Stronger Variants of KDM Security

Thesis submitted by

Dhairya Gupta
2019CS50428

under the guidance of **Prof. Venkata Koppula, Indian Institute of Technology Delhi**

in partial fulfilment of the requirementsfor the award of the degree of

Bachelor and Master of Technology

Department Of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY DELHI

June 2024

THESIS CERTIFICATE

This is to certify that the thesis titled **Stronger Variants of KDM Security**, submitted by **Dhairya Gupta (2019CS50428)**, to the Indian Institute of Technology, Delhi, for the award of the degree of **Bachelor and Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Venkata Koppula

Professor

Dept. of Computer Science

IIT-Delhi, 600 036

ACKNOWLEDGEMENTS

First and foremost, I would like to express profound gratitude towards my thesis advisor, **Prof. Venkata Koppula**, for their patience, enthusiasm and vast knowledge throughout this journey, His insights, dedication and feedback towards me. He consistently shaped my ideas and nudged me in the right direction whenever I needed it.

I am also immensely greatful to **Mahesh Sreekumar Rajasree** for his creative insights, long and fruitful discussions and constructive feedback which greatly improved the quality of my work.

I am also thankful to my friends and colleagues **Aman Verma** and **Abhinav Kumar** for our collaborative discussions, late-night study sessions and making the past 5 years of college fun and memorable.

Finally, I am forever indebted to my parents, for their constant encouraging words and unwavering belief in me. Their endless support and dedication to my education laid the foundation upon which I built my personal and academic achievements.

ABSTRACT

We present new construction for Key Dependent Message (KDM) secure Public Key Encryption(PKE) schemes in the Incompressible[JWZ22, BDD22] setting as well as leakage-resilient KDM in the presence of adversarial bounded leakage of the bits of the secret key[AGV09] via Hash Proof Systems[CS01, NS09]. In particular, we present the following constructions:

- **Incompressible KDM PKE in random oracle(RO) model:** We show a construction for incompressible KDM SKEs in RO and using incompressible KDM SKE and a CPA-secure PKE scheme, we show a construction for incompressible KDM PKEs for any general class of functions
- **Transformation to Incompressible KDM CCA2 secure:** We construct an Incompressible KDM secure CCA encryption scheme via a Non-Interactive Zero Knowledge proving scheme[GOS12], a one-time secure Incompressible projective-KDM SKE[KKRS24], an incompressible CCA secure PKE[BDD22], an incompressible CPA secure PKE[JWZ22] and a Garbling scheme[Yao86]
- **Leakage Resilient - KDM from Hash Proof Systems** We show a construction for a homomorphic leakage-resilient smooth Hash Proof System[CS01, Wee15] from d-LIN assumptions, and construct a leakage-resilient KDM public key encryption scheme via homomorphic LR-smooth HPS for the class of affine functions over the bits of the secret key.

Contents

Chapter 1

INTRODUCTION

1.1 Overview

Symmetric Key Encryption: We trace back the earliest works of cryptography to symmetric key encryption (SKE), which allowed for some information to be securely hidden in a cipher using a "secret" key which was necessary to later decrypt the information. In symmetric key encryption, the sender encrypts the message and the receiver decrypts the message with the same "secret" key.

The earliest symmetric key encryption schemes can be traced back to ancient times, such as substitution ciphers where the secret information was the mapping of letters from plaintext to ciphertext. As time progressed and the need for stronger encryption schemes grew, stronger encryption schemes were discovered. The earliest groundbreaking contribution to modern day symmetric encryption was the One Time Pad, which is theoretically unbreakable when the key is completely random and used only once. However, it has many practical limitations such as large key sizes and non-reusability. In the computing age, SKE schemes such as DES and AES had widespread applications for systems and real-world protocols and various SKE schemes continue to be used to this day.

Public Key Encryption: However, with technological advancements, the limitations of symmetric key encryption became apparent. Managing a large collection of secret keys became impractical for communicating parties as well as it was cumbersome to ensure the secure distribution of keys, which was necessary for sharing symmetric keys between intended parties. This led to the rise of public key encryption(PKE) schemes such as RSA encryption[RSA78] and Elliptic Curve Cryptosystems. Public key encryption schemes notably had a pair of keys $(\mathsf{pk}, \mathsf{sk})$ where pk was the public key which did not need to be hidden from malicious parties and could be used to encrypt any message, while decryption of encrypted messages could only be done using a secret key sk. This allowed for secure communication across an insecure channel even if the sending party did not have access to sk.

These schemes are conventionally secure against Chosen Plaintext Attacks. We briefly mention the simplest formal notion for secure encryption for public key and symmetric key encryption.

CPA-SKE secure schemes A scheme $(\mathsf{Enc}, \mathsf{Dec})$ is said to be CPA-SKE secure if for a key s generated from some distribution $\mathcal{S}$, on an adversary seeing an encryption for any message

$\mathsf{Enc}(s, m)$, it cannot learn any (significant) amount of information about the underlying message. We define this more formally through the following mathematical game: For any pair of messages (m_0, m_1) chosen by the adversary, if we choose a message uniformly randomly from (m_0, m_1) and show the adversary its encryption: $\mathsf{Enc}(s, m)$, the adversary cannot correctly guess with probability more than $\frac{1}{2}$ + some negligible quantity the message which was encrypted. We can also consider a multi-query version of this game

CPA-PKE secure schemes A scheme $(\mathsf{Enc}, \mathsf{Dec})$ is said to be CPA-PKE secure if for a key-pair $(\mathsf{pk}, \mathsf{sk})$ generated from some distribution, seeing an encryption for any message $\mathsf{Enc}(\mathsf{pk}, m)$ cannot help the adversary learn any (significant) amount of information about the underlying message. We define this more formally through the following game: Given the public key pk associated with $(\mathsf{Enc}, \mathsf{Dec})$ [which in turn allows the adversary to calculate the ciphertexts to multiple message queries of its choice] and for any pair of messages (m_0, m_1) chosen by the adversary: On being given the encryption to one of them chosen randomly, the adversary still cannot guess with probability significantly better than $\frac{1}{2}$, which message's encryption was given

CCA security Most of the encryption notions talked about above ensure that the encryption is secure as long as the adversary can encrypt plaintexts of his choice but may be vulnerable in cases when the adversaries can manipulate ciphertexts and view their decryptions. During the 1990s, Bleichenbacher [Ble98] was able to show an attack on the RSA encryption used in SSL/TLS. By sending carefully modified ciphertexts to the SSL server and getting notified of whether the server decrypted successfully, Bleichenbacher was able to obtain the session key and hence, break the security of RSA. Such attacks which were able to exploit vulnerabilities during decryption, as well as ongoing constructions in literature at the time, gave rise to the stronger security notion of CCA secure encryption[NY90, RS91].

We define the CCA notion for public key encryption schemes. Note that CCA-PKE security is strictly stronger than CPA-PKE which is easy to see from its formal description. Formally, we describe CCA-PKE security through the following game: the adversary is given access to a decryption oracle throughout the CCA-PKE game, which it can query without any restrictions(assuming a computational adversary can make only polynomially many queries). The remainder of the game is identical to the CPA-PKE game, the only difference being access to the decryption oracle. A scheme $(\mathsf{Enc}, \mathsf{Dec})$ is a CCA-secure PKE if in this game, the adversary cannot guess the final bit correctly with probability more than $\frac{1}{2}$

KDM secure encryption We have seen that the most common notion of secure encryption and arguably the simplest in cryptography literature is semantically secure encryption against standard plaintext attacks(CPA-secure)[BDJR97, GM84]. Over the years, encryption schemes against increasingly stronger adversaries have been studied such as the notion of CCA which arose due to vulnerabilities in CPA. Nonetheless, most widely used notions of semantically secure encryption are restricted to the setting where the plaintext message

being encrypted is independent of the secret decryption key. However, there has been a growing interest in studying the case where the message being encrypted is indeed dependent on the secret decryption keys. Such notions become necessary in increasingly complex systems, such as disk encryption schemes, where the secret decryption key is naturally a part of the information being stored on disk. A widely used example was the BitLocker disk encryption utility used on Windows Vista. In accordance with commonplace literature terminology, we call such encryption schemes as KDM-secure. KDM security is also a crucial building block in constructions for various cryptographic primitives such as "bootstrapping" in fully homomorphic encryption[Gen09].

Note that a PKE scheme Formally, a SKE scheme is KDM-CPA secure if instead of the encryption queries encrypting messages(this is the case of CPA secure SKE) as queried by the adversary, the encryption queries return encryption of a certain function of the secret key. To draw a parallelism between CPA and KDM-CPA SKE, we replace all encryption queries m which returned $\mathsf{Enc}(s, m)$ with encryption queries f which returns $\mathsf{Enc}(s, f(s))$. Similarly, for PKE vs KDM-PKE CPA, instead of querying for m and getting $\mathsf{Enc}(\mathsf{pk}, m)$, we query for f and in return get $\mathsf{Enc}(\mathsf{pk}, f(\mathsf{sk}))$. Any KDM-CPA secure scheme is also a CPA secure scheme, simply because we can choose the function $f(\mathsf{sk})$ to return a constant message m, which is equivalent to querying for the regular encryption of m.

The earliest construction for a KDM secure encryption scheme was the work on anonymous credential systems[CL01] which looked into encrypting a shifted set of n secret keys $(\mathsf{sk}_1, \mathsf{sk}_2 \ldots \mathsf{sk}_n)$ under their own public keys $(\mathsf{pk}_1, \mathsf{pk}_2, \ldots \mathsf{pk}_n)$ as the following
$\mathsf{ct} = (\mathsf{Enc}(\mathsf{pk}_1, \mathsf{sk}_2), \mathsf{Enc}(\mathsf{pk}_2, \mathsf{sk}_3), \ldots, \mathsf{Enc}(\mathsf{pk}_n, \mathsf{sk}_1))$. Similarly, [BRS02] proposed the earliest construction for KDM SKE. Both these constructions are in the Random Oracle model. Boneh et al[BHHO08] gave the first known construction of KDM secure PKE from DDH assumptions. While the notions of KDM secure encryption and CCA might seem unrelated, they are not as indpendent as they seem to be. A recent interesting work which connects the two. A recent work by Koppula, Waters[KW18] has showed a transition from CPA secure PKE to CCA secure PKE via a stronger variant of regular PRGs known as Hinting PRGs. A similar later work by Kitagawa, Matsuda and Tanaka [KMT19] showed how to construct CCA secure schemes from CPA secure schemes via a KDM secure SKE scheme. We also have constructions for KDM-CCA secure PKE schemes via KDM-CPA secure SKEs[Wee15]. Here a KDM-CCA PKE is a scheme which satisfies CCA security where the messages in encryption queries may also be functions of the secret key.

Incompressible Encryption Another seemingly unrelated yet fascinating line of work in cryptography literature deals with the notion of "incompressible" encryption[Dzi06, JWZ22]. In the modern age, even in the presence of comprehensive measures such as Multi-Factor Authentication and robust encryption schemes, targeted and increasingly sophisticated cyber-attacks as well as poor management of private credentials by institutions often leave user credentials vulnerable[Gof23]. This compels us to ask the following question: Is it possible

that the encrypted messages remain secure even when the secret decryption key is leaked at some later point in time? Clearly this is impossible in the standard model as the adversary can simply store the ciphertext in its entirety and later use the decryption key to decrypt to the original message. One might think of a "store and decrypt later" adversary which stores the ciphertext and waits for the secret decryption key to leak. However, in real-world scenarios, it can be prohibitively expensive and inconvenient for the adversary to store large ciphertexts for potentially large periods of time while the adversary waits for the decryption key to leak. In the presence of high-speed internet and gigantic bandwidth and transmission capabilities, several terabytes of data being received and transmitted by the user in a given day exacerbates and highlights the long-term storage bounds of computational adversaries even further, making such a notion of security sensible and practical. Formally, we define a scheme to be incompressible if an adversary is not able to learn any information significantly better than random guessing assuming that the adversary can only store a bounded number of bits of information for any encrypted ciphertext[JWZ22]

The earliest works on incompressible Encryption was introduced via the all-or-nothing encryption framework put forward by Rivest[Riv97] which dealt with the block encryption framework. The property of all-or-nothing security was that determining information about any message block necessarily required decrypting all ciphertext blocks, which does capture the notion of the ciphertext being incompressible. Dziembowski[Dzi06] defined the notion of Incompressible SKEs and gave a construction using randomness extractors. [JWZ22] introduced Incompressible CPA-PKE constructions with similar message and ciphertext sizes. [BDD22] extended their work to Incompressible CCA-PKE.

