf{1})$. It computes $c_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, m)$ and outputs $\mathsf{ct} = (c_0, c_1)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key $\mathsf{sk} = (k, \mathsf{ske.sk})$ and a ciphertext $\mathsf{ct} = (c_0, c_1)$. It returns $\mathsf{SKE.Dec}(\mathsf{ske.sk}, c_1)$.

The correctness of the above scheme is straight-forward. We will now show that SKE is ℓ-leakage-resilient and f-KDM secure separately, where f will be defined later.

Claim 1. Assuming that SKE' is ℓ-leakage-resilient, then SKE is ℓ-leakage-resilient secure.

Proof. Assume the contrary that there exists an adversary $\mathcal{A}$ that wins the leakage-resilient game for SKE scheme with non-negligible advantage. We will construct an adversary $\mathcal{B}$ that wins the leakage-resilient game for SKE'.

- **Initialization Phase:** The challenger runs $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$. $\mathcal{B}$ generates $k \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
- **Leakage Phase:** $\mathcal{A}$ sends a function $h(\cdot, \cdot)$ to $\mathcal{B}$. $\mathcal{B}$ sends the function $h'(\cdot) = h(k, \cdot)$ to the challenger. The challenger sends the leakage $h'(\mathsf{ske.sk}) = h(k, \mathsf{ske.sk})$ to $\mathcal{B}$ who relays it to $\mathcal{A}$.
- **Challenge Phase:** $\mathcal{A}$ sends m_0, m_1 to $\mathcal{B}$ who relays it to the challenger. The challenger randomly chooses $b \in \{0, 1\}$ and computes a ciphertext $c_1^* = \mathsf{SKE.Enc}(\mathsf{ske.sk}, m_0)$ if $b = 0$, else it computes $c_1^* = \mathsf{SKE.Enc}(\mathsf{ske.sk}, m_1)$. It sends c_1^* to $\mathcal{B}$. $\mathcal{B}$ generates $c_0 = \mathsf{PRF.Eval}(k, \mathbf{0})$ and sends $\mathsf{ct}^* := (c_0, c_1^*)$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ receives ct^* and returns b'. $\mathcal{B}$ outputs b'.

Observe that $\mathcal{B}$ has exactly simulated the leakage-resilient game for $\mathcal{A}$ except when $\mathsf{ske.sk}$ is either equal to m_0 or m_1. But, this happens with negligible probability[4]. So,

$$\Pr[\mathcal{B} \text{ wins the experiment}] = \Pr[\mathcal{A} \text{ wins the experiment}] - \mathrm{negl}(\lambda).$$

[3] Clearly, the size of the secret key, $|\mathsf{ske.sk}|$ must also be greater than λ

[4] Otherwise, SKE' is not even CPA-secure.

This is a contradiction because SKE' is ℓ-leakage-resilient. Hence, SKE is ℓ-leakage-resilient secure. □

Claim 2. Assuming that PRF is a secure PRF and SKE' is circular-KDM secure, then SKE is f-KDM secure where $f(x, y) = y$.

Proof. It is crucial to note that the output of the function f on input sk is only $\mathsf{ske.sk}$. In other words, $\mathsf{SKE.Enc}(\mathsf{ske.sk}, f(\mathsf{sk})) = \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{ske.sk})$, and this fact will be used in our proof. Let us first consider a hybrid experiment for the f-KDM game (refer Section 3.2) for the above scheme where the challenger does not generate a PRF key. As a consequence, it generates a truly random value c_0 to compute the challenge ciphertext.

- **Initialization Phase:** The challenger runs $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.
- **Challenge Phase:** $\mathcal{A}$ sends the function f to the challenger[5]. The challenger randomly chooses $b \in \{0, 1\}$ and computes a ciphertext $c_1^* = \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathbf{0})$ if $b = 0$, else it computes $c_1^* = \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{ske.sk})$. It randomly generates c_0 and sends $\mathsf{ct}^* := (c_0, c_1^*)$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ receives ct^* and returns b'.

Observe that there is no PPT adversary $\mathcal{A}$ that can distinguish the original KDM game from this hybrid game because of the PRF security. This is because the PRF key k is only used to generate c_0 and isn't used in the generation of c_1. In other words, if there exists an adversary $\mathcal{A}$ that can distinguish these two games, then we can break the security of the underlying PRF scheme (see Section 3.1).

Now, assume the contrary that there exists an adversary $\mathcal{A}$ that wins in this hybrid game. We will construct an adversary $\mathcal{B}$ that wins the KDM game for SKE'.

- **Initialization Phase:** The challenger runs $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.
- **Challenge Phase:** $\mathcal{A}$ sends the function f to $\mathcal{B}$. $\mathcal{B}$ sends the function $f'(y) = y$ to the challenger. The challenger randomly chooses $b \in \{0, 1\}$ and computes a ciphertext $c_1^* = \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathbf{0})$ if $b = 0$. Else, it computes $c_1^* = \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{ske.sk})$. It sends c_1^* to $\mathcal{B}$. $\mathcal{B}$ randomly generates c_0 and sends $\mathsf{ct}^* := (c_0, c_1^*)$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ receives ct^* and returns b'. $\mathcal{B}$ outputs b'.

Observe that $\mathcal{B}$ has exactly simulated this hybrid game for $\mathcal{A}$. So,

$$\Pr[\mathcal{B} \text{ wins the experiment}] = \Pr[\mathcal{A} \text{ wins the experiment}]$$

This is a contradiction. Hence, SKE is f-KDM secure. □

We will now show that SKE is not (ℓ, f)-LR-KDM secure.

[5] Here, we consider the case where the security is against a single challenge ciphertext. It can be trivially extended to multi-challenge ciphertexts.

Claim 3. SKE is not (ℓ, f)-LR-KDM secure.

Proof. We give the description of an adversary $\mathcal{A}$ that wins the LR-KDM security game for SKE.

- **Initialization Phase:** The challenger runs $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and $k \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ and set $\mathsf{sk} = (k, \mathsf{ske.sk})$.
- **Leakage Phase:** $\mathcal{A}$ sends a function $h(\cdot,\cdot)$ such that $h(x, y) = x$. The challenger sends the leakage $h(k, \mathsf{ske.sk}) = k$ to $\mathcal{A}$.
- **Challenge Phase:** $\mathcal{A}$ sends the function f to the challenger. The challenger randomly chooses $b \in \{0, 1\}$ and computes a ciphertext $c_1^* = \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathbf{0})$ and $c_0^* = \mathsf{PRF.Eval}(k, \mathbf{0})$ if $b = 0$, else it computes $c_1^* = \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{ske.sk})$ and $c_0^* = \mathsf{PRF.Eval}(k, \mathbf{1})$. It sends $\mathsf{ct}^* := (c_0^*, c_1^*)$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ receives ct^*. It checks if and outputs 0 if $c_0^* = \mathsf{PRF.Eval}(k, \mathbf{0})$. Else, it outputs 1.

We first note that $h(\mathsf{sk}) = k$, therefore, $|h(\mathsf{sk})| = |k| = \lambda \leq \ell(\lambda)$. Therefore, $\mathcal{A}$ is a valid adversary that follows the required bounds in the experiment. Observe that $\mathcal{A}$ wins this game with high probability except when $\mathsf{PRF.Eval}(k, \mathbf{1}) = \mathsf{PRF.Eval}(k, \mathbf{0})$. But, this happens with negligible probability, otherwise a PPT adversary can distinguish PRF from a truly random function. This is because for a truly random function F whose outputs are of length λ, the probability of $F(0) = F(1)$ is negligible in λ. □

5 LR-KDM from Hash Proof Systems

In this section, we present our LR-KDM construction from *leakage-resilient* HPS and show that by using LR-HPS in the construction given by Wee [42], we can achieve LR-KDM.

5.1 LR-KDM from Leakage-Resilient Homomorphic Hash Proof System

In this section, we start by formally defining a leakage-resilient homomorphic HPS.