Leakage-Resilient Encryption Finally, we also mention the notion of security in the presence of leakage of bits of the secret key s. We call a SKE scheme Leakage-Resilient if for some adversarially chosen leakage function h with function output size $\leq S$, the SKE scheme is CPA secure even when the adversary has $h(s)$. A similar notion for leakage-resilience can be defined for a PKE. [AGV09] Akavia et al showed how to construct leakage-resilient PKE encryption where the leakage function was chosen by the adversary as described in the definition for Leakage-Resilience PKE game above. Naor and Segev's work [NS09] showed how to build leakage resilient encryption schemes via hash proof systems.

We are especially interested in Leakage-Resilient secure encryption schemes while we study Incompressible encryption because Leakage-Resilient cryptography is, in some sense, a dual of Incompressible cryptography. For example, consider the case of public key encryption: In LR-PKE, the adversary sees the following in its view after making an encryption query and receiving ct :$(h(\mathsf{sk}), \mathsf{ct})$ where h is adversarially chosen and $|h(\mathsf{sk})| < S$ for some bound S. Compare this with incompressible encryption: After an encryption query whose output is ct: the final adversary's view (after sk is leaked) is $(\mathsf{sk}, f(\mathsf{ct}))$ where f is adversarially chosen and $|f(\mathsf{ct})| < S$ for some bound S. Thus, the leakage-resilient security is limiting the information seen for sk, while incompressible schemes bound the information seen for ct.

1.2 Our Results:

While KDM security ensures the robustness of encryption schemes even when the plaintext messages depend on the secret keys, incompressible encryption mandates that an adversary must store the entire ciphertext to gain any meaningful information upon key leakage. This thesis explores the novel intersection of these two notions. Along with incompressible encryption, we also look into the dual analogue of incompressibility which is Leakage-Resilient encryption. Our work is a part of a series of constructions in the incompressible encryption regime. We would like to mention that the current thesis is one of many works in the incompressible secure encryption domain. Particularly, we mention that recent work by Koppula et al[KKRS24] has shown the existence of constructions of Incompressible KDM secure encryption in the standard model for both SKE (via randomness extractors) and PKE(via Incompressible KDM SKE). Note that these constructions are KDM secure for circuits of projective functions or a-priori bounded size. We provide the following contributions in our thesis:

- **Incompressible KDM SKE in the Random Oracle model:** We construct an incompressible KDM SKE scheme in the Random Oracle Model using no other primitives over the universal class of functions. Most KDM schemes in the standard model are KDM secure w.r.t a certain class of functions. However, in the Random Oracle Model, our scheme is Incompressible SKE secure for any class of functions with some fixed input and output size

- **Incompressible KDM PKE in the Random Oracle model:** In the Random Oracle model, using a standard CPA PKE scheme and an incompressible KDM SKE scheme(as mentioned above) can be used to build an incompressible KDM PKE scheme which is KDM secure for any class of functions with some fixed input and output size.

- **Incompressible KDM CCA:** We construct an Incompressible KDM secure CCA encryption scheme via a Non-Interactive Zero Knowledge proving scheme, a one-time Incompressible projective-KDM SKE, an incompressible CCA secure PKE, an incompressible CPA secure PKE and a Garbling scheme. Firstly, we look at each of these primitives: NIZKs for NP-languages can be obtained through various assumptions such as trapdoor permutations[Gro10] and d-LIN assumptions[GOS12]. Recent work by Koppula et al[KKRS24] has shown the existence of a projective KDM Incompressible SKE via a modification to Dziembowski's scheme[Dzi06] for constructing Incompressible SKEs. Recent works in incompressible cryptography give us rate-1 CPA secure Incompressible PKE[JWZ22] and CCA secure incompressible PKE[BDD22] schemes. We also use Yao's garbled circuits[JW16]. Our scheme is a modification of the work on CPA-to-CCA transformation of KDM security[KM19a]

- **LR-KDM from Hash Proof Systems:** We construct a LR-smooth HPS from d-LIN assumptions which satisfies homomorphism, leakage-resilient smoothness. Then, we construct a Leakage-Resilient KDM secure Public Key Encryption scheme for the class of affine functions over the bits of the secret key. Our ideas are based on hash proof system[CS01] and the transformation from HPS to KDM[Wee15]

Chapter 2

Technical Overview

In this chapter, we give an outline of the constructions and their security proofs in brief. In the first section, we talk about construct Incompressible KDM PKE via Incompressible KDM SKE in the RO model, as well as discuss the construction for Incompressible KDM SKE in the RO model. In the second section, we discuss our construction for Incompressible KDM CCA via weaker Incompressible primitives and NIZKs. Finally, in section 3, we discuss the construction of Leakage-Resilient Smooth Hash Proof Systems from DDH and LR KDM secure encryption via Leakage-Resilient Smooth Hash Proof systems

2.1 Incompressible KDM encryption in the RO model

Recall that the a construction in RO model has one or more deterministic functions that the adversary has oracle access to and that behave like truly random functions. We note that our construction in RO is KDM secure for all classes of functions without any restrictions.

Define two random oracles $G : \{0,1\}^{l_k} \times \{0,1\}^{\mathsf{poly}(\lambda)} \longrightarrow \{0,1\}^n, H : \{0,1\}^{l_k} \times \{0,1\}^n \longrightarrow \{0,1\}^{\mathsf{poly}(\lambda)}$. We claim the following encryption scheme is a KDM Incompressible SKE:

- $\mathsf{KeyGen}(1^\lambda, 1^S)$: Sample a uniform key $k \in \{0,1\}^{\mathsf{poly}(\lambda)}$. Return k.
- $\mathsf{Enc}(k, m)$: Choose a random $r \in \{0,1\}^{\mathsf{poly}(\lambda)}$. Let $d = G(k, r) \oplus (m)$. Let $c = H(k, d) \oplus r$. Return $\mathsf{ct} = (c, d)$
- $\mathsf{Dec}(k, \mathsf{ct} = (c, d))$: Compute $r = H(k, d) \oplus c$. Compute $m = G(k, r) \oplus d$. Return m.

We concisely define the (one-time) KDM Incompressible SKE game below:

- The challenger samples secret k and a challenge bit b.
- It receives f_0, f_1 from $\mathcal{A}_1$ and auxiliary information
- Challenger sends $\mathsf{ct} = \mathsf{Enc}(k, f_b(k))$ to $\mathcal{A}_1$, receives st from $\mathcal{A}_1$ and sends $(\mathsf{st}, f_0, f_1, k)$ and auxiliary information to $\mathcal{A}_2$ which guesses the final bit b

Here we describe how the hybrids change

1. In hybrid G_0, we calculate $\mathsf{Enc}(k, f_b(k))$ as $\mathsf{ct} = (d = G(k,r) \oplus f_b(k), c = H(k,d) \oplus r)$
2. In hybrid G_1, we assume that $\mathcal{A}_1$ never queries $H(k,x)$ or $G(k,x)$. This is because H and G are random functions, hence the output (c,d) has no information about k by the security of one-time pad in the view of $\mathcal{A}_1$.
3. In hybrid G_2, we switch to a table simulating the function and assume that c and d are sampled randomly, and set $G(k,r) = d \oplus f_k$ and $H(k,d) = c \oplus r$. The distribution in G_1, G_2 are identical
4. In hybrid G_3, we assume $H(k,d)$ must be queried before any query to $G(k,r)$ is made by $\mathcal{A}_2$(which has the key k). This follows because if $G(k,r)$ is queried before, then the adversary $\mathcal{A}_2$ was able to guess r without accessing the only place where any information about r is stored which is the table entry for $H(k,d) = c \oplus r$.
5. In hybrid G_4, since $G(k,r)$ has not been queried, d is information-theoretically hidden from the adversary's view. Hence the query $H(k,d)$ cannot be made with a non-negligible probability by $\mathcal{A}_2$
6. Finally, we have in G_4 that the output (c,d) is completely random as no query of the form $H(k,x), G(k,x)$ has been made. Hence, b is information theoretically hidden in G_4

The above sequence of hybrids show that the above scheme is KDM Incompressible SKE secure. Similarly, the KDM Incompressible PKE game is analogous to the KDM incompressible SKE game. Below we show the construction for a KDM Incompressible PKE scheme in RO via KDM Incompressible SKE and CPA-PKE:

- $\mathsf{KeyGen}(1^\lambda, 1^S)$: Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE'.KeyGen'}(1^\lambda)$. Return $(\mathsf{pk}, \mathsf{sk})$
- $\mathsf{Enc}(\mathsf{pk}, m)$: Sample uniformly random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$. Compute $c \leftarrow \mathsf{PKE'.Enc}(\mathsf{pk}, r)$ and $d \leftarrow \mathsf{KdmIncSKE.Enc}(H'(r), m)$. Return $\mathsf{ct} = (c,d)$
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (c,d))$: Compute $r = \mathsf{PKE'.Dec'}(\mathsf{sk}, c)$. Return $m = \mathsf{KdmIncSKE.Dec}(H'(r), d)$

We describe how the hybrids change in the security proof for the above PKE scheme

1. In G_1, we assume that $\mathcal{A}_1$ does not query $H'(r)$ where r is the randomness sampled by the challenger. Clearly d is independent of r as the output of $H'(r)$ is independent of r. Hence, if I can distinguish between G_0, G_1, I must be able to get information about r via PKE', which violates the CPA security of PKE
2. In G_2, replace $H'(r)$ with a random k, as they are identical.
3. Now, if an adversary can win in G_2 with non-negligible advantage, it must be due to d, as c had no information about r. But $\mathsf{KdmIncSKE}$ is a secure KDM Incompressible SKE. Hence, the Incompressible PKE game in G_2 cannot be won with a non-negligble advantage as then a reduction can win the security game for KDM Incompressible SKE.

The above sequence of hybrids show that the above scheme is KDM Incompressible PKE secure.

2.2 Incompressible KDM CCA

We construct an Incompressible KDM secure CCA encryption scheme via a Non-Interactive Zero Knowledge proving scheme, a one-time Incompressible projective-KDM SKE, an incompressible CCA secure PKE, an incompressible CPA secure PKE and a Garbling scheme. Firstly, we look at each of these primitives:
NIZKs for NP-languages can be obtained through various assumptions such as trapdoor permutations[Gro10] and d-LIN assumptions[GOS12]. Recent work by Koppula et al[**?**] has shown the existence of a projective KDM Incompressible SKE via a modification to Dziembowski's scheme[Dzi06] for constructing Incompressible SKEs. Recent works in incompressible cryptography give us rate-1 CPA secure Incompressible PKE[JWZ22] and CCA secure incompressible PKE[BDD22] schemes. We also use Yao's garbled circuits[JW16]. Our scheme is a modification of the work on CPA-to-CCA transformation of KDM security[KM19a]

We describe the $\mathsf{IncKDMCCA}$ scheme in brief: Assume that IncPKE is an incompressible CPA secure PKE, PRJ is a projection secure KDM Incompressible SKE, IncCCA is an incompressible CCA2 PKE, NIZK is a non-interactive zero knowledge protocol for NP language
$\mathcal{L} = \big\{(\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}} | \forall i \in [l_s] : \exists(\mathsf{lab}_i, r_{i,0}, r_{i,1}) :$
$\mathsf{ct}_{i,0} = \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,0}, \mathsf{lab}_i; r_{i,0}), \mathsf{ct}_{i,1} = \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,1}, \mathsf{lab}_i; r_{i,1})\big\}$
and $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ be a secure garbling scheme $\mathsf{KeyGen}(.)$: we sample $2 * l_s$ pairs of keys as $(\mathsf{pk}_{i,\gamma}, \mathsf{sk}_{i,\gamma}) \leftarrow \mathsf{IncPKE.KeyGen}(.)$, we sample $(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}) \leftarrow \mathsf{IncCCA.KeyGen}(.)$, $\mathsf{crs} \leftarrow \mathsf{NIZK.K}(.), s \leftarrow \mathsf{PRJ.KeyGen}(.), \mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}(s, (\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}))$. Finally $\mathsf{IncKDMCCA.sk} := s$ and $\mathsf{IncKDMCCA.pk} := (\mathsf{ct}_{\mathsf{sk}}, (\mathsf{pk}_{i,\gamma}, \mathsf{pk}_{\mathsf{CCA}}))$ $\mathsf{Enc}(\mathsf{pk}, m)$:

- sample $(\tilde{C}_m, (\mathsf{lab}_i)_{i\in[l_s]}) \leftarrow \mathsf{Sim}(1^\lambda, |C_m|, m)$
- $\forall(i,\gamma) \in [l_s] \times \{0,1\}$: sample $r_{i,\gamma} \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,\gamma}, \mathsf{lab}_i; r_{i,\gamma})$
- Let $x = (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$
- Let $w = (\mathsf{lab}_i, r_{i,0}, r_{i,1})_{i\in[l_s]}$
- $\Pi \leftarrow \mathsf{NIZK.P}(\mathsf{crs}, x, w)$
- Finally return

$$\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, ((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_m), \Pi)\big)$$

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_{\mathsf{CCA}})$:

- $\big(\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\big) = \mathsf{PRJ.Dec}(s, \mathsf{ct}_{\mathsf{sk}})$
- $\big((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_m), \Pi\big) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct}_{\mathsf{CCA}})$
- Let $x = (pk_{i,\gamma}, ct_{i,\gamma})$. If $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) \neq 1$: return $\perp$

- $\forall i \in [l_s]$:
  $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,s_i}, \mathsf{sk}_{i,s_i}, ct_{i,s_i})$
- Return $\mathsf{Eval}(\tilde{C}_m, (\mathsf{lab}_i)_{i\in[l_s]})$

Correctness is easy to verify. For proof of security, see the hybrid transitions below:

- $\mathbf{G_1}$: switch to the hybrid which simulates the proof and key generation for NIZK. This step removes the dependence of w on Π. In fact w is not needed anymore. This transition is valid because of zero-knowledge property of NIZK.
- $\mathbf{G_2}$: switch to Garble instead of circuit and labels generated by Sim. Transition is valid due to security of (Garble, Eval, Sim).
- $\mathbf{G_3}$: for $ct_{i,1\oplus s_i}$ which were earlier encryptions of lab_{i,s_i}, now switch to encryption of $\mathsf{lab}_{i,1\oplus s_i}$. This step is valid by the Incompressible PKE security of IncPKE. Also, this step removes the dependence of encryption queries and $\mathsf{ct}_{i,\gamma}$ on s.
- $\mathbf{G_4}$: In decryption switch from $lab_i = \mathsf{IncPKE.Dec}(pk_{i,s_i}, sk_{i,s_i}, ct_{i,s_i})$ to $lab_i = \mathsf{IncPKE.Dec}(pk_{i,0}, sk_{i,0}, ct_{i,0})$. This transition will work due to the soundness property of NIZK. For a valid proof Π which is accepted, $\mathsf{IncPKE.Dec}(pk_{i,0}, sk_{i,0}, ct_{i,0}) = \mathsf{IncPKE.Dec}(pk_{i,1}, sk_{i,1}, ct_{i,1})$ must hold $\forall i$ if the elements are in language $\mathcal{L}$. This step removes the dependence of decryption queries on s.
- $\mathbf{G_5}$: We move from $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}(s, (\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}))$ to $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}(s, 0)$. This is valid because of Incompressible Projection KDM SKE security of PRJ. This step removes the dependence of s in KeyGen.
- $\mathbf{G_6}$: For CCA encryption, we switch to an encryption of $\mathbf{0}$. This is now feasible as the dependence of the CCA step on s is not there at all. This transition is valid because of Incompressible CCA security of KDM.
- $\mathbf{G_7}$: Finally we revert to honest proofs for NIZK instead of simulated proofs. The transition holds valid due to zero-knowledgeness, We also move to this hybrid to show $\mathsf{IncPKE.Dec}(pk_{i,s_i}, sk_{i,s_i}, ct_{i,s_i})$ to $lab_i = \mathsf{IncPKE.Dec}(pk_{i,0}, sk_{i,0}, ct_{i,0})$ using soundness of NIZK. Note that, since in G_7 the challenge bit is information-theoretically hidden from the adversary's view, an adversary cannot win in G_7 with any advantage.

From the above claims, we can see that IncKDMCCA is an Incompressible KDM CCA secure encryption scheme

2.3 LR-KDM

In this section, we provide a brief overview for LR-smooth homomorphic hash proof systems and LR-KDM secure PKE schemes:

2.3.1 Leakage-Resilient Hash Proof System

A hash proof system $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$ associated with the 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 $\mathsf{k} \in \mathcal{L}'$
- $\mathsf{Decaps}(\mathsf{sk}, x)$: The decoding algorithm takes as input the secret key sk, an input string $x \in \mathcal{L} \cup \overline{\mathcal{L}}$ and string and outputs a string $\mathsf{k} \in \mathcal{L}'$.

A leakage-resilient (homomorphic) hash proof system must satisfy the following properties:

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

$$\Pr[\mathsf{Encaps}(\mathsf{pk}, (x, w)) = \mathsf{Decaps}(\mathsf{sk}, x)] \geq 1 - \mathrm{negl}(\lambda)$$

Homomorphism For all $\lambda : \forall(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\forall x, y$ in the domain of $\mathsf{Decaps}(\mathsf{sk}, .)$,

$$\mathsf{Decaps}(\mathsf{sk}, x \odot y) = \mathsf{Decaps}(\mathsf{sk}, x) \odot \mathsf{Decaps}(\mathsf{sk}, y)$$

Leakage-Resilient Smoothness A hash proof system is $\ell(\lambda)$-smooth if for large enough λ, any (leakage) function f such that the size of the output of f is $\ell(\lambda)$

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

where the distribution is over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $x \leftarrow \overline{\mathcal{L}}$ and $t \leftarrow \mathcal{L}'$.

We define LR-KDM PKE schemes according to the notion given below

2.3.2 Leakage-Resilient Key-Dependent Message PKE Security.

Consider the following experiment with an adversary $\mathcal{A}$ where $\mathcal{F}_{\lambda,S}$ is a class of functions $f : \mathcal{SK}_{\lambda,S} \to \mathcal{M}_{\lambda,S}$ and $\mathcal{H}_{\lambda,S}$ is a class of functions $h : \mathcal{SK}_{\lambda,S} \to \mathcal{L}_S$, and $\mathcal{L}_S$ is the class of Leakage strings of length at most S.

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$
- **Leakage Phase:** $\mathcal{A}_1$ receives (pk) and outputs a function $h \in \mathcal{H}_{\lambda,S}$. The challenger outputs leakage $h(\mathsf{sk})$ such that $|h(\mathsf{sk})| < S$.

- **Query Phase:** $\mathcal{A}$ chooses message query $m_0, m_1 \in \mathcal{F}_{\lambda,S}$. The challenger randomly chooses $d \in \{0,1\}$ and computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{pk}, m_b)$ and sends ct^* to the adversary $\mathcal{A}$
- **Challenge Phase:** $\mathcal{A}$ guesses bit d'. $\mathcal{A}$ wins the experiment if $d = d'$.

We also require that the language $\mathcal{L}$ is subgroup indisinguishable

Subgroup Indistinguishability We say that $\mathcal{L}$ is subgroup indistinguishable if the distributions $\mathcal{D}_1 = \{x : x \leftarrow \mathcal{L}\}$ and $\mathcal{D}_2 = \{x : x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}\}$ are computationally indistinguishable.

We can construct LR-smooth homomorphic Hash Proof Systems from DDH. We refer the reader to section 6.2. Using the construction in section 6.2, we can construct LR-KDM PKE schemes (refer to section 6.3) over the class of affine functions.

Chapter 3

Preliminaries

3.1 Preliminaries and Notations

Throughout this paper, we will use λ to denote the security parameter and $\text{negl}(\cdot)$ 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 two binary string x and y, $x||y$ denotes the concatenation of x and y. For ℓ-length binary string x, x_i denotes the i^{th} bit of x.

3.1.1 Function Classes

In our thesis, we deal with the following functions. These are listed below:
Projection functions: A function $f : \{0,1\}^m \to \{0,1\}^n$ is said to be a projection function if each of its output bits depends on at most a single bit of its input, i.e., for all $j \in [n]$, $f(x)_j \in \{0, 1, x_i, 1 \oplus x_i\}$ for some $i \in [m]$.

Affine linear functions: We define the following class

$$\mathcal{F}_{\text{aff}} = \left\{ f_{\mathbf{a},\mathbf{b}} : \{0,1\}^m \to \{0,1\} | \forall \mathbf{s} \in \{0,1\}^m \; \exists \; \phi : f_{\mathbf{a},b}(\mathbf{s}) = \phi(\mathbf{a}^\top \mathbf{s} + b) \right\}$$

.