Definition 6 (Leakage-Resilient Homomorphic Hash Proof System). *A homomorphic hash proof system* $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$ *is said to be* ℓ*-leakage-resilient if it satisfies the following leakage-resilient smoothness property - For large enough* λ*, any function* f *such that the size of the output of* f *is at most* $\ell(\lambda)$*,*

$$(\mathsf{pk}, f(\mathsf{sk}), x, \mathsf{Decaps}(\mathsf{sk}, x)) \approx_s (\mathsf{pk}, f(\mathsf{sk}), x, t)$$

where the distribution is over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$*,* $x \leftarrow \overline{\mathcal{L}}$ *and* t *is chosen from uniform distribution.*

Construction of homomorphic LR-Hash Proof System We will present a construction for homomorphic ℓ-LR-HPS from d-LIN assumptions

Let $\mathbb{G}$ be a group of prime order q. Let $k \geq (d+2)\log q + \ell$. Sample $\mathbf{P} \leftarrow_R \mathbb{Z}_q^{d\times k}$ and output

$$\mathsf{pp} := (\mathbb{G}, q, g, g^{\mathbf{P}})$$

We define

$$\mathcal{L} := \left\{ g^{\mathbf{r}^\top \mathbf{P}} : \mathbf{r} \in \mathbb{Z}_q^d \right\} \text{ and } \overline{\mathcal{L}} := \left\{ g^{\mathbf{a}^\top} : \mathbf{a} \in \mathbb{Z}_q^k \right\} \setminus \mathcal{L}$$

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter 1^λ. It computes $\mathsf{pp} = (G, q, g, g^{\mathbf{P}})$ and randomly generates $\mathbf{s} \leftarrow \{0,1\}^k$. It sets $\mathsf{pk} := (\mathsf{pp}, g^{\mathbf{Ps}})$ and $\mathsf{sk} := \mathbf{s}$.
- $\mathsf{Encaps}\left(\mathsf{pk}, (x = g^{\mathbf{r}^\top \mathbf{P}}, w = \mathbf{r})\right)$: The encapsulation algorithm takes as input a public key $\mathsf{pk} := (\mathsf{pp}, g^{\mathbf{Ps}})$ and $x \in \mathcal{L}$ along with its witness w. Using $\mathbf{r} = w$, and $g^{\mathbf{Ps}}$, it outputs $\mathsf{Encaps}(\mathsf{pk}, (x, w)) := g^{\mathbf{r}^\top \mathbf{Ps}}$.
- $\mathsf{Decaps}(\mathsf{sk}, x = g^{\mathbf{a}^\top})$: The decapsulation algorithm takes as input a secret key $\mathsf{sk} = \mathbf{s}$ and $x = g^{\mathbf{a}^\top}$. It computes $\mathsf{Decaps}(\mathsf{sk}, x) := g^{\mathbf{a}^\top \mathbf{s}}$

Theorem 4. *The above scheme is a valid Leakage-Resilient Hash Proof System*

Proof (Proof sketch). Correctness holds by construction. The language indistinguishability follows from the d-LIN assumption[6] and homomorphism is also trivial. For leakage-resilient smoothness, we want to show that

$$(\mathsf{pk}, f(\mathsf{sk}), x, \mathsf{Decaps}(\mathsf{sk}, x)) \approx_c (\mathsf{pk}, f(\mathsf{sk}), x, t)$$

where t is chosen uniformly at random. More precisely,

$$((\mathbb{G}, q, g, g^{\mathbf{P}}, g^{\mathbf{Ps}}), f(\mathbf{s}), g^{\mathbf{a}^\top}, g^{\mathbf{a}^\top \mathbf{s}}) \approx_s ((\mathbb{G}, q, g, g^{\mathbf{P}}, g^{\mathbf{Ps}}), f(\mathbf{s}), g^{\mathbf{a}^\top}, g^{\mathbf{t}})$$

This reduces to

$$(\mathbf{P}, \mathbf{a}^\top, \mathbf{Ps}, \mathbf{a}^\top \mathbf{s}, f(\mathbf{s})) \approx_s (\mathbf{P}, \mathbf{a}^\top, \mathbf{Ps}, \mathbf{t}, f(\mathbf{s}))$$

which follows from LHL theorem because $|f(\mathbf{s})| \leq \ell$.

LR-KDM Secure Encryption from LR Hash Proof System In this section, we give a construction for Leakage-Resilient Key-Dependent Secure Encryption from LR-HPS.

[6] For more details, see [39, Appendix A].

Construction Let $\mathsf{HPS} = (\mathsf{HPS.Setup}, \mathsf{HPS.Encaps}, \mathsf{HPS.Decaps})$ be a ℓ-leakage-resilient homomorphic hash proof system. Consider the following public key encryption scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter 1^λ and runs $(\mathsf{hps.pk}, \mathsf{hps.sk}) \leftarrow \mathsf{HPS.Setup}(1^\lambda)$. It outputs $\mathsf{pk} := \mathsf{hps.pk}$ and $\mathsf{sk} := \mathsf{hps.sk}$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: The encryption algorithm takes as input a public key $\mathsf{pk} = \mathsf{hps.pk}$ and a message m. It samples a string along with its witness $(x, w) \in \mathcal{R}_{\mathcal{L}}$ and outputs $\mathsf{ct} = (x, \mathsf{HPS.Encaps}(\mathsf{pk}, (x, w)) \oplus m)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key $\mathsf{sk} = \mathsf{hps.sk}$ and a ciphertext $\mathsf{ct} = (c_0, c_1)$. It computes $m = \mathsf{HPS.Decaps}(\mathsf{sk}, c_0) \oplus c_1$ and outputs m.

Correctness. A ciphertext of a message m in the above scheme is $(c_0, c_1) = (x, k \oplus m)$ where $k \leftarrow \mathsf{HPS.Encaps}(\mathsf{pk}, (x, w))$. From the correctness of HPS, we have $k = \mathsf{HPS.Decaps}(\mathsf{sk}, x)$. Therefore, we get $\mathsf{HPS.Decaps}(\mathsf{sk}, c_0) \oplus c_1 = k \oplus (k \oplus m) = m$.

Theorem 5. *Assuming that* HPS *is an ℓ-leakage resilient homomorphic hash proof system, the above encryption scheme is a 1-ciphertext $(\ell, \mathcal{F})$-LR-KDM where $\mathcal{F} = \{\mathcal{F}_\lambda\}$*

$$\mathcal{F}_\lambda = \{f_{e,k}(\mathsf{sk}) = \mathsf{Decaps}(\mathsf{sk}, e) \oplus k\}_{e,k}$$

The proof for the above theorem is provided in Section 9. Since, the LR-HPS scheme based on d-Lin presented in the previous section has the property that $\mathsf{Decaps}(\mathbf{s}, g^{\mathbf{a}^\top}) = g^{\mathbf{a}^\top \mathbf{s}} = [\mathbf{a}^\top \mathbf{s}]$ which is an linear function in $\mathbf{s}$ with respect to $[\cdot]$, we have the following theorem.

Theorem 6. *Assuming the hardness of d-LIN, there exists 1-ciphertext LR-KDM secure encryption scheme for the class of affine functions.*

6 LR-KDM from Batch Encryption

In this section, we show that the construction presented by Brakerski *et al.* [13] is LR-KDM secure over the class of affine functions over the bits of the secret key. Similar to the assumption made in [13, Section 7.2], we will also assume that the public key of the batch encryption scheme is of size λ.

Construction Let $\mathsf{BENC} = (\mathsf{Batch.Setup}, \mathsf{Batch.Gen}, \mathsf{Batch.Enc}, \mathsf{Batch.Dec})$ be a batch encryption scheme with public key size λ. Consider the following public key encryption scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter λ. It generates $\mathsf{crs} \leftarrow \mathsf{Batch.Setup}(1^{\lambda+n})$ and samples a uniformly random $\mathsf{sk} \leftarrow \{0,1\}^n$. It computes $\mathsf{pk} = \mathsf{Batch.Gen}(\mathsf{crs}, \mathsf{sk})$ and outputs $(\mathsf{crs}, \mathsf{pk}, \mathsf{sk})$.

- $\mathsf{Enc}(\mathsf{crs}, \mathsf{pk}, m)$: The encryption algorithm takes as input crs, the public key pk and a message m. It constructs an XOR secret sharing $\{v_i\}_{i\in[n]}$ for m so that $m = v_1 \oplus \cdots \oplus v_n$. It constructs a $n \times 2$ matrix $\mathbf{M}$ such that $\mathbf{M}_{i,b} = v_i$ for $i \in [n], b \in \{0,1\}$. Finally, it outputs $\mathsf{ct} = \mathsf{Batch.Enc}(\mathsf{crs}, \mathsf{pk}, \mathbf{M})$.
- $\mathsf{Dec}(\mathsf{crs}, \mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input crs, the secret key sk and a ciphertext ct. It computes $\{v'_i\} = \mathsf{Batch.Dec}(\mathsf{crs}, \mathsf{sk}, \mathsf{ct})$. It outputs $v'_1 \oplus \cdots \oplus v'_n$.

Correctness. From the correctness of $\mathsf{Batch.Enc}$, we get

$$\begin{aligned}\mathsf{Dec}(\mathsf{crs}, \mathsf{sk}, \mathsf{Enc}(\mathsf{crs}, \mathsf{pk}, m)) &= \bigoplus_{i=1}^{n} \mathsf{Batch.Dec}(\mathsf{crs}, \mathsf{sk}, \mathsf{Batch.Enc}(\mathsf{crs}, \mathsf{pk}, \mathbf{M}))_i \\ &= \bigoplus_{i=1}^{n} \mathbf{M}_{i,\mathsf{sk}_i} \\ &= v_1 \oplus \cdots \oplus v_n = m,\end{aligned}$$

where we had computed $\mathbf{M}_{i,b} = v_i$ with $v_1 \oplus \cdots \oplus v_n = m$.

Theorem 7. *Assuming* BENC *is a secure batch encryption, for any* $n = \mathsf{poly}(\lambda)$*, the above encryption scheme is LR-KDM secure over the class of affine functions over the bits of the secret key*

$$\mathcal{F}_\lambda = \left\{ f_a \mid f_a(\mathsf{sk}) = a_{n+1} \oplus \left(\bigoplus_{i=1}^{n} a_i \cdot \mathsf{sk}_i \right) \right\}$$

and with leakage at most $n - 2\lambda$[7]*.*

Proof. We will show this using a sequence of hybrid arguments.

G_0: This is the original LR-KDM game.

- **Initialization Phase:**
  1. The challenger randomly generates $\mathsf{crs} \leftarrow \mathsf{Batch.Setup}(1^{\lambda+n})$ and $\mathsf{sk} \leftarrow \{0,1\}^n$.
  2. It sets $\mathsf{pk} = \mathsf{Batch.Gen}(\mathsf{crs}, \mathsf{sk})$ and sends $\mathsf{crs}, \mathsf{pk}$ to the adversary $\mathcal{A}$
- **Leakage Phase:**
  1. $\mathcal{A}$ sends a function h to the challenger.
  2. The challenger responds with $h(\mathsf{sk})$.
- **Challenge Phase:**
  1. The challenger randomly picks $\beta \leftarrow \{0,1\}$.
  2. The adversary makes Q queries as follows:
     - $\mathcal{A}$ sends a function $f_a \in \mathcal{F}_\lambda$ to the challenger.
     - Let $m_0 = \mathbf{0}$ and $m_1 = f_a(\mathsf{sk})$.
     - Construct the matrix $\mathbf{M}$ such that $\mathbf{M}_{i,b} = v_i$ for all $i \in [n], b \in \{0,1\}$ where $\bigoplus_{i=1}^{n} v_i = m_\beta$.

[7] By setting $n = \omega(\lambda)$, we get $1 - o(1)$ leakage-rate.

- It sends $\mathsf{Batch.Enc}(\mathsf{crs}, \mathsf{pk}, \mathbf{M})$ to $\mathcal{A}$.

– **Response Phase:** $\mathcal{A}$ outputs $\beta' \in \{0,1\}$.

G_1: The challenger modifies the matrix being encrypted. Observe that the knowledge of sk is not required while generating the matrix $\mathbf{M}$.

– **Challenge Phase:**
  1. The challenger randomly picks $\beta \leftarrow \{0,1\}$.
  2. The adversary makes Q queries as follows:
     - $\mathcal{A}$ sends a function $f_a \in \mathcal{F}_\lambda$ to the challenger.
     - Let $m_0 = \mathbf{0}$ and $m_1 = f_a(\mathsf{sk})$.
     - If $\beta = 0$, then construct the matrix $\mathbf{M}$ as in the previous game. If $\beta = 1$, construct the matrix $\mathbf{M}$ such that $\mathbf{M}_{i,b} = ba_i \oplus v_i$ for $i < n$ and $\mathbf{M}_{n,b} = a_{n+1} \oplus ba_n \oplus v_n$ where $v_1 \oplus \cdots \oplus v_n = 0$.
     - It sends $\mathsf{Batch.Enc}(\mathsf{crs}, \mathsf{pk}, \mathbf{M})$ to $\mathcal{A}$.

– **Response Phase:** $\mathcal{A}$ outputs $\beta' \in \{0,1\}$.

G_2: The challenger modifies the matrix being encrypted.

– **Challenge Phase:**
  1. The challenger randomly picks $\beta \leftarrow \{0,1\}$.
  2. The adversary makes Q queries as follows:
     - $\mathcal{A}$ sends a function $f_a \in \mathcal{F}_\lambda$ to the challenger.
     - Let $m_0 = \mathbf{0}$ and $m_1 = f_a(\mathsf{sk})$.
     - The matrix $\mathbf{M}$ is sampled uniformly at random.
     - It sends $\mathsf{Batch.Enc}(\mathsf{crs}, \mathsf{pk}, \mathbf{M})$ to $\mathcal{A}$.

– **Response Phase:** $\mathcal{A}$ outputs $\beta' \in \{0,1\}$.

Analysis: Let $p_{\mathcal{A},i}$ denote the probability that $\mathcal{A}$ wins in Game G_i. We will show that the success probability in each game is close to the previous one.

Lemma 1. *From the semantic security of* BENC*, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that* $|p_{\mathcal{A},1} - p_{\mathcal{A},0}| \leq \mathrm{negl}(\lambda)$.

Proof. We show the result by constructing $Q - 1$ hybrids between the games where in each hybrid we successively modify the matrix $\mathbf{M}$ for the ith query to be constructed as per game 1. Observe that for each successive hybrid, the distributions of $\{\mathbf{M}_{i,\mathsf{sk}_i}\}_{i\in[n]}$ are identical. Thus by the security of BENC, the hybrids are computationally indistinguishable. Summing over the hybrids and using the triangle inequality gives us our desired result. □

Lemma 2. *By the leftover hash lemma, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that* $|p_{\mathcal{A},2} - p_{\mathcal{A},1}| \leq \mathrm{negl}(\lambda)$.

Proof. We again split the two games with $Q-1$ hybrids in between where in each hybrid we successively modify the matrix $\mathbf{M}$ for the ith query to be constructed as per game 2. Consider the hybrids i and $i+1$. We construct a hybrid $\mathcal{H}'$ between these two where the challenger constructs the matrix $\mathbf{M}$ uniformly at random subject to the condition that $\mathbf{M}_{1,\mathsf{sk}_1} \oplus \cdots \oplus \mathbf{M}_{n,\mathsf{sk}_n} = m_\beta$. Then the ith hybrid and $\mathcal{H}'$ are computationally indistinguishable by the security of BENC.

In $\mathcal{H}'$, the matrix $\mathbf{M}$ is sampled uniformly at random subject to the condition $\mathbf{M}_{1,\mathsf{sk}_1} \oplus \cdots \oplus \mathbf{M}_{n,\mathsf{sk}_n} = m_\beta$. Equivalently, every element of $\mathbf{M}$ except $\mathbf{M}_{n,0}$ is sampled uniformly at random and we set

$$\mathbf{M}_{n,0} = m_\beta \oplus \left(\bigoplus_{i=1}^{n-1} \mathbf{M}_{i,0}\right) \oplus \left(\bigoplus_{i=1}^{n} \mathsf{sk}_i \cdot (\mathbf{M}_{i,0} \oplus \mathbf{M}_{i,1})\right).$$

By the leftover hash lemma, the distribution $\mathsf{sk} \mid \mathsf{crs}, \mathsf{pk}, h(\mathsf{sk})$ has average min-entropy at least $n - \lambda - (n - 2\lambda) = \lambda$. This is because crs is independent of sk and $|\mathsf{pk}| = \lambda, |h(\mathsf{sk})| \leq n - 2\lambda$. Thus the distribution of $\bigoplus_{i=1}^{n} \mathsf{sk}_i \cdot (\mathbf{M}_{i,0} \oplus \mathbf{M}_{i,1})$ is statistically indistinguishable from a uniformly random value independent of $\mathsf{crs}, \mathsf{pk}$ and $(\mathbf{M}_{i,0} \oplus \mathbf{M}_{i,1})_i$. Since the distribution of $\mathbf{M}$ used in the two games are statistically indistinguishable, $\mathcal{H}'$ and the $(i+1)$th hybrid are also statistically indistinguishable. Summing over the hybrids and using the triangle inequality gives us our desired result. □

Using the above lemmas and the fact that $p_{\mathcal{A},2} = \frac{1}{2}$, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $p_{\mathcal{A},0} \leq \frac{1}{2} + \mathrm{negl}(\lambda)$.