Constant functions: Although trivial, we mention the class of constant functions as well. f is a constant function if $\forall x_1, x_2$ in the domain of f, $f(x_1) = f(x_2)$

3.1.2 Average Min-Entropy

For a random variable X, the average min-entropy is defined as

$$H_\infty(X) = -\log \max_x \Pr[X = x]$$

. For two jointly distributed random variables (X, Y), the average min-entropy, the average min-entropy of X conditioned on Y is defined as

$$H_\infty(X|Y) = -\log \mathbb{E}_{y \leftarrow Y}\big[\max_x \Pr[X = x|Y = y]\big]$$

Claim 1 ([DRS04])**.** *For any random variable X, Y where Y is supported over a set of size T, $H_\infty(X|Y) \geq H_\infty(X, Y) - \log(T) \geq H_\infty(X) - \log(T)$.*

3.1.3 Strong Average Min-Entropy Extractor

A (k, ϵ)-strong average min-entropy extractor is an efficient function $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ such that for all jointly distributed random variable X, Y where X takes values $\{0,1\}^d$ and $H_\infty(X|Y) \geq k$, we have $(U_d, \mathsf{Ext}(X, U_d), Y) \approx_\epsilon (U_d, U_m, Y)$ where U_d, U_m are uniformly random strings of length d, m respectively. [Nis90]

3.1.4 Leftover Hash Lemma

Universal Hashing: A family $\mathcal{H}$ of deterministic functions $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ is a universal hash family if $\forall x_1, x_2 \in \mathcal{X} : x_1 \neq x_2$, we have $\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = h(x_2)] = \frac{1}{|\mathcal{Y}|}$
Leftover Hash Lemma: "If $\mathcal{H}$ is a universal hash family with $\mathcal{H} = \{H : \mathcal{X} \to \{0,1\}^m\}$ and $|\mathcal{H}| = 2^d$ and if $H_\infty(X|Y) \geq k$ and $m = k - 2\log(\frac{1}{\epsilon})$, then $(H(X), H, Y) \approx_{\frac{\epsilon}{2}} (U_m, U_d, Y)$"[SGV11]. Hence, the universal hash family $\mathcal{H}$ gives a $(k, \frac{\epsilon}{2})$ extractor

3.1.5 Non-Interactive Zero Knowledge proof systems (NIZK)

A non-interactive zero knowledge proof system(NIZK) for an NP language $\mathcal{L}$ is defined by the following 5 algorithms:

- $K(1^\lambda)$: Outputs a common reference string (crs)
- $S_1(1^\lambda)$: Outputs a simulated pair common reference string and trapdoor (crs, τ)
- $P(\mathsf{crs}, x, w)$: Given the common reference string crs, a string x with a witness w for language $\mathcal{L}$, P returns a proof π
- $S_2(\mathsf{crs}, \tau)$: Outputs a simulated proof π
- $V(\mathsf{crs}, x, \pi)$: Outputs 1 if the proof is accepted, else outputs 0

Correctness: $\forall x \in \mathcal{L}$ with witness w:

$$\Pr[V(\mathsf{crs}, x, \pi) = 1 | \pi - P(\mathsf{crs}, x, w)] \geq 1 - \mathrm{negl}(\lambda)$$

Soundness:

$$\Pr[\exists \pi : V(\mathsf{crs}, x, \pi) = 1 \wedge x \notin \mathcal{L}] \leq \mathrm{negl}(\lambda)$$

Zero Knowledgeness: A NIZK for language $\mathcal{L}$ is zero-knowledge if $\exists$ a ppt simulator $S = (S_1, S_2)$ such that for all $x \in \mathcal{L}$ with a witness w:

$$\{(\mathsf{crs}, \pi) : \mathsf{crs} \leftarrow K(1^\lambda), \pi \leftarrow P(\sigma, x, w)\} \approx$$

$$\{(\mathsf{crs}, \pi) : (\mathsf{crs}, \tau) \leftarrow S_1(1^\lambda), \pi \leftarrow S_2(\mathsf{crs}, \tau)\}$$

3.1.6 Garbling schemes

Let $\mathcal{C}_n$ be some class of circuits where the input length of each circuit is n. A garbling scheme $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ has the following algorithms[Yao86, KM19b]:

- $\mathsf{Garble}(1^\lambda, \mathcal{C} \in \mathcal{C}_n)$: Outputs the garbled circuit $\tilde{C}$ and $(\mathsf{lab}_{i,\gamma})_{i\in[n],\gamma\in\{0,1\}}$
- $\mathsf{Eval}(\tilde{C}, ((\mathsf{lab})_i)_{i\in[n]})$: Outputs a result string y
- $\mathsf{Sim}(1^\lambda, |\mathcal{C}|, C(x))$: Outputs the garbled circuit $\tilde{C}$ and $(\mathsf{lab}_i)_{i\in[n]}$

Correctness: For correctness, we require both of the following to be true:

1. $\mathsf{Eval}(\tilde{\mathcal{C}}, (\mathsf{lab}_{i,x_i})_{i\in[n]}) = x$ where $(\tilde{\mathcal{C}}, (\mathsf{lab}_{i,\gamma})_{i\in[n],\gamma\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)$
2. $\mathsf{Eval}(\tilde{\mathcal{C}}, (\mathsf{lab}_i)_{i\in[n]}) = x$ where $(\tilde{\mathcal{C}}, (\mathsf{lab}_i)_{i\in[n]}) \leftarrow \mathsf{Sim}(1^\lambda, |C|, C(x))$

Security: For security, the following distributions must be indistinguishable:

- $\mathcal{D}_1 = \{\tilde{\mathcal{C}}, (\mathsf{lab}_{i,x_i})_{i\in[n]} | (\tilde{\mathcal{C}}, (\mathsf{lab}_{i,\gamma})_{i\in[n],\gamma\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)\}$
- $\mathcal{D}_2 = \{\tilde{\mathcal{C}}, (\mathsf{lab}_i)_{i\in[n]} | (\tilde{\mathcal{C}}, (\mathsf{lab}_i)_{i\in[n]}) \leftarrow \mathsf{Sim}(1^\lambda, |C|, C(x))\}$

3.1.7 Symmetric and Public Key Encryption

A symmetric key encryption scheme $\mathsf{SKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\{\mathcal{M}_{\lambda,S}\}_{\lambda,S}$, key space $\{\mathcal{K}_{\lambda,S}\}_{\lambda,S}$ and ciphertext space $\{\mathcal{C}_{\lambda,S}\}_{\lambda,S}$ consists of the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda, 1^S)$: The setup algorithm is a randomized algorithm that takes as input the security parameter λ and a parameter 1^S and outputs a secret key sk.
- $\mathsf{Enc}(\mathsf{sk}, m)$: The encryption algorithm is a randomized algorithm takes as input a secret key sk and a message $m \in \mathcal{M}_{\lambda,S}$ 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 $m \in \mathcal{M}_{\lambda,S}$ or $\perp$.

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

$$\Pr_r[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m \mid \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m; r)] = 1$$

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

CPA-SKE Security. Consider the following experiment with an adversary $\mathcal{A}$ where Setup algorithm takes only 1^λ as input.

- **Initialization Phase:** The challenger runs $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$.
- **Pre-Challenge Query Phase:** $\mathcal{A}$ is allowed to make polynomially many queries. For each query m, the challenger computes $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}, m)$ and returns ct to Adv.
- **Challenge Phase:** $\mathcal{A}$ outputs two message m_0, m_1. The challenger randomly chooses $b \in \{0, 1\}$ and computes a ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b)$ and sends it to $\mathcal{A}$.
- **Post-Challenge Query Phase:** $\mathcal{A}$ is allowed to make polynomially many queries. For each query m, the challenger computes $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}, m)$ and returns ct to Adv.
- **Response Phase:** $\mathcal{A}$ outputs $b' \in \{0, 1\}$ and wins the experiment if $b = b'$.

An SKE scheme is said to be secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

Similarly, a public key encryption scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\{\mathcal{M}_{\lambda,S}\}_{\lambda,S}$, key space $\{\mathcal{K}_{\lambda,S}\}_{\lambda,S}$ and ciphertext space $\{\mathcal{C}_{\lambda,S}\}_{\lambda,S}$ consists of the following PPT algorithms.

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

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

$$\Pr_r[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m \mid \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)] = 1$$

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

CPA-PKE Security. Consider the following experiment with an adversary $\mathcal{A}$ where Setup algorithm takes only 1^λ as input.

- **Initialization Phase:** The challenger runs $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$.
- **Pre-Challenge Query Phase:** $\mathcal{A}$ is allowed to make polynomially many queries. For each query m, the challenger computes $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}, m)$ and returns ct to Adv.
- **Challenge Phase:** $\mathcal{A}$ outputs two message m_0, m_1. The challenger randomly chooses $b \in \{0, 1\}$ and computes a ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$ and sends it to $\mathcal{A}$.
- **Post-Challenge Query Phase:** $\mathcal{A}$ is allowed to make polynomially many queries. For each query m, the challenger computes $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}, m)$ and returns ct to Adv.
- **Response Phase:** $\mathcal{A}$ outputs $b' \in \{0, 1\}$ and wins the experiment if $b = b'$.

A PKE scheme is said to be CPA secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

3.2 Variants of Key-Dependent Message (KDM) Security

First we introduce the notion of Key-Dependent Message SKE and PKE secure schemes, after which we add the notion of incompressible encryption to KDM schemes. We also mention a notion of KDM secure encryption in the presence of adversarial leakage of the key's bits and the notion of leakage resilience in the KDM PKE setting.

3.2.1 KDM SKE encryption

Consider the following experiment with an adversary $\mathcal{A} = (\mathcal{A})$ where $\mathcal{F}_{\lambda,S}$ is a class of functions $f : \mathcal{K}_{\lambda,S} \to \mathcal{M}_{\lambda,S}$.

- **Initialization Phase:** The challenger runs $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$.

- **Challenge Phase:** $\mathcal{A}$ outputs two functions $f_0, f_1 \in \mathcal{F}_{\lambda,S}$. The challenger randomly chooses $d \in \{0,1\}$ and computes a ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{sk}, f_b(\mathsf{sk}))$ and sends ct^* to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs its guess d' and wins the experiment iff $d = d'$

An SKE scheme is said to be key-dependent message secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, S \in \mathbb{N}$,

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

3.2.2 KDM PKE encryption

Consider the following experiment with an adversary $\mathcal{A} = (\mathcal{A})$ where $\mathcal{F}_{\lambda,S}$ is a class of functions $f : \mathcal{K}_{\lambda,S} \to \mathcal{M}_{\lambda,S}$.

- **Initialization Phase:** The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$.
- **Challenge Phase:** $\mathcal{A}$ outputs two functions $f_0, f_1 \in \mathcal{F}_{\lambda,S}$. The challenger randomly chooses $d \in \{0,1\}$ and computes a ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, f_b(\mathsf{sk}))$ and sends ct^* to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs its guess d' and wins the experiment iff $d = d'$

A PKE scheme is said to be key-dependent message secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, S \in \mathbb{N}$,

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

3.2.3 KDM Incompressible SKE encryption.

Consider the following experiment with an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{F}_{\lambda,S}$ is a class of functions $f : \mathcal{K}_{\lambda,S} \to \mathcal{M}_{\lambda,S}$.

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$.
- **Challenge Phase:** $\mathcal{A}_1$ outputs a function $f \in \mathcal{F}_{\lambda,S}$, along with an auxiliary information aux. The challenger randomly chooses $d \in \{0,1\}$ and computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{sk}, \mathbf{0})$ if $d = 0$, else it computes $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{sk}, f(\mathsf{sk}))$. It sends ct^* to $\mathcal{A}_1$.
- **First Response Phase:** $\mathcal{A}_1$ computes a state st such that $|\mathsf{st}| \leq S$.
- **Second Response Phase:** $\mathcal{A}_2$ receives $(\mathsf{sk}, \mathsf{aux}, \mathsf{st})$ and outputs d'. $\mathcal{A}$ wins the experiment if $d = d'$.

An SKE scheme is said to be key-dependent message incompressible secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, S \in \mathbb{N}$,

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \text{negl}(\lambda)$$

3.2.4 KDM Incompressible PKE Security.

Consider the following experiment with an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{F}_{\lambda,S}$ is a class of functions $f : \mathcal{SK}_{\lambda,S} \to \mathcal{M}_{\lambda,S}$.

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$
- **Challenge Phase:** $\mathcal{A}_1$ outputs a function $f \in \mathcal{F}_{\lambda,S}$, along with an auxiliary information aux. The challenger randomly chooses $d \in \{0, 1\}$ and computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{pk}, \mathbf{0})$ if $d = 0$, else it computes $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{pk}, f(\mathsf{sk}))$. It sends ct^* to $\mathcal{A}_1$.
- **First Response Phase:** $\mathcal{A}_1$ computes a state st such that $|\mathsf{st}| \leq S$.
- **Second Response Phase:** $\mathcal{A}_2$ receives $(\mathsf{pk}, \mathsf{sk}, \mathsf{aux}, \mathsf{st})$ and outputs d'. $\mathcal{A}$ wins the experiment if $d = d'$.