7 LR-KDM SKE to LR-KDM PKE

In this section, we give a construction for LR-KDM secure PKE scheme for circuits from LR-KDM secure SKE for projection functions and LR secure PKE.

Our Construction Let $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be an LR-PKE scheme and $(\mathsf{Garble}, \mathsf{Eval})$ be a garbled circuit scheme. Let $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be an LR-KDM SKE for projection functions with secret key size $\beta = \beta(\lambda)$. Consider the following public key encryption scheme:

- $\mathsf{Setup}(1^\lambda)$: The key generation algorithm takes as input the security parameter λ. For each $i \in [\beta]$ and $b \in \{0,1\}$, it samples $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$. It then generates $k \leftarrow \{0,1\}^\beta$. Finally, it sets
$$\overline{\mathsf{pk}} = \left\{\mathsf{pk}_{i,b}\right\}_{i\in[\beta],b\in\{0,1\}}, \ \overline{\mathsf{sk}} = \left(k, \{\mathsf{sk}_{i,k_i}\}_{i\in[\beta]}\right)$$
- $\mathsf{Enc}(\overline{\mathsf{pk}}, m)$: The encryption algorithm takes as input the public key $\overline{\mathsf{pk}}$ and a message m. Let $\left(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C_m)$ where C_m is the circuit that outputs m on any input. For each $i \in [\beta]$ and $b \in \{0,1\}$, it samples $\mathsf{ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,b}, \mathsf{lab}_{i,b})$ and returns $\overline{\mathsf{ct}} = \left(\widehat{C}, \{\mathsf{ct}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}\right)$.

– $\mathsf{Dec}(\overline{\mathsf{sk}}, \overline{\mathsf{ct}})$: The decryption algorithm takes as input the secret key $\overline{\mathsf{sk}}$ and a ciphertext $\overline{\mathsf{ct}}$. For each $i \in [\beta]$, it computes $\mathsf{lab}_{i,k_i} = \mathsf{PKE.Dec}(\mathsf{sk}_{i,k_i}, \mathsf{ct}_{i,k_i})$. It returns $m \leftarrow \mathsf{Eval}(\widehat{C}, \{\mathsf{lab}_{i,k_i}\})$.

Correctness. Let $\overline{\mathsf{ct}}$ be an encryption of m using the keys $\overline{\mathsf{pk}} = \left\{\mathsf{pk}_{i,b}\right\}_{i\in[\beta],b\in\{0,1\}}$ and let $\overline{\mathsf{sk}} = \left(k, \{\mathsf{sk}_{i,k_i}\}_{i\in[\beta]}\right)$. The decryption algorithm gets the right labels $\{\mathsf{lab}_{i,k_i}\}_{i\in[\beta]}$ used during encryption by the correctness of the underlying PKE scheme. By the correctness of the garbling scheme, we have $\mathsf{Eval}(\widehat{C}, \{\mathsf{lab}_{i,k_i}\}_{i\in[\beta]}) = C_m(k) = m$, as desired.

Leakage Rate. The leakage rate of PKE is $\frac{\ell(\lambda)}{|\mathsf{pke.sk}|}$, whereas the above scheme's leakage rate is $\frac{\ell(\lambda)}{\beta(|\mathsf{pke.sk}|+1)}$. There is a degradation factor of β in the leakage ratio.

Theorem 8. *Let* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *be a* ℓ*-LR PKE scheme,* $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ *be an* $(\ell, \mathcal{P})$*-LR-KDM secure SKE where* $\mathcal{P}$ *is the class of all projection functions and* $(\mathsf{Garble}, \mathsf{Eval})$ *a secure garbling scheme. Then the above scheme is a secure* $(\ell, \mathcal{F})$*-LR-KDM incompressible public key encryption scheme where* $\mathcal{F} = \{\mathcal{F}_\lambda\}$ *and* $\mathcal{F}_\lambda$ *is the class of bounded polynomial sized circuits.*

The proof of the above theorem is provided in Section 10.

8 Functionality Amplification for LR-KDM

In this section, we give a construction for LR-KDM secure PKE scheme for circuits from LR-KDM secure PKE for projection functions using garbling schemes.

Our Construction Let $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be an public key encryption scheme and $(\mathsf{Garble}, \mathsf{Eval})$ be a garbled circuit scheme. Consider the following public key encryption scheme:

– $\mathsf{Setup}(1^\lambda)$: The key generation algorithm takes as input the security parameter λ. It samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and returns $(\mathsf{pk}, \mathsf{sk})$.
– $\mathsf{Enc}(\mathsf{pk}, m)$: The encryption algorithm takes as input a public key pk and a message m. Let $\left(\widehat{C}, \{\mathsf{lab}_i\}_{i\in[|m|]}\right) \leftarrow \mathsf{Sim}(1^\lambda, |m|, |C_m|, m)$ where C_m is the circuit that outputs m on any input. It computes $\mathsf{pke.ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \{\mathsf{lab}_i\}_i)$ and outputs $\mathsf{ct} := (\hat{C}, \mathsf{pke.ct})$.
– $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input the secret key sk and a ciphertext $\mathsf{ct} = (\mathsf{Garble}C, \mathsf{pke.ct})$. It decrypts ct using sk to obtain $\{\mathsf{lab}_i\}_i$. It returns $m = \mathsf{Eval}(\widehat{C}, \{\mathsf{lab}_i\})$.

Correctness. Let ct be an encryption of m using the keys pk and let sk be the associated secret key. The decryption algorithm gets the right labels $\{\mathsf{lab}_i\}_{i\in[|m|]}$ used during encryption by the correctness of the underlying PKE scheme. By the correctness and security of the garbling scheme, we have $\mathsf{Eval}(\widehat{C}, \{\mathsf{lab}_i\}_{i\in[\beta]}) = m$, as desired.

Theorem 9. *Let* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *be a* ℓ*-LR-KDM PKE scheme for projection functions, and* $(\mathsf{Garble}, \mathsf{Eval})$ *a secure garbling scheme. Then the above scheme is a secure* $(\ell, \mathcal{C})$*-LR-KDM PKE scheme where* $\mathcal{C}$ *is the class of all bounded polynomial-sized circuits.*

The proof for the above theorem is provided in Section 11. By combining Theorem 3, Theorem 7 and Theorem 9, we obtain the following theorem.

Theorem 10. *Assuming the hardness of* $X \in \{\mathsf{CDH}, \mathsf{LWE}, \mathsf{LPN}\}$*, there exists a leakage-rate-1 LR-KDM secure scheme for the class of all bounded polynomial-sized circuits.*

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The em side—channel (s). In: Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4. pp. 29–45. Springer (2003) 1
2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Theory of cryptography conference. pp. 474–495. Springer (2009) 2
3. Ananth, P., Kaleoglu, F., Yuen, H.: Simultaneous haar indistinguishability with applications to unclonable cryptography. arXiv preprint arXiv:2405.10274 (2024) 2
4. Applebaum, B.: Key-dependent message security: Generic amplification and completeness. Journal of cryptology **27**(3), 429–451 (2014) 2, 3, 7
5. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. pp. 595–618. Springer (2009) 2, 9
6. Applebaum, B., Harnik, D., Ishai, Y.: Semantic security under related-key attacks and applications. Cryptology ePrint Archive (2010) 2
7. Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29. pp. 423–444. Springer (2010) 2
8. Bhushan, K., Goyal, R., Koppula, V., Narayanan, V., Prabhakaran, M., Rajasree, M.S.: Leakage-resilient incompressible cryptography: Constructions and barriers. Springer-Verlag (2024) 2
9. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002 St. John's, Newfoundland, Canada, August 15–16, 2002 Revised Papers 9. pp. 62–75. Springer (2003) 2
10. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision diffie-hellman. In: Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28. pp. 108–125. Springer (2008) 2, 3

11. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability: (or: Quadratic residuosity strikes back). In: Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30. pp. 1–20. Springer (2010) 2, 3
12. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. pp. 501–510. IEEE (2010) 2
13. Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous ibe, leakage resilience and circular security from new assumptions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 535–564. Springer (2018) 2, 3, 6, 17
14. Braverman, M., Hassidim, A., Kalai, Y.T.: Leaky pseudo-entropy functions. In: ICS. vol. 2011, p. 2nd (2011) 2
15. Cakan, A., Goyal, V., Liu-Zhang, C.D., Ribeiro, J.: Unbounded leakage-resilience and intrusion-detection in a quantum world. Cryptology ePrint Archive (2023) 2
16. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19. pp. 453–469. Springer (2000) 1
17. Di Crescenzo, G., Lipton, R., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Theory of Cryptography Conference. pp. 225–244. Springer (2006) 2
18. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. pp. 511–520. IEEE (2010) 2
19. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16. pp. 613–631. Springer (2010) 2
20. Dodis, Y., Karthikeyan, H., Wichs, D.: Updatable public key encryption in the standard model. In: Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part III 19. pp. 254–285. Springer (2021) 2, 3, 12
21. Dodis, Y., Sahai, A., Smith, A.: On perfect and adaptive security in exposure-resilient cryptography. In: Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20. pp. 301–324. Springer (2001) 1
22. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Theory of Cryptography Conference. pp. 207–224. Springer (2006) 2
23. Faonio, A., Nielsen, J.B., Venturi, D.: Mind your coins: Fully leakage-resilient signatures with graceful degradation. In: International Colloquium on Automata, Languages, and Programming. pp. 456–468. Springer (2015) 2
24. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. In: Theory of Cryptography: 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings 7. pp. 343–360. Springer (2010) 2

25. Galindo, D., Vivek, S.: A leakage-resilient pairing-based variant of the schnorr signature scheme. In: Cryptography and Coding: 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings 14. pp. 173–192. Springer (2013) 2
26. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM (JACM) **33**(4), 792–807 (1986) 5
27. Hajiabadi, M., Kapron, B.M., Srinivasan, V.: On generic constructions of circularly-secure, leakage-resilient public-key encryption schemes. In: Public-Key Cryptography–PKC 2016, pp. 129–158. Springer (2016) 2, 3
28. Han, S., Liu, S., Gu, D.: Almost tight multi-user security under adaptive corruptions & leakages in the standard model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 132–162. Springer (2023) 2
29. Hazay, C., López-Alt, A., Wee, H., Wichs, D.: Leakage-resilient cryptography from minimal assumptions. Journal of Cryptology **29**(3), 514–551 (2016) 2
30. Kitagawa, F., Matsuda, T.: Cpa-to-cca transformation for kdm security. In: Theory of Cryptography: 17th International Conference, TCC 2019, Nuremberg, Germany, December 1–5, 2019, Proceedings, Part II 17. pp. 118–148. Springer (2019) 2
31. Kitagawa, F., Matsuda, T.: Circular security is complete for kdm security. In: Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part I 26. pp. 253–285. Springer (2020) 2
32. Kitagawa, F., Matsuda, T., Tanaka, K.: Simple and efficient kdm-cca secure public key encryption. In: Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part III 25. pp. 97–127. Springer (2019) 2
33. Kitagawa, F., Matsuda, T., Tanaka, K.: Cca security and trapdoor functions via key-dependent-message security. Journal of Cryptology **35**(2), 9 (2022) 2
34. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19. pp. 388–397. Springer (1999) 1
35. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16. pp. 104–113. Springer (1996) 1
36. Koppula, V., Kumar, A., Rajasree, M.S., Swaminathan, H.: Incompressible encryption beyond CPA/CCA security. Manuscript submitted (2024) 2
37. Lewko, A., Lewko, M., Waters, B.: How to leak on key updates. In: Proceedings of the forty-third annual ACM symposium on Theory of computing. pp. 725–734 (2011) 2
38. Marcedone, A., Pass, R., Shelat, A.: Bounded kdm security from io and owf. In: International Conference on Security and Cryptography for Networks. pp. 571–586. Springer (2016) 2
39. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. pp. 18–35. Springer (2009) 2, 3, 16

40. Nishimaki, R., Yamakawa, T.: Leakage-resilient identity-based encryption in bounded retrieval model with nearly optimal leakage-ratio. In: IACR International Workshop on Public Key Cryptography. pp. 466–495. Springer (2019) 2
41. Waters, B., Wichs, D.: Universal amplification of kdm security: From 1-key circular to multi-key kdm. In: Annual International Cryptology Conference. pp. 674–693. Springer (2023) 2, 3, 7
42. Wee, H.: Kdm-security via homomorphic smooth projective hashing. In: Public-Key Cryptography–PKC 2016: 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II 19. pp. 159–179. Springer (2016) 2, 5, 6, 11, 15

9 Proof of Theorem 5

We will show this using a sequence of hybrid arguments.

G_0: This is the original LR-KDM game.

- **Initialization Phase:**
  1. The challenger randomly generates $(\mathsf{hps.pk}, \mathsf{hps.sk}) \leftarrow \mathsf{HPS.Setup}(1^\lambda)$.
  2. It sets $\mathsf{sk} := \mathsf{hps.sk}$ and sends $\mathsf{pk} := \mathsf{hps.pk}$ to the adversary $\mathcal{A}$
- **Leakage Phase:**
  1. $\mathcal{A}$ sends a function h to the challenger.
  2. The challenger responds with $h(\mathsf{sk})$.
- **Challenge Phase:**
  1. $\mathcal{A}$ sends a function $f_{e,k} \in \mathcal{F}_\lambda$ to the challenger.
  2. The challenger randomly picks $(x, w) \in \mathcal{R}_\mathcal{L}$.
  3. It sends $(x, \mathsf{HPS.Encaps}(\mathsf{pk}, (x, w)) \oplus \mathsf{HPS.Decaps}(\mathsf{sk}, e) \oplus k)$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

G_1: This is game, the challenge ciphertext is modified to $(x, \mathsf{HPS.Decaps}(\mathsf{sk}, x) \oplus \mathsf{HPS.Decaps}(\mathsf{sk}, e) \oplus k)$.

- **Challenge Phase:**
  1. $\mathcal{A}$ sends a function $f_{e,k} \in \mathcal{F}_\lambda$ to the challenger.
  2. The challenger randomly picks $(x, w) \in \mathcal{R}_\mathcal{L}$.
  3. It sends $(x, \mathsf{HPS.Decaps}(\mathsf{sk}, x) \oplus \mathsf{HPS.Decaps}(\mathsf{sk}, e) \oplus k)$ to $\mathcal{A}$.

G_2: This is game, the string x is chosen not from the language.

- **Challenge Phase:**
  1. $\mathcal{A}$ sends a function $f_{e,k} \in \mathcal{F}_\lambda$ to the challenger.
  2. The challenger randomly picks $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$.
  3. It sends $(x, \mathsf{HPS.Decaps}(\mathsf{sk}, x) \oplus \mathsf{HPS.Decaps}(\mathsf{sk}, e) \oplus k)$ to $\mathcal{A}$.

G_3: This is game, the challenge ciphertext is modified to $(x, t \oplus k)$ where t is a truly random string.

– **Challenge Phase:**
  1. $\mathcal{A}$ sends a function $f_{e,k} \in \mathcal{F}_\lambda$ to the challenger.
  2. The challenger randomly picks $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$.
  3. It sends $(x, t \oplus k)$ to $\mathcal{A}$.

G_4: This is game, the challenge ciphertext is modified to $(x, \mathsf{HPS.Decaps}(\mathsf{sk}, x))$ where t is a truly random string.

– **Challenge Phase:**
  1. $\mathcal{A}$ sends a function $f_{e,k} \in \mathcal{F}_\lambda$ to the challenger.
  2. The challenger randomly picks $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$.
  3. It sends $(x, \mathsf{HPS.Decaps}(\mathsf{sk}, x))$ to $\mathcal{A}$.

G_5: This is game, the string x is chosen from the language.

– **Challenge Phase:**
  1. $\mathcal{A}$ sends a function $f_{e,k} \in \mathcal{F}_\lambda$ to the challenger.
  2. The challenger randomly picks $(x, w) \in \mathcal{R}_\mathcal{L}$.
  3. It sends $(x, \mathsf{HPS.Decaps}(\mathsf{sk}, x))$ to $\mathcal{A}$.

G_6: This is game, the challenge ciphertext is modified to $(x, \mathsf{HPS.Encaps}(\mathsf{pk}, (x, w)))$.

– **Challenge Phase:**
  1. $\mathcal{A}$ sends a function $f_{e,k} \in \mathcal{F}_\lambda$ to the challenger.
  2. The challenger randomly picks $(x, w) \in \mathcal{R}_\mathcal{L}$.
  3. It sends $(x, \mathsf{HPS.Encaps}(\mathsf{pk}, (x, w)))$ to $\mathcal{A}$.

Analysis: Let $p_{\mathcal{A},i}$ denote the probability of $\mathcal{A}$ that outputs 0 in Game G_i. We will show that the probability in each game is close to the previous one.