A PKE scheme is said to be key-dependent message incompressible secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, S \in \mathbb{N}$,

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \text{negl}(\lambda)$$

3.2.5 Leakage-Resilient Key-Dependent Message PKE Security.

Consider the following experiment with an adversary $\mathcal{A}$ where $\mathcal{F}_{\lambda,S}$ is a class of functions $f : \mathcal{SK}_{\lambda,S} \to \mathcal{M}_{\lambda,S}$ and $\mathcal{H}_{\lambda,S}$ is a class of functions $h : \mathcal{SK}_{\lambda,S} \to \mathcal{L}_S$, and $\mathcal{L}_S$ is the class of Leakage strings of length at most S.

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$
- **Leakage Phase:** $\mathcal{A}_1$ receives (pk) and outputs a function $h \in \mathcal{H}_{\lambda,S}$. The challenger outputs leakage $h(\mathsf{sk})$ such that $|h(\mathsf{sk})| < S$.
- **Query Phase:** $\mathcal{A}$ chooses message query $m_0, m_1 \in \mathcal{F}_{\lambda,S}$. The challenger randomly chooses $d \in \{0, 1\}$ and computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{pk}, m_b)$ and sends ct^* to the adversary $\mathcal{A}$
- **Challenge Phase:** $\mathcal{A}$ guesses bit d'. $\mathcal{A}$ wins the experiment if $d = d'$.

A PKE scheme is said to be leakage-resilient key-dependent message secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, S \in \mathbb{N}$,

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \text{negl}(\lambda)$$

Chapter 4

KDM Incompressible encryption in the random oracle model

KDM Incompressible SKE from RO: Let λ, S be security parameters, with l_k being polynomial in λ. Given the functions $G : \{0,1\}^{l_k} \times \{0,1\}^{\mathsf{poly}(\lambda)} \longrightarrow \{0,1\}^n, H : \{0,1\}^{l_k} \times \{0,1\}^n \longrightarrow \{0,1\}^{\mathsf{poly}(\lambda)}$ modelled as random oracles with the adversary having access to both oracles, We can construct a KDM Incompressible SKE scheme for all class of functions f_0, f_1 of unbounded size in the random oracle model $\mathsf{KdmIncSKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for the message space $\{0,1\}^n$ with ciphertext size $\mathsf{poly}(\lambda)$ and state size $n = |S| + \mathsf{poly}(\lambda)$ as given below:

- $\mathsf{KeyGen}(1^\lambda, 1^S)$: Sample a uniform key $k \in \{0,1\}^{\mathsf{poly}(\lambda)}$. Return k.
- $\mathsf{Enc}(k, m)$: Choose a random $r \in \{0,1\}^{\mathsf{poly}(\lambda)}$. Let $d = G(k,r) \oplus (m)$. Let $c = H(k,d) \oplus r$. Return $\mathsf{ct} = (c,d)$
- $\mathsf{Dec}(k, \mathsf{ct} = (c,d))$: Compute $r = H(k,d) \oplus c$. Compute $m = G(k,r) \oplus d$. Return m.

Theorem 4.0.1. *The above SKE scheme is Incompressible KDM secure in the Random Oracle Model*

Proof. We prove security using a sequence of hybrids.

G_0: This is the incompressible KDM SKE game

- Sample a uniformly random $k \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Receive function queries f_0, f_1 from $\mathcal{A}_1$ and auxiliary information aux
- Sample a bit $b \leftarrow \{0,1\}$
- Sample a uniformly random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Compute $d = G(k,r) \oplus f_b(k)$
- Compute $c = H(k,d) \oplus r$
- Compute $\mathsf{ct} = (c,d)$
- Send ct to $\mathcal{A}_1$ and receive state st of size at most S

- Send $\mathsf{st}, f_0, f_1, k, \mathsf{aux}$ to $\mathcal{A}_2$, which outputs bit b'

G_1:

- Sample a uniformly random $k \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Receive function queries f_0, f_1 from $\mathcal{A}_1$ and auxiliary information aux
- Sample a bit $b \leftarrow \{0,1\}$
- Sample a uniformly random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Compute $d = G(k,r) \oplus f_b(k)$
- Compute $c = H(k,d) \oplus r$
- Compute $\mathsf{ct} = (c,d)$
- Send ct to $\mathcal{A}_1$ and receive state st of size at most S
- If $\mathcal{A}_1$ queries G or H on (k,x) for some x, abort
- Send $\mathsf{st}, f_0, f_1, k, \mathsf{aux}$ to $\mathcal{A}_2$, which outputs bit b'

G_2: Maintain a table simulating the random oracle. For a new query, sample a random response add it to the table, else return the existing entry in the table.

- Sample a uniformly random $k \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Receive function queries f_0, f_1 from $\mathcal{A}_1$ and auxiliary information aux
- Sample a bit $b \leftarrow \{0,1\}$
- Maintain a table for random oracle queries for G and H.
- Compute $d \leftarrow \{0,1\}^n$
- Compute $c \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Sample a uniformly random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- In the table, set $G(k,r)$ as $d \oplus f_b(k)$
- In the table, set $H(k,d)$ as $c \oplus r$
- Compute $\mathsf{ct} = (c,d)$
- Send ct to $\mathcal{A}_1$ and receive state st of size at most S
- If $\mathcal{A}_1$ queries G or H on (k,x) for some x, abort

- Send $\mathsf{st}, f_0, f_1, k, \mathsf{aux}$ to $\mathcal{A}_2$, which outputs bit b'

G_3:

- Sample a uniformly random $k \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Receive function queries f_0, f_1 from $\mathcal{A}_1$ and auxiliary information aux
- Sample a bit $b \leftarrow \{0,1\}$
- Maintain a table for random oracle queries for G and H.
- Compute $d \leftarrow \{0,1\}^n$
- Compute $c \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Sample a uniformly random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- In the table, set $G(k,r)$ as $d \oplus f_b(k)$
- In the table, set $H(k,d)$ as $c \oplus r$
- Compute $\mathsf{ct} = (c,d)$
- Send ct to $\mathcal{A}_1$ and receive state st of size at most S
- If $\mathcal{A}_1$ queries G or H on (k,x) for some x, abort
- Send $\mathsf{st}, f_0, f_1, k, \mathsf{aux}$ to $\mathcal{A}_2$, which outputs bit b'
- If $\mathcal{A}_2$ queries $G(k,r)$ before querying $H(k,d)$, abort

G_4:

- Sample a uniformly random $k \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Receive function queries f_0, f_1 from $\mathcal{A}_1$ and auxiliary information aux
- Sample a bit $b \leftarrow \{0,1\}$
- Maintain a table for random oracle queries for G and H.
- Compute $d \leftarrow \{0,1\}^n$
- Compute $c \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Sample a uniformly random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- In the table, set $G(k,r)$ as $d \oplus f_b(k)$
- In the table, set $H(k,d)$ as $c \oplus r$

- Compute $\mathsf{ct} = (c, d)$
- Send ct to $\mathcal{A}_1$ and receive state st of size at most S
- If $\mathcal{A}_1$ queries G or H on (k, x) for some x, abort
- Send $\mathsf{st}, f_0, f_1, k, \mathsf{aux}$ to $\mathcal{A}_2$, which outputs bit b'
- If $\mathcal{A}_2$ queries $G(k, r)$ before querying $H(k, d)$, abort
- If $\mathcal{A}_2$ queries $H(k, d)$, abort

Claim 2. *G_0 and G_1 are indistinguishable*

Proof. If an adversary can distinguish between G_0, G_1 the adversary must with non-negligible probability query G or H on (k, x) for some k. However, this is impossible as for any k, $G(k, r)$ and $H(k, d)$ are randomly sampled, hence the distribution (c, d) is identical to a random distribution □

Claim 3. *G_1 and G_2 are indistinguishable*

Proof. Note that since $\mathcal{A}_1$ has not queried (k, x) for any x on F, G oracles, all values of $H(k, x)$ and $G(k, x)$ are uniformly randomly chosen whenever queried except for $G(k, r)$ and $H(k, d)$ in G_1. Now since at the time of initial sampling $G(k, r)$ is also random, we can switch to the following game: compute d randomly and c randomly as well. Compute r randomly and in our table, set $G(k, r)$ as $f(k) \oplus d$ and set $H(k, d)$ as $c \oplus r$. This new game is logically equivalent to G_2. Now, it is impossible that an adversary can query $G(k, r)$ before $H(k, d)$ with non-negligible probability, as c and d have no information about r and r is uniformly random given all other information except $H(k, d)$. Hence, G_1 and G_2 are indistinguishable □

Claim 4. *G_2 and G_3 are indistinguishable*

Proof. Since in G_2, (c, d) are now randomly sampled independent of r, and the only information that the adversary can get about r is through a random oracle query to $H(k, d)$, it is impossible for an adversary to query $G(k, r)$ before $H(k, d)$ with non-negligible probability. Hence, G_2 and G_3 are indistinguishable. □

Claim 5. *G_3 and G_4 are indistinguishable*

Proof. Since the adversary does not query $G(k, r)$ before $H(k, d)$, and d is uniformly randomly sampled, and the adversary's view has only st which is $\mathsf{poly}(\lambda)$ bits smaller than n, the probability of querying $H(k, d)$ is negligible in λ. Hence, G_3 and G_4 are indistinguishable. □

Claim 6. *The adversary can win in G_4 with at most $\frac{1}{2}$ probability*

Proof. Since no queries to $G(k, r)$ and $H(k, d)$ are made by both adversaries, and c and d are randomly sampled, there is no information about $f_b(k)$ in the adversary's view. Hence, the adversary can win with at most $\frac{1}{2}$ probability □

From the above claims, it is easy to show that for any adversary, $\mathsf{KdmIncSKE}$ is a KDM Incompressible SKE secure scheme, as for all adversaries, the probability of winning in the security game of $\mathsf{KdmIncSKE} \leq \frac{1}{2} + \text{negl}(\lambda)$ □

KDM Incompressible PKE from RO: Using the above construction along with a CPA secure PKE scheme $\mathsf{PKE}' = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, and assuming a random hash function $H' : \{0,1\}^{\mathsf{poly}(\lambda)} \longrightarrow \{0,1\}^{l_k}$ modelled as a random oracle, we construct the incompressible KDM PKE scheme $\mathsf{IncKDMPKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ over message space $\{0,1\}^n$ and any class of functions $\mathcal{F}$ for the KDM query as given below:

- $\mathsf{KeyGen}(1^\lambda, 1^S)$: Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE}'.\mathsf{KeyGen}'(1^\lambda)$. Return $(\mathsf{pk}, \mathsf{sk})$
- $\mathsf{Enc}(\mathsf{pk}, m)$: Sample uniformly random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$. Compute $c \leftarrow \mathsf{PKE}'.\mathsf{Enc}(\mathsf{pk}, r)$ and $d \leftarrow \mathsf{KdmIncSKE}.\mathsf{Enc}(H'(r), m)$. Return $\mathsf{ct} = (c, d)$
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (c, d))$: Compute $r = \mathsf{PKE}'.\mathsf{Dec}'(\mathsf{sk}, c)$. Return $m = \mathsf{KdmIncSKE}.\mathsf{Dec}(H'(r), d)$

Theorem 4.0.2. *The above PKE scheme is Incompressible KDM secure in the Random Oracle Model*

Proof. Correctness holds trivially. For proving security, we use a sequence of hybrids. G_0:

This is the incompressible KDM PKE game

- Challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE}'.\mathsf{KeyGen}(1^\lambda)$
- Receive function queries f_0, f_1 from $\mathcal{A}_1$ and auxiliary information aux
- Sample a bit $b \leftarrow \{0,1\}$
- Sample a uniformly random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Compute $c \leftarrow \mathsf{PKE}'.\mathsf{Enc}(\mathsf{pk}, r)$
- Compute $d \leftarrow \mathsf{KdmIncSKE}.\mathsf{Enc}(H'(r), f_b(\mathsf{sk}))$
- Compute $\mathsf{ct} = (c, d)$. Send ct to $\mathcal{A}_1$ and receive state st of size at most S
- Send $\mathsf{st}, f_0, f_1, k, \mathsf{aux}$ to $\mathcal{A}_2$, which outputs bit b'

G_1:

- Challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE}'.\mathsf{KeyGen}(1^\lambda)$
- Receive function queries f_0, f_1 from $\mathcal{A}_1$ and auxiliary information aux
- Sample a bit $b \leftarrow \{0,1\}$
- Sample a uniformly random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Compute $c \leftarrow \mathsf{PKE}'.\mathsf{Enc}(\mathsf{pk}, r)$
- Compute $d \leftarrow \mathsf{KdmIncSKE.Enc}(H'(r), f_b(\mathsf{sk}))$
- Compute $\mathsf{ct} = (c, d)$. Send ct to $\mathcal{A}_1$ and receive state st of size at most S
- If $\mathcal{A}_1$ queries $H'(r)$ abort
- Send $\mathsf{st}, f_0, f_1, k, \mathsf{aux}$ to $\mathcal{A}_2$, which outputs bit b'

G_2: Maintain a table simulating the random oracle. For a new query, sample a random response add it to the table, else return the existing entry in the table.

- Challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE}'.\mathsf{KeyGen}(1^\lambda)$
- Receive function queries f_0, f_1 from $\mathcal{A}_1$ and auxiliary information aux
- Sample a bit $b \leftarrow \{0,1\}$
- Sample a uniformly random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
- Compute $c \leftarrow \mathsf{PKE}'.\mathsf{Enc}(\mathsf{pk}, r)$
- Sample k uniformly randomly, and in the simulation table set $H'(r)$ as k
- Compute $d \leftarrow \mathsf{KdmIncSKE.Enc}(k, f_b(\mathsf{sk}))$
- Compute $\mathsf{ct} = (c, d)$. Send ct to $\mathcal{A}_1$ and receive state st of size at most S
- If $\mathcal{A}_1$ queries $H'(r)$ abort
- Send $\mathsf{st}, f_0, f_1, k, \mathsf{aux}$ to $\mathcal{A}_2$, which outputs bit b'

Claim 7. *G_0 and G_1 are indistinguishable*

Proof. If there exists an adversary $\mathcal{A}_1$ that queries $H'(r)$ with non-negligible probability, we can construct a reduction that can break the CPA security of Pke'. Note that since H' is a random oracle, d is independent of r and has no information that affects the distribution of r in the adversary's view. □

Claim 8. *G_1 and G_2 are indistinguishable*

Proof. The games are equivalent as the behaviour of the random oracle is equivalent to simulating a randomized table. □

Claim 9. *The adversary can win in G_2 with at most $\frac{1}{2} + \text{negl}(\lambda)$ probability*

Proof. This holds as k is independent of r and hence, c does not have any information about bit b. If an adversary can win G_2 with non-negligible advantage, we can construct a reduction that can also win the KDM Incompressible security game for $\mathsf{KdmIncSKE}$. □

From the above claims, it is easy to show that for any adversary, $\mathsf{IncKDMPKE}$ is a KDM Incompressible PKE secure scheme, as for all adversaries, the probability of winning in the security game of $\mathsf{IncKDMPKE} \leq \frac{1}{2} + \text{negl}(\lambda)$ □

Chapter 5

Incompressible KDM CCA

- Let IncPKE be an incompressible CPA secure PKE with randomness space $\mathcal{R}$
- Let PRJ be a one-time secure projection KDM incompressible SKE with key space $\mathcal{K} = \{0,1\}^{l_s}$
- Let IncCCA be an incompressible CCA2 secure PKE.
- Let NIZK be a Non-Interactive Zero Knowledge proving scheme with adaptive soundness for the following NP language $\mathcal{L}$:
  - $\mathcal{L} = \big\{(\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}} | \forall i \in [l_s] : \exists(\mathsf{lab}_i, r_{i,0}, r_{i,1}) :$
    $\mathsf{ct}_{i,0} = \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,0}, \mathsf{lab}_i; r_{i,0}), \mathsf{ct}_{i,1} = \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,1}, \mathsf{lab}_i; r_{i,1})\big\}$
- Let $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ be a secure circuit garbling scheme.

Consider the scheme $\mathsf{IncKDMCCA} = (\mathsf{IncKDMCCA.KeyGen}, \mathsf{IncKDMCCA.Enc}, \mathsf{IncKDMCCA.Dec})$ with message space $\mathcal{M}_{\lambda,S}$, ciphertext space $\mathcal{C}_{\lambda,S}$:

- $\mathsf{IncKDMCCA.KeyGen}(1^\lambda, 1^S)$: Let l_s be the key length for PRJ
  - $\forall(i,\gamma) \in [l_s] \times \{0,1\}$: sample $(\mathsf{pk}_{i,\gamma}, \mathsf{sk}_{i,\gamma}) \leftarrow \mathsf{IncPKE.KeyGen}(1^\lambda, 1^S)$
  - $(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}) \leftarrow \mathsf{IncCCA.KeyGen}(1^\lambda, 1^S)$
  - $(\mathsf{crs}) \leftarrow \mathsf{NIZK.K}(1^\lambda)$
  - $s \leftarrow \mathsf{PRJ.KeyGen}(1^\lambda, 1^S)$ where $s \in \{0,1\}^{l_s}$
  - $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}\big(s, (\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]})\big)$
  - $\mathsf{IncKDMCCA.sk} := s$
  - $\mathsf{IncKDMCCA.pk} := \big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big)$
  - Return $(\mathsf{IncKDMCCA.pk}, \mathsf{IncKDMCCA.sk})$
- $\mathsf{IncKDMCCA.Enc}(\mathsf{IncKDMCCA.pk}, m)$: Here C_m denotes a circuit hardwired to output the constant m for all input configurations
  - $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}$
  - $(\tilde{C}_m, (\mathsf{lab}_i)_{i\in[l_s]}) \leftarrow \mathsf{Sim}(1^\lambda, |C_m|, m)$
  - $\forall(i,\gamma) \in [l_s] \times \{0,1\}$: sample $r_{i,\gamma} \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,\gamma}, \mathsf{lab}_i; r_{i,\gamma})$
  - Let $x = (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$
  - Let $w = (\mathsf{lab}_i, r_{i,0}, r_{i,1})_{i\in[l_s]}$
  - $\Pi \leftarrow \mathsf{NIZK.P}(\mathsf{crs}, x, w)$
  - Finally return

$$\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, ((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_m), \Pi)\big)$$

- $\mathsf{IncKDMCCA.Dec}(\mathsf{IncKDMCCA.pk}, \mathsf{IncKDMCCA.sk}, \mathsf{ct_{CCA}})$
  - $\left(\mathsf{pk_{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct_{sk}}\right) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
  - $\left(\mathsf{sk_{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\right) = \mathsf{PRJ.Dec}(s, \mathsf{ct_{sk}})$
  - $\left((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_m), \Pi\right) = \mathsf{IncCCA.Dec}(\mathsf{pk_{CCA}}, \mathsf{sk_{CCA}}, \mathsf{ct_{CCA}})$
  - Let $x = (pk_{i,\gamma}, ct_{i,\gamma})$. If $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) \neq 1$: return $\perp$
  - $\forall i \in [l_s]$:
    $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,s_i}, \mathsf{sk}_{i,s_i}, ct_{i,s_i})$
  - Return $\mathsf{Eval}(\tilde{C}_m, (\mathsf{lab}_i)_{i\in[l_s]})$

Theorem 5.0.1. *For the above scheme* IncKDMCCA*, correctness must hold valid*

$$\forall m \in \mathcal{M}_{\lambda,S} : \Pr[\mathsf{IncKDMCCA.Dec}(\mathsf{IncKDMCCA.pk}, \mathsf{IncKDMCCA.sk}, x) = m : x \leftarrow \mathsf{IncKDMCCA.Enc}(\mathsf{IncKDMCCA.pk}, m)] \geq 1 - \mathrm{negl}(\lambda)$$

Proof. Let $m \in \mathcal{M}_{\lambda,S}$. Let $\left(\mathsf{pk_{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct_{sk}}\right) = \mathsf{IncKDMCCA.pk}$ and $s = \mathsf{IncKDMCCA.sk}$. Let $\mathsf{ct_{CCA}} \leftarrow \mathsf{IncKDMCCA.Enc}(\mathsf{IncKDMCCA.pk}, m)$. Then $\mathsf{IncKDMCCA.Dec}(\mathsf{IncKDMCC}$ is as follows:

1. $\left(\mathsf{sk_{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\right) = \mathsf{PRJ.Dec}(s, \mathsf{ct_{sk}})$ (... correctness of PRJ)
2. $\left((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_m), \Pi\right) = \mathsf{IncCCA.Dec}(\mathsf{sk_{CCA}}, \mathsf{ct_{CCA}})$ (... correctness of IncCCA)
3. check for $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi)$ must output 1 for $m \in \mathcal{M}_{\lambda,S}$ (... correctness of NIZK)
4. $\forall i \in [l_s]$:
   $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,s_i}, \mathsf{sk}_{i,s_i}, ct_{i,s_i})$ (... correctness of IncPKE)
5. $m = \mathsf{Eval}(\tilde{C}_m, (\mathsf{lab}_i)_{i\in[l_s]})$ (... correctness of (Garble, Eval, Sim))

Hence, correctness for IncKDMCCA must hold valid

□

Theorem 5.0.2. IncKDMCCA *is a* CCA2 *KDM Incompressible secure PKE*

Proof. We prove the CCA2 KDM Incompressible security through a sequence of hybrids.

G_0: This is the hybrid for the original incompressible KDM CCA2 game

1. **Initialization Phase**

   (a) The adversary chooses a bit $\beta \leftarrow \{0,1\}$

   (b) The first adversary $\mathcal{A}_1$ sends (λ, S) to the challenger.

   (c) The challenger computes the following:

   - $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $(\mathsf{pk}_{i,\gamma}, \mathsf{sk}_{i,\gamma}) \leftarrow \mathsf{IncPKE.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}) \leftarrow \mathsf{IncCCA.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{crs}) \leftarrow \mathsf{NIZK.K}(1^\lambda)$
   - $s \leftarrow \mathsf{PRJ.KeyGen}(1^\lambda, 1^S)$ where $s \in \{0,1\}^{l_s}$
   - $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}\big(s, (\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]})\big)$
   - $\mathsf{IncKDMCCA.sk} := s$
   - $\mathsf{IncKDMCCA.pk} := \big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big)$

   (d) Challenger maintains a database τ storing ciphertexts returned for encryption queries. Challenger sends $\mathsf{IncKDMCCA.pk}$ to $\mathcal{A}_1$

   (e) $\mathcal{A}_1$ sends aux to the challenger

2. **Challenge Phase**
   Encryption queries:

   - Challenger receives the query (f_0, f_1) and computes the encryption of $f_\beta(s)$
   - $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
   - $(\tilde{C}_{f_\beta(s)}, (\mathsf{lab}_i)_{i\in[l_s]}) \leftarrow \mathsf{Sim}(1^\lambda, |C_{f_\beta(s)}|, f_\beta(s))$
   - $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $r_{i,\gamma} \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,\gamma}, \mathsf{lab}_i; r_{i,\gamma})$
   - Let $x = (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$
   - Let $w = (\mathsf{lab}_i, r_{i,0}, r_{i,1})_{i\in[l_s]}$
   - $\Pi \leftarrow \mathsf{NIZK.P}(\mathsf{crs}, x, w)$
   - $\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, ((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_{f_\beta(s)}), \Pi)\big)$
   - Challenger adds $\mathsf{ct}_{\mathsf{CCA}}$ to database τ sends $\mathsf{ct}_{\mathsf{CCA}}$ as response to the query

   Decryption queries:

   - Challenger receives the query ct. If ct is in database τ, return $\perp$
   - $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
   - Compute $\big(\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\big) = \mathsf{PRJ.Dec}(s, \mathsf{ct}_{\mathsf{sk}})$
   - Compute $\big((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}), \Pi\big) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct})$
   - Let $x = (pk_{i,\gamma}, ct_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$. If $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) \neq 1$: return $\perp$
   - $\forall i \in [l_s]$:
     Compute $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,s_i}, \mathsf{sk}_{i,s_i}, ct_{i,s_i})$

- Return $m = \mathsf{Eval}(\tilde{C}, (\mathsf{lab}_i)_{i\in[l_s]})$ as response to decryption query

3. **First Response Phase** $\mathcal{A}_1$ sends st to the challenger where $|\mathsf{st}| < S$
4. **Second Response Phase**

   (a) The challenger sends $(\mathsf{aux}, \mathsf{sk}, \mathsf{st})$ to the second adversary $\mathcal{A}_2$.