Lemma 3. *Assuming* HPS *is a secure HPS, for all PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,*

$$|p_{\mathcal{A},1} - p_{\mathcal{A},0}| \leq \mathrm{negl}(\lambda)$$

$$|p_{\mathcal{A},6} - p_{\mathcal{A},5}| \leq \mathrm{negl}(\lambda)$$

Proof. The proof follows from the correctness of HPS which states that

$$\mathsf{HPS.Encaps}(\mathsf{pk}, (x, w)) = \mathsf{HPS.Decaps}(\mathsf{sk}, x)$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HPS.Setup}(1^\lambda)$ and $(x, w) \in \mathcal{R}_\mathcal{L}$.

Lemma 4. *Assuming indistinguishability property of $\mathcal{L}$, for all PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$,

$$|p_{\mathcal{A},2} - p_{\mathcal{A},1}| \leq \mathrm{negl}(\lambda)$$

$$|p_{\mathcal{A},5} - p_{\mathcal{A},4}| \leq \mathrm{negl}(\lambda)$$

Proof. The proof follows from the fact that it is computationally hard to distinguish an element from $\mathcal{L}$ from an element from $\mathcal{L} \cup \overline{\mathcal{L}}$.

Lemma 5. *Assuming* HPS *is a secure LR-HPS, for all PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$,

$$|p_{\mathcal{A},3} - p_{\mathcal{A},2}| \leq \mathrm{negl}(\lambda)$$

$$|p_{\mathcal{A},4} - p_{\mathcal{A},3}| \leq \mathrm{negl}(\lambda)$$

Proof. This follows from the leakage-resilience and homomorphism property of the LR-HPS.

Using the above lemmas and triangular inequality, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},0} - p_{\mathcal{A},6}| \leq \mathrm{negl}(\lambda)$.

10 Proof of Theorem 8

We will show the security for a single challenge ciphertext of the above scheme through a series of hybrid arguments. The proof can be modified for multi-challenge ciphertexts.

G_0: This is the LR-KDM game where the challenger picks the challenge bit $d = 1$ or equivalently, chooses to encrypt a function of the secret key.

- **Initialization Phase**
  1. The challenger randomly generates $k \leftarrow \{0,1\}^{\beta}$.
  2. It samples $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \leftarrow \mathsf{PKE.Setup}(1^{\lambda})$ for $i \in [\beta], b \in \{0,1\}$.
  3. It sets $\overline{\mathsf{pk}} = \{\mathsf{pk}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}$ and $\overline{\mathsf{sk}} = (k, \{\mathsf{sk}_{i,k_i}\}_{i\in[\beta]})$.
  4. It sends $\overline{\mathsf{pk}}$ to $\mathcal{A}$.
- **Leakage Phase**
  1. $\mathcal{A}$ sends h to the challenger.
  2. The challenger return $h(\overline{\mathsf{sk}})$ to $\mathcal{A}$.
- **Challenge Phase**
  1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_{\lambda}$.
  2. The challenger computes $(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^{\lambda}, C_m)$ where $m = f(\mathsf{sk})$.
  3. It computes $\mathsf{ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,b}, \mathsf{lab}_{i,b})$ for each $i \in [\beta], b \in \{0,1\}$.
  4. It sets $\overline{\mathsf{ct}} = (\widehat{C}, \{\mathsf{ct}_{i,b}\}_{i\in[\beta],b\in\{0,1\}})$.

5. It sends $\overline{\mathsf{ct}}$ to $\mathcal{A}$.

– **Response Phase**
1. $\mathcal{A}$ outputs $d' \in \{0,1\}$.

G_1: The challenger changes the way the secret key is sampled.

– **Initialization Phase**
1. The challenger samples $v \leftarrow \{0,1\}^\beta, k_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and sets $k = v \oplus k_{\mathsf{SKE}}$.
2. It samples $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ for $i \in [\beta], b \in \{0,1\}$.
3. It sets $\overline{\mathsf{pk}} = \{\mathsf{pk}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}$ and $\overline{\mathsf{sk}} = (k, \{\mathsf{sk}_{i,k_i}\}_{i\in[\beta]})$.
4. It sends $\overline{\mathsf{pk}}$ to $\mathcal{A}$.
5. It maintains $\widehat{\mathsf{ct}} \leftarrow \mathsf{SKE.Enc}(k_{\mathsf{SKE}}, \overline{\mathsf{sk}})$.

G_2: The challenger modifies the encryption of certain labels.

– **Challenge Phase**
1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
2. The challenger computes $(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C_m)$ where $m = f(\mathsf{sk})$.
3. It computes $\mathsf{ct}_{i,k_i} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,k_i}, \mathsf{lab}_{i,k_i})$ and $\mathsf{ct}_{i,1\oplus k_i} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,1\oplus k_i}, 0^\lambda)$.
4. It sets $\overline{\mathsf{ct}} = (\widehat{C}, \{\mathsf{ct}_{i,b}\}_{i\in[\beta],b\in\{0,1\}})$.
5. It sends $\overline{\mathsf{ct}}$ to $\mathcal{A}$.

G_3: The challenger modifies the circuit being garbled.

– **Challenge Phase**
1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
2. The challenger constructs a circuit $\mathcal{C}$ that on input x :
   (a) Computes $k' = x \oplus v$.
   (b) Computes $\mathsf{sk}' \leftarrow \mathsf{SKE.Dec}(k', \widehat{\mathsf{ct}})$.
   (c) Returns $f(\mathsf{sk}')$.
3. The challenger computes $(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C})$.
4. It computes $\mathsf{ct}_{i,k_i} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,k_i}, \mathsf{lab}_{i,k_i})$ and $\mathsf{ct}_{i,1\oplus k_i} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,1\oplus k_i}, 0^\lambda)$.
5. It sets $\overline{\mathsf{ct}} = (\widehat{C}, \{\mathsf{ct}_{i,b}\}_{i\in[\beta],b\in\{0,1\}})$.
6. It sends $\overline{\mathsf{ct}}$ to $\mathcal{A}$.

G_4: The challenger modifies the encryption of labels.

– **Challenge Phase**
  1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
  2. The challenger constructs a circuit $\mathcal{C}$ that on input x:
     (a) Computes $k' = x \oplus v$.
     (b) Computes $\mathsf{sk}' \leftarrow \mathsf{SKE.Dec}(k', \widehat{\mathsf{ct}})$.
     (c) Returns $f(\mathsf{sk}')$.
  3. The challenger computes $(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C})$.
  4. It computes $\mathsf{ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,b}, \mathsf{lab}_{i,b})$.
  5. It sets $\overline{\mathsf{ct}} = (\widehat{C}, \{\mathsf{ct}_{i,b}\}_{i\in[\beta],b\in\{0,1\}})$.
  6. It sends $\overline{\mathsf{ct}}$ to $\mathcal{A}$.

G_5: The challenger modifies the ciphertext of the secret key.

– **Initialization Phase**
  1. The challenger samples $v \leftarrow \{0,1\}^\beta, k_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and sets $k = v \oplus k_{\mathsf{SKE}}$.
  2. It samples $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ for $i \in [\beta], b \in \{0,1\}$.
  3. It sets $\overline{\mathsf{pk}} = \{\mathsf{pk}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}$ and $\overline{\mathsf{sk}} = (k, \{\mathsf{sk}_{i,k_i}\}_{i\in[\beta]})$.
  4. It sends $\overline{\mathsf{pk}}$ to $\mathcal{A}$.
  5. It maintains $\widehat{\mathsf{ct}} \leftarrow \mathsf{SKE.Enc}(k_{\mathsf{SKE}}, 0^\lambda)$.

G_6: The challenger modifies the encryption of all labels.

– **Challenge Phase**
  1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
  2. The challenger constructs a circuit $\mathcal{C}$ that on input x:
     (a) Computes $k' = x \oplus v$.
     (b) Computes $\mathsf{sk}' \leftarrow \mathsf{SKE.Dec}(k', \widehat{\mathsf{ct}})$.
     (c) Returns $f(\mathsf{sk}')$.
  3. The challenger computes $(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C})$ where $m = f(\mathsf{sk})$.
  4. It computes $\mathsf{ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,b}, 0^\lambda)$.
  5. It sets $\overline{\mathsf{ct}} = (\widehat{C}, \{\mathsf{ct}_{i,b}\}_{i\in[\beta],b\in\{0,1\}})$.
  6. It sends $\overline{\mathsf{ct}}$ to $\mathcal{A}$.

G_7: The challenger modifies the circuit being garbled.