   (b) $\mathcal{A}_2$ outputs $\beta' \in \{0,1\}$. $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins if $\beta' = \beta$

G_1 :

1. **Initialization Phase**

   (a) The adversary chooses a bit $\beta \leftarrow \{0,1\}$

   (b) The first adversary $\mathcal{A}_1$ sends (λ, S) to the challenger.

   (c) The challenger computes the following:

   - $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $(\mathsf{pk}_{i,\gamma}, \mathsf{sk}_{i,\gamma}) \leftarrow \mathsf{IncPKE.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}) \leftarrow \mathsf{IncCCA.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK}.S_1(1^\lambda)$
   - $s \leftarrow \mathsf{PRJ.KeyGen}(1^\lambda, 1^S)$ where $s \in \{0,1\}^{l_s}$
   - $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}\big(s, (\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]})\big)$
   - $\mathsf{IncKDMCCA.sk} := s$
   - $\mathsf{IncKDMCCA.pk} := \big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big)$

   (d) Challenger maintains a database τ storing ciphertexts returned for encryption queries. Challenger sends $\mathsf{IncKDMCCA.pk}$ to $\mathcal{A}_1$

   (e) $\mathcal{A}_1$ sends aux to the challenger

2. **Challenge Phase**
   Encryption queries:

   - Challenger receives the query (f_0, f_1)
   - $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
   - $(\tilde{C}_{f_\beta(s)}, (\mathsf{lab}_i)_{i\in[l_s]}) \leftarrow \mathsf{Sim}(1^\lambda, |C_{f_\beta(s)}|, f_\beta(s))$
   - $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $r_{i,\gamma} \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,\gamma}, \mathsf{lab}_i; r_{i,\gamma})$
   - Let $x = (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$
   - ~~Let $w = (\mathsf{lab}_i, r_{i,0}, r_{i,1})_{i\in[l_s]}$~~
   - $\Pi \leftarrow \mathsf{NIZK}.S_2(\mathsf{td}, x)$
   - $\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, ((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_{f_\beta(s)}), \Pi)\big)$
   - Challenger adds $\mathsf{ct}_{\mathsf{CCA}}$ to database τ sends $\mathsf{ct}_{\mathsf{CCA}}$ as response to the query

   Decryption queries:

- Challenger receive the query ct. If ct is in database τ, return $\perp$
- $\left(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\right) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
- Compute $\left(\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\right) = \mathsf{PRJ.Dec}(s, \mathsf{ct}_{\mathsf{sk}})$
- Compute $\left((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}), \Pi\right) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct})$
- Let $x = (pk_{i,\gamma}, ct_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$. If $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) \neq 1$: return $\perp$
- $\forall i \in [l_s]$:
  Compute $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,s_i}, \mathsf{sk}_{i,s_i}, ct_{i,s_i})$
- Return $m = \mathsf{Eval}(\tilde{C}, (\mathsf{lab}_i)_{i\in[l_s]})$ as response to decryption query

3. **First Response Phase** $\mathcal{A}_1$ sends st to the challenger where $|\mathsf{st}| < S$
4. **Second Response Phase**

   (a) The challenger sends $(\mathsf{aux}, \mathsf{sk}, \mathsf{st})$ to the second adversary $\mathcal{A}_2$.
   (b) $\mathcal{A}_2$ outputs $\beta' \in \{0,1\}$. $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins if $\beta' = \beta$

G_2:

1. **Initialization Phase**

   (a) The adversary chooses a bit $\beta \leftarrow \{0,1\}$
   (b) The first adversary $\mathcal{A}_1$ sends (λ, S) to the challenger.
   (c) The challenger computes the following:

   - $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $(\mathsf{pk}_{i,\gamma}, \mathsf{sk}_{i,\gamma}) \leftarrow \mathsf{IncPKE.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}) \leftarrow \mathsf{IncCCA.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK}.S_1(1^\lambda)$
   - $s \leftarrow \mathsf{PRJ.KeyGen}(1^\lambda, 1^S)$ where $s \in \{0,1\}^{l_s}$
   - $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}\left(s, (\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]})\right)$
   - $\mathsf{IncKDMCCA.sk} := s$
   - $\mathsf{IncKDMCCA.pk} := \left(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\right)$

   (d) Challenger maintains a database τ storing ciphertexts returned for encryption queries. Challenger sends $\mathsf{IncKDMCCA.pk}$ to $\mathcal{A}_1$
   (e) $\mathcal{A}_1$ sends aux to the challenger

2. **Challenge Phase**
   Encryption queries:

   - Challenger receive the query (f_0, f_1)
   - $\left(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\right) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
   - $(\tilde{C}_{f_\beta}, (\mathsf{lab}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, f_\beta)$
   - $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $r_{i,\gamma} \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,\gamma}, \mathsf{lab}_{i,s_i}; r_{i,\gamma})$

- Let $x = (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$
- $\Pi \leftarrow \mathsf{NIZK}.S_2(\mathsf{td}, x)$
- $\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, ((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_{f_\beta}), \Pi)\big)$
- Challenger adds $\mathsf{ct}_{\mathsf{CCA}}$ to database τ sends $\mathsf{ct}_{\mathsf{CCA}}$ as response to the query

Decryption queries:

- Challenger receives the query ct. If ct is in database τ, return $\perp$
- $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
- Compute $\big(\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\big) = \mathsf{PRJ.Dec}(s, \mathsf{ct}_{\mathsf{sk}})$
- Compute $\big((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}), \Pi\big) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct})$
- Let $x = (pk_{i,\gamma}, ct_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$. If $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) \neq 1$: return $\perp$
- $\forall i \in [l_s]$:
  Compute $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,s_i}, \mathsf{sk}_{i,s_i}, ct_{i,s_i})$
- Return $m = \mathsf{Eval}(\tilde{C}, (\mathsf{lab}_i)_{i\in[l_s]})$ as response to decryption query

3. **First Response Phase** $\mathcal{A}_1$ sends st to the challenger where $|\mathsf{st}| < S$
4. **Second Response Phase**
   (a) The challenger sends $(\mathsf{aux}, \mathsf{sk}, \mathsf{st})$ to the second adversary $\mathcal{A}_2$.
   (b) $\mathcal{A}_2$ outputs $\beta' \in \{0,1\}$. $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins if $\beta' = \beta$

G_3:

1. **Initialization Phase**
   (a) The adversary chooses a bit $\beta \leftarrow \{0,1\}$
   (b) The first adversary $\mathcal{A}_1$ sends (λ, S) to the challenger.
   (c) The challenger computes the following:
   - $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $(\mathsf{pk}_{i,\gamma}, \mathsf{sk}_{i,\gamma}) \leftarrow \mathsf{IncPKE.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}) \leftarrow \mathsf{IncCCA.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK}.S_1(1^\lambda)$
   - $s \leftarrow \mathsf{PRJ.KeyGen}(1^\lambda, 1^S)$ where $s \in \{0,1\}^{l_s}$
   - $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}\big(s, (\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]})\big)$
   - $\mathsf{IncKDMCCA.sk} := s$
   - $\mathsf{IncKDMCCA.pk} := \big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big)$

   (d) Challenger maintains a database τ storing ciphertexts returned for encryption queries. Challenger sends $\mathsf{IncKDMCCA.pk}$ to $\mathcal{A}_1$
   (e) $\mathcal{A}_1$ sends aux to the challenger

2. **Challenge Phase**
Encryption queries:

- Challenger receives the query (f_0, f_1)
- $\left(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\right) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
- $(\tilde{C}_{f_\beta}, (\mathsf{lab}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, f_\beta)$
- $\forall (i,\gamma) \in [l_s]\times\{0,1\}$: sample $r_{i,\gamma} \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,\gamma}, \mathsf{lab}_{i,\gamma}; r_{i,\gamma})$
- Let $x = (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$
- $\Pi \leftarrow \mathsf{NIZK}.S_2(\mathsf{td}, x)$
- $\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, ((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_{f_\beta}), \Pi)\big)$
- Challenger adds $\mathsf{ct}_{\mathsf{CCA}}$ to database τ sends $\mathsf{ct}_{\mathsf{CCA}}$ as response to the query

Decryption queries:

- Challenger receives the query ct. If ct is in database τ, return $\perp$
- $\left(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\right) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
- Compute $\left(\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\right) = \mathsf{PRJ.Dec}(s, \mathsf{ct}_{\mathsf{sk}})$
- Compute $\left((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}), \Pi\right) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct})$
- Let $x = (pk_{i,\gamma}, ct_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$. If $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) \neq 1$: return $\perp$
- $\forall i \in [l_s]$:
Compute $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,s_i}, \mathsf{sk}_{i,s_i}, ct_{i,s_i})$
- Return $m = \mathsf{Eval}(\tilde{C}, (\mathsf{lab}_i)_{i\in[l_s]})$ as response to decryption query

3. **First Response Phase** $\mathcal{A}_1$ sends st to the challenger where $|\mathsf{st}| < S$

4. **Second Response Phase**

(a) The challenger sends $(\mathsf{aux}, \mathsf{sk}, \mathsf{st})$ to the second adversary $\mathcal{A}_2$.
(b) $\mathcal{A}_2$ outputs $\beta' \in \{0,1\}$. $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins if $\beta' = \beta$

G_4:

1. **Initialization Phase**

(a) The adversary chooses a bit $\beta \leftarrow \{0,1\}$
(b) The first adversary $\mathcal{A}_1$ sends (λ, S) to the challenger.
(c) The challenger computes the following:

- $\forall (i,\gamma) \in [l_s]\times\{0,1\}$: sample $(\mathsf{pk}_{i,\gamma}, \mathsf{sk}_{i,\gamma}) \leftarrow \mathsf{IncPKE.KeyGen}(1^\lambda, 1^S)$
- $(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}) \leftarrow \mathsf{IncCCA.KeyGen}(1^\lambda, 1^S)$
- $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK}.S_1(1^\lambda)$
- $s \leftarrow \mathsf{PRJ.KeyGen}(1^\lambda, 1^S)$ where $s \in \{0,1\}^{l_s}$

- $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}\big(s, (\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]})\big)$
- $\mathsf{IncKDMCCA.sk} := s$
- $\mathsf{IncKDMCCA.pk} := \big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big)$

(d) Challenger maintains a database τ storing ciphertexts returned for encryption queries. Challenger sends $\mathsf{IncKDMCCA.pk}$ to $\mathcal{A}_1$

(e) $\mathcal{A}_1$ sends aux to the challenger

2. **Challenge Phase**
Encryption queries:

- Challenger receives the query (f_0, f_1)
- $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
- $(\tilde{C}_{f_\beta}, (\mathsf{lab}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, f_\beta)$
- $\forall (i,\gamma) \in [l_s]\times\{0,1\}$: sample $r_{i,\gamma} \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,\gamma}, \mathsf{lab}_{i,\gamma}; r_{i,\gamma})$
- Let $x = (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$
- $\Pi \leftarrow \mathsf{NIZK}.S_2(\mathsf{td}, x)$
- $\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, ((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_{f_\beta}), \Pi)\big)$
- Challenger adds $\mathsf{ct}_{\mathsf{CCA}}$ to database τ sends $\mathsf{ct}_{\mathsf{CCA}}$ as response to the query

Decryption queries:

- Challenger receives the query ct. If ct is in database τ, return $\perp$
- $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
- Compute $\big(\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\big) = \mathsf{PRJ.Dec}(s, \mathsf{ct}_{\mathsf{sk}})$
- Compute $\big((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}), \Pi\big) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct})$
- Let $x = (pk_{i,\gamma}, ct_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$. If $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) \neq 1$: return $\perp$
- $\forall i \in [l_s]$:
  Compute $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,0}, \mathsf{sk}_{i,0}, ct_{i,0})$
- Return $m = \mathsf{Eval}(\tilde{C}, (\mathsf{lab}_i)_{i\in[l_s]})$ as response to decryption query

3. **First Response Phase** $\mathcal{A}_1$ sends st to the challenger where $|\mathsf{st}| < S$

4. **Second Response Phase**

(a) The challenger sends $(\mathsf{aux}, \mathsf{sk}, \mathsf{st})$ to the second adversary $\mathcal{A}_2$.