– **Initialization Phase**
  1. The challenger samples $v \leftarrow \{0,1\}^\beta, k_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and sets $k = v \oplus k_{\mathsf{SKE}}$.
  2. It samples $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ for $i \in [\beta], b \in \{0,1\}$.

3. It sets $\overline{\mathsf{pk}} = \{\mathsf{pk}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}$ and $\overline{\mathsf{sk}} = (k, \{\mathsf{sk}_{i,k_i}\}_{i\in[\beta]})$.
4. It sends $\overline{\mathsf{pk}}$ to $\mathcal{A}$.
5. ~~It maintains $\widehat{\mathsf{ct}} \leftarrow \mathsf{SKE.Enc}(k_{\mathsf{SKE}}, 0^\lambda)$.~~

– **Leakage Phase**
1. $\mathcal{A}$ sends h to the challenger.
2. The challenger return $h(\overline{\mathsf{sk}})$ to $\mathcal{A}$.

– **Challenge Phase**
1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
2. ~~The challenger constructs a circuit $\mathcal{C}$ that on input x:~~
   (a) ~~Computes $k' = x \oplus v$.~~
   (b) ~~Computes $\mathsf{sk}' \leftarrow \mathsf{SKE.Dec}(k', \widehat{\mathsf{ct}})$.~~
   (c) ~~Returns $f(\mathsf{sk}')$.~~
3. The challenger computes $(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C_{0^\lambda})$
4. It computes $\mathsf{ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,b}, 0^\lambda)$.
5. It sets $\overline{\mathsf{ct}} = (\widehat{C}, \{\mathsf{ct}_{i,b}\}_{i\in[\beta],b\in\{0,1\}})$.
6. It sends $\overline{\mathsf{ct}}$ to $\mathcal{A}$.

– **Response Phase**
1. $\mathcal{A}$ outputs $d' \in \{0,1\}$.

G_8: The challenger modifies the encryption of the labels. This is the incompressible KDM game where the challenger picks $d = 0$ or equivalently, chooses to encrypt 0^λ.

– **Challenge Phase**
1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
2. The challenger computes $(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[\beta],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C_m)$ where $m = 0^\lambda$.
3. It computes $\mathsf{ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,b}, \mathsf{lab}_{i,b})$.
4. It sets $\overline{\mathsf{ct}} = (\widehat{C}, \{\mathsf{ct}_{i,b}\}_{i\in[\beta],b\in\{0,1\}})$.
5. It sends $\overline{\mathsf{ct}}$ to $\mathcal{A}$.

Analysis: Let $p_{\mathcal{A},i}$ denote the probability of $\mathcal{A}$ that outputs 0 in Game G_i. We will show that the success probability in each game is close to the previous one.

Lemma 6. *The equality $p_{\mathcal{A},0} = p_{\mathcal{A},1}$ holds.*

Proof. The games G_0 and G_1 are identical since the sampling distribution of k remains the same. Also, the ciphertext $\widehat{\mathsf{ct}}$ is only maintained by the challenger and is not used in the game. □

Lemma 7. *From the CPA security of* PKE*, for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},2} - p_{\mathcal{A},1}| \leq \mathrm{negl}(\lambda)$.*

Proof. Observe that the secret keys $\mathsf{sk}_{i,1\oplus k_i}$ are not used in the game after generation. We construct an adversary $\mathcal{B}$ that plays the CPA security game of PKE and acts as the challenger against the given adversary $\mathcal{A}$. The algorithm $\mathcal{B}$ queries the challenger of the CPA security game on $(\mathsf{lab}_{i,1\oplus k_i}, 0^\lambda)$ and uses it to simulate the LR-KDM game for $\mathcal{A}$. We get $|p_{\mathcal{A},2} - p_{\mathcal{A},1}|$ is bounded above by the advantage of $\mathcal{B}$ in the CPA security game of PKE which is negligible due to the assumption that PKE is CPA secure. □

Lemma 8. *From the Indistinguishibility Security of* Garble*, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*,* $|p_{\mathcal{A},3} - p_{\mathcal{A},2}| \leq \mathrm{negl}(\lambda)$.

Proof. The difference between G_3 and G_2 is that the circuit C_m is used in G_2 whereas $\mathcal{C}$ is used in G_3. Observe that the circuit C_m and $\mathcal{C}$ have the same output on k, i.e., $C_m(k) = \mathcal{C}(k)$. Also, the adversary's algorithm can only depend on $\widehat{C}$ and $\{\mathsf{lab}_{i,k_i}\}_{i\in[\beta]}$. Thus by the Indistinguishibility Security of Garble, we get the desired result. □

Lemma 9. *From the CPA security of* PKE*, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*,* $|p_{\mathcal{A},4} - p_{\mathcal{A},3}| \leq \mathrm{negl}(\lambda)$

Proof. The proof is similar to Lemma 7. □

Lemma 10. *From the LR-KDM security of* SKE*, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*,* $|p_{\mathcal{A},5} - p_{\mathcal{A},4}| \leq \mathrm{negl}(\lambda)$.

Proof. Consider an algorithm $\mathcal{B}$ that plays the LR-KDM security game of SKE. This algorithm first samples $v, \mathsf{pk}_{i,b}$ and $\mathsf{sk}_{i,b}$ for each $i \in [\beta]$ and $b \in \{0,1\}$ as specified in the initialization phase of both the games. It then queries the SKE challenger on the function which evaluates to $(x, \{\mathsf{sk}_{i,x_i}\}_{i\in[\beta]})$ on input x. This is a projective function since each output bit depends on at most one input bit. Now, it uses the output it receives from the challenger of the SKE game as $\widehat{\mathsf{ct}}$ and runs $\mathcal{A}$. On receiving the leakage function h from $\mathcal{A}$, $\mathcal{B}$ sends $h'_{\{\mathsf{sk}_{i,b}\}}$ to the challenger – $h'_{\{\mathsf{sk}_{i,b}\}}$ on input x, returns $h(x, \{\mathsf{sk}_{i,x_i}\})$. It can simulate the rest of the game, and we get $|p_{\mathcal{A},5} - p_{\mathcal{A},5}|$ is bounded above by the advantage of $\mathcal{B}$ in the LR-KDM security game of SKE. Thus since SKE is secure, we have the desired result. □

Lemma 11. *From the LR security of* PKE*, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*,* $|p_{\mathcal{A},6} - p_{\mathcal{A},5}| \leq \mathrm{negl}(\lambda)$.

Proof. We consider $2\beta + 1$ intermediate hybrids $G_{i,b}$ where $i \in [\beta], b \in \{0,1\}$ and $G_5 = G_{0,1}$ with the following description - The game $G_{i,j}$ is exactly similar to G_5, except that for all $a \leq i, b \leq j$, $\mathsf{ct}_{a,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{a,b}, 0^\lambda)$. For the rest, $\mathsf{ct}_{a,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{a,b}, \mathsf{lab}_{a,b})$.

To show that $G_{i-1,1} \approx_c G_{i,0}$ (and similarly $G_{i,0} \approx_c G_{i,1}$), we construct an adversary $\mathcal{B}$ that plays the LR security game of PKE and acts as the challenger

against the given adversary $\mathcal{A}$. Similar to Lemma 10, on receiving the leakage function h from $\mathcal{A}$, $\mathcal{B}$ will either return the leakage value if $k_i \neq 0$ (in the other case $k_i \neq 1$) or construct an appropriate function h' and sends it to the challenger and receives the leakage value. The algorithm $\mathcal{B}$ queries the LR security game on $(\mathsf{lab}_{i,0}, 0^\lambda)$ (in the other case $(\mathsf{lab}_{i,1}, 0^\lambda)$) and forwards the response to $\mathcal{A}$. Then we get $|p_{\mathcal{A},6} - p_{\mathcal{A},5}|$ is bounded above by the advantage of $\mathcal{B}$ in the LR security game of PKE. This is negligible because PKE is LR secure. □

Lemma 12. *From the garbled circuits security with no labels of* Garble*, for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},7} - p_{\mathcal{A},6}| \leq \mathrm{negl}(\lambda)$.*

Proof. It is important to note that the labels $\mathsf{lab}_{i,b}$ are never used in the game after generation. Thus, by the garbled circuits security with no labels of Garble, we get our desired result. □

Lemma 13. *From the LR- security of* PKE*, for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},8} - p_{\mathcal{A},7}| \leq \mathrm{negl}(\lambda)$.*

Proof. The proof is similar to Lemma 11. □

Using the above lemmas and triangular inequality, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},0} - p_{\mathcal{A},8}| \leq \mathrm{negl}(\lambda)$.

11 Proof of Theorem 9

We will show the security for a single challenge ciphertext of the above scheme through a series of hybrid arguments. The proof can be modified for multi-challenge ciphertexts. Here $\mathcal{F}_\lambda$ consists of circuits of a priori bounded size.

G_0: This is the LR-KDM game where the challenger picks the challenge bit $d = 1$ or equivalently, chooses to encrypt a function of the secret key.

- **Initialization Phase**
  1. The samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
  2. It sends pk to $\mathcal{A}$.
- **Leakage Phase**
  1. $\mathcal{A}$ sends h to the challenger.
  2. The challenger return $h(\overline{\mathsf{sk}})$ to $\mathcal{A}$.
- **Challenge Phase**
  1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
  2. The challenger computes $\left(\widehat{C}, \{\mathsf{lab}_i\}_{i\in[|m|]}\right) \leftarrow \mathsf{Sim}(1^\lambda, |m|, |C_m|, m)$ where $m = f(\mathsf{sk})$.
  3. It $\mathsf{pke.ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \{\mathsf{lab}_i\}_i)$.
  4. It sets $\mathsf{ct} := (\hat{C}, \mathsf{pke.ct})$

5. It sends ct to $\mathcal{A}$.

– **Response Phase**
1. $\mathcal{A}$ outputs $d' \in \{0,1\}$.

G_1: In this game, $\hat{C}$ and its associated label are modified as follows.

– **Initialization Phase**
1. The samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
2. It sends pk to $\mathcal{A}$.

– **Leakage Phase**
1. $\mathcal{A}$ sends h to the challenger.
2. The challenger return $h(\overline{\mathsf{sk}})$ to $\mathcal{A}$.

– **Challenge Phase**
1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
2. The challenger computes $\left(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[|m|],b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C_m)$ where $m = f(\mathsf{sk})$.
3. It $\mathsf{pke.ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \{\mathsf{lab}_{i,m_i}\}_i)$.
4. It sets $\mathsf{ct} := (\hat{C}, \mathsf{pke.ct})$
5. It sends ct to $\mathcal{A}$.

– **Response Phase**
1. $\mathcal{A}$ outputs $d' \in \{0,1\}$.

G_2: In this game, circuit C_m is replaced with the circuit C_f such that $C_f(x) = f(x)$.

– **Initialization Phase**
1. The samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
2. It sends pk to $\mathcal{A}$.

– **Leakage Phase**
1. $\mathcal{A}$ sends h to the challenger.
2. The challenger return $h(\overline{\mathsf{sk}})$ to $\mathcal{A}$.

– **Challenge Phase**
1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
2. The challenger computes $\left(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[|\mathsf{sk}|],b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C_f)$ where $C_f(x) = f(x)$.
3. It $\mathsf{pke.ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \{\mathsf{lab}_{i,\mathsf{sk}_i}\}_i)$.
4. It sets $\mathsf{ct} := (\hat{C}, \mathsf{pke.ct})$
5. It sends ct to $\mathcal{A}$.

– **Response Phase**
1. $\mathcal{A}$ outputs $d' \in \{0,1\}$.

G_3: In this game, $\mathsf{pke.ct}$ is an encryption of 0 message rather than labels.

- **Initialization Phase**
  1. The samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
  2. It sends pk to $\mathcal{A}$.
- **Leakage Phase**
  1. $\mathcal{A}$ sends h to the challenger.
  2. The challenger return $h(\overline{\mathsf{sk}})$ to $\mathcal{A}$.
- **Challenge Phase**
  1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
  2. The challenger computes $\left(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[|m|],b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C_f)$ where $C_f(x) = f(x)$.
  3. It $\mathsf{pke.ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathbf{0})$.
  4. It sets $\mathsf{ct} := (\hat{C}, \mathsf{pke.ct})$
  5. It sends ct to $\mathcal{A}$.
- **Response Phase**
  1. $\mathcal{A}$ outputs $d' \in \{0, 1\}$.

G_4: In this game, the circuit C_f is replaced with C_0.

- **Initialization Phase**
  1. The samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
  2. It sends pk to $\mathcal{A}$.
- **Leakage Phase**
  1. $\mathcal{A}$ sends h to the challenger.
  2. The challenger return $h(\overline{\mathsf{sk}})$ to $\mathcal{A}$.
- **Challenge Phase**
  1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
  2. The challenger computes $\left(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[|m|],b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C_0)$ where $C_0 = 0$.
  3. It $\mathsf{pke.ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathbf{0})$.
  4. It sets $\mathsf{ct} := (\hat{C}, \mathsf{pke.ct})$
  5. It sends ct to $\mathcal{A}$.
- **Response Phase**
  1. $\mathcal{A}$ outputs $d' \in \{0, 1\}$.

G_5: In this game, $\mathsf{pke.ct}$ is replaced with the encryption of the labels.

- **Initialization Phase**
  1. The samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
  2. It sends pk to $\mathcal{A}$.
- **Leakage Phase**
  1. $\mathcal{A}$ sends h to the challenger.

2. The challenger return $h(\overline{\mathsf{sk}})$ to $\mathcal{A}$.

– **Challenge Phase**
1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
2. The challenger computes $\left(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[|m|],b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C_0)$ where $C_0 = 0$.
3. It $\mathsf{pke.ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \{\mathsf{lab}_{i,0}\}_i)$.
4. It sets $\mathsf{ct} := (\hat{C}, \mathsf{pke.ct})$
5. It sends ct to $\mathcal{A}$.

– **Response Phase**
1. $\mathcal{A}$ outputs $d' \in \{0, 1\}$.

G_6: In this game, $\hat{C}_0$ and its associated labels are simulated.

– **Initialization Phase**
1. The samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
2. It sends pk to $\mathcal{A}$.

– **Leakage Phase**
1. $\mathcal{A}$ sends h to the challenger.
2. The challenger return $h(\overline{\mathsf{sk}})$ to $\mathcal{A}$.

– **Challenge Phase**
1. $\mathcal{A}$ sends a function $f \in \mathcal{F}_\lambda$.
2. The challenger computes $\left(\widehat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[|m|],b\in\{0,1\}}\right) \leftarrow \mathsf{Sim}(1^\lambda, |\mathbf{0}|, |C_0|, \mathbf{0})$ where $C_0 = 0$.
3. It $\mathsf{pke.ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \{\mathsf{lab}_{i,0}\}_i)$.
4. It sets $\mathsf{ct} := (\hat{C}, \mathsf{pke.ct})$
5. It sends ct to $\mathcal{A}$.

– **Response Phase**
1. $\mathcal{A}$ outputs $d' \in \{0, 1\}$.

Analysis: Let $p_{\mathcal{A},i}$ denote the probability of $\mathcal{A}$ that outputs 0 in Game G_i. We will show that the success probability in each game is close to the previous one.

Lemma 14. *Assuming* GC *is a secure garbling scheme, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},1} - p_{\mathcal{A},0}| \leq \mathrm{negl}(\lambda)$.

Proof. This directly follows from the simulation security of GC (see Definition 2).

Lemma 15. *Assuming* GC *is a secure garbling scheme, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},2} - p_{\mathcal{A},1}| \leq \mathrm{negl}(\lambda)$.

Proof. This directly follows from the security of GC (see Theorem 1).

Lemma 16. *Assuming* PKE *is a secure LR-KDM scheme for projection functions, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*,* $|p_{\mathcal{A},3} - p_{\mathcal{A},2}| \leq \mathrm{negl}(\lambda)$.

Proof. This follows from the LR-KDM for projection functions security of the PKE scheme. Observe that the each label lab_{i,m_i} depends only on the i bit m_i. Therefore, the message $\{\mathsf{lab}_{i,\mathsf{sk}_i}\}_i$ is the output of a projection function.

Lemma 17. *Assuming* GC *is a secure garbling scheme, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*,* $|p_{\mathcal{A},3} - p_{\mathcal{A},4}| \leq \mathrm{negl}(\lambda)$.

Proof. This directly follows from the security of GC (see Theorem 2).

Lemma 18. *Assuming* PKE *is a secure CPA scheme for projection functions, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*,* $|p_{\mathcal{A},5} - p_{\mathcal{A},4}| \leq \mathrm{negl}(\lambda)$.

Proof. This directly follows from the CPA security of PKE.

Lemma 19. *Assuming* GC *is a secure garbling scheme, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*,* $|p_{\mathcal{A},6} - p_{\mathcal{A},5}| \leq \mathrm{negl}(\lambda)$.

Proof. This directly follows from the simulation security of GC (see Definition 2).