(b) $\mathcal{A}_2$ outputs $\beta' \in \{0,1\}$. $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins if $\beta' = \beta$

G_5:

1. **Initialization Phase**

(a) The adversary chooses a bit $\beta \leftarrow \{0,1\}$

(b) The first adversary $\mathcal{A}_1$ sends (λ, S) to the challenger.

(c) The challenger computes the following:

- $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $(\mathsf{pk}_{i,\gamma}, \mathsf{sk}_{i,\gamma}) \leftarrow \mathsf{IncPKE.KeyGen}(1^\lambda, 1^S)$
- $(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}) \leftarrow \mathsf{IncCCA.KeyGen}(1^\lambda, 1^S)$
- $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK}.S_1(1^\lambda)$
- $s \leftarrow \mathsf{PRJ.KeyGen}(1^\lambda, 1^S)$ where $s \in \{0,1\}^{l_s}$
- $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}\big(s, \mathbf{0}\big)$
- $\mathsf{IncKDMCCA.sk} := s$
- $\mathsf{IncKDMCCA.pk} := \big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big)$

(d) Challenger maintains a database τ storing ciphertexts returned for encryption queries. Challenger sends $\mathsf{IncKDMCCA.pk}$ to $\mathcal{A}_1$

(e) $\mathcal{A}_1$ sends aux to the challenger

2. **Challenge Phase**
Encryption queries:

- Challenger receives the query (f_0, f_1)
- $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
- $(\tilde{C}_{f_\beta}, (\mathsf{lab}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, f_\beta)$
- $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $r_{i,\gamma} \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,\gamma}, \mathsf{lab}_{i,\gamma}; r_{i,\gamma})$
- Let $x = (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$
- $\Pi \leftarrow \mathsf{NIZK}.S_2(\mathsf{td}, x)$
- $\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, ((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_{f_\beta}), \Pi)\big)$
- Challenger adds $\mathsf{ct}_{\mathsf{CCA}}$ to database τ sends $\mathsf{ct}_{\mathsf{CCA}}$ as response to the query

Decryption queries:

- Challenger receives the query ct. If ct is in database τ, return $\perp$
- $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
- Compute $\big(\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\big) = \mathsf{PRJ.Dec}(s, \mathsf{ct}_{\mathsf{sk}})$
- Compute $\big((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}), \Pi\big) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct})$
- Let $x = (pk_{i,\gamma}, ct_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$. If $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) \neq 1$: return $\perp$
- $\forall i \in [l_s]$:
Compute $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,0}, \mathsf{sk}_{i,0}, ct_{i,0})$
- Return $m = \mathsf{Eval}(\tilde{C}, (\mathsf{lab}_i)_{i\in[l_s]})$ as response to decryption query

3. **First Response Phase** $\mathcal{A}_1$ sends st to the challenger where $|\mathsf{st}| < S$

4. **Second Response Phase**

(a) The challenger sends $(\mathsf{aux}, \mathsf{sk}, \mathsf{st})$ to the second adversary $\mathcal{A}_2$.

(b) $\mathcal{A}_2$ outputs $\beta' \in \{0,1\}$. $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins if $\beta' = \beta$

G_6:

1. **Initialization Phase**

   (a) The adversary chooses a bit $\beta \leftarrow \{0,1\}$

   (b) The first adversary $\mathcal{A}_1$ sends (λ, S) to the challenger.

   (c) The challenger computes the following:

   - $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $(\mathsf{pk}_{i,\gamma}, \mathsf{sk}_{i,\gamma}) \leftarrow \mathsf{IncPKE.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}) \leftarrow \mathsf{IncCCA.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK}.S_1(1^\lambda)$
   - $s \leftarrow \mathsf{PRJ.KeyGen}(1^\lambda, 1^S)$ where $s \in \{0,1\}^{l_s}$
   - $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}\big(s, \mathbf{0}\big)$
   - $\mathsf{IncKDMCCA.sk} := s$
   - $\mathsf{IncKDMCCA.pk} := \big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big)$

   (d) Challenger maintains a database τ storing ciphertexts returned for encryption queries. Challenger sends $\mathsf{IncKDMCCA.pk}$ to $\mathcal{A}_1$

   (e) $\mathcal{A}_1$ sends aux to the challenger

2. **Challenge Phase**
   Encryption queries:

   - Challenger receives the query (f_0, f_1)
   - $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
   - ~~$(\tilde{C}_{f_\beta}, (\mathsf{lab}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, f_\beta)$~~
   - ~~$\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $r_{i,\gamma} \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,\gamma}, \mathsf{lab}_{i,\gamma}; r_{i,\gamma})$~~
   - ~~Let $x = (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$~~
   - ~~$\Pi \leftarrow \mathsf{NIZK}.S_2(\mathsf{td}, x)$~~
   - $\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, \mathbf{0}\big)$
   - Challenger adds $\mathsf{ct}_{\mathsf{CCA}}$ to database τ sends $\mathsf{ct}_{\mathsf{CCA}}$ as response to the query

   Decryption queries:

   - Challenger receives the query ct. If ct is in database τ, return $\perp$
   - $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
   - Compute $\big(\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\big) = \mathsf{PRJ.Dec}(s, \mathsf{ct}_{\mathsf{sk}})$
   - Compute $\big((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}), \Pi\big) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct})$
   - Let $x = (pk_{i,\gamma}, ct_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$. If $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) \neq 1$: return $\perp$

- $\forall i \in [l_s]$:
  Compute $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,0}, \mathsf{sk}_{i,0}, ct_{i,0})$
- Return $m = \mathsf{Eval}(\tilde{C}, (\mathsf{lab}_i)_{i\in[l_s]})$ as response to decryption query

3. **First Response Phase** $\mathcal{A}_1$ sends st to the challenger where $|\mathsf{st}| < S$
4. **Second Response Phase**

   (a) The challenger sends $(\mathsf{aux}, \mathsf{sk}, \mathsf{st})$ to the second adversary $\mathcal{A}_2$.
   (b) $\mathcal{A}_2$ outputs $\beta' \in \{0,1\}$. $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins if $\beta' = \beta$

G_7 :

1. The adversary chooses a bit $\beta \leftarrow \{0,1\}$
2. **Initialization Phase**

   (a) The first adversary $\mathcal{A}_1$ sends (λ, S) to the challenger.
   (b) The challenger computes the following:

   - $\forall (i,\gamma) \in [l_s] \times \{0,1\}$: sample $(\mathsf{pk}_{i,\gamma}, \mathsf{sk}_{i,\gamma}) \leftarrow \mathsf{IncPKE.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}) \leftarrow \mathsf{IncCCA.KeyGen}(1^\lambda, 1^S)$
   - $(\mathsf{crs}) \leftarrow \mathsf{NIZK.K}(1^\lambda)$
   - $s \leftarrow \mathsf{PRJ.KeyGen}(1^\lambda, 1^S)$ where $s \in \{0,1\}^{l_s}$
   - $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}\big(s, \mathbf{0}\big)$
   - $\mathsf{IncKDMCCA.sk} := s$
   - $\mathsf{IncKDMCCA.pk} := \big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big)$

   (c) Challenger maintains a database τ storing ciphertexts returned for encryption queries. Challenger sends $\mathsf{IncKDMCCA.pk}$ to $\mathcal{A}_1$
   (d) $\mathcal{A}_1$ sends aux to the challenger

3. **Challenge Phase**
   Encryption queries:

   - Challenger receives the query (f_0, f_1)
   - $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
   - $\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, \mathbf{0}\big)$
   - Challenger adds $\mathsf{ct}_{\mathsf{CCA}}$ to database τ sends $\mathsf{ct}_{\mathsf{CCA}}$ as response to the query

   Decryption queries:

   - Challenger receives the query ct. If ct is in database τ, return $\perp$
   - $\big(\mathsf{pk}_{\mathsf{CCA}}, (\mathsf{pk}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \mathsf{crs}, \mathsf{ct}_{\mathsf{sk}}\big) = \mathsf{IncKDMCCA.pk}, s = \mathsf{IncKDMCCA.sk}$
   - Compute $\big(\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}\big) = \mathsf{PRJ.Dec}(s, \mathsf{ct}_{\mathsf{sk}})$

- Compute $\left((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}), \Pi\right) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct})$
- Let $x = (pk_{i,\gamma}, ct_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$. If $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) \neq 1$: return $\perp$
- $\forall i \in [l_s]$:
  Compute $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,0}, \mathsf{sk}_{i,0}, ct_{i,0})$
- Return $m = \mathsf{Eval}(\tilde{C}, (\mathsf{lab}_i)_{i\in[l_s]})$ as response to decryption query

4. **First Response Phase** $\mathcal{A}_1$ sends st to the challenger where $|\mathsf{st}| < S$
5. **Second Response Phase**
   (a) The challenger sends $(\mathsf{aux}, \mathsf{sk}, \mathsf{st})$ to the second adversary $\mathcal{A}_2$.
   (b) $\mathcal{A}_2$ outputs $\beta' \in \{0,1\}$. $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins if $\beta' = \beta$

Below we describe the hybrids in brief and the security proof. For ease of notation let $p_{\mathcal{A},i}$ denote the probability of $\mathcal{A}$ outputting 0 in game G_i:

$\mathbf{G_0}$: This is the CCA2 incompressible KDM security game

$\mathbf{G_1}$: This is similar to game G_0, but we change to the simulated key generation and proof algorithms for NIZK. Instead of sampling $(\mathsf{crs}) \leftarrow \mathsf{NIZK.K}(1^\lambda)$ and $\Pi \leftarrow \mathsf{NIZK.P}(\mathsf{crs}, x, w)$, we switch to $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK}.S_1(1^\lambda)$ and $\Pi \leftarrow \mathsf{NIZK}.S_2(\mathsf{crs}, x, w)$. Note that this removes the dependence of the proof generated on lab_i. When we switch later from lab_i to lab_{i,s_i}, the NIZK proof still remains independent of s

Claim 10. *For any p.p.t adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : |p_{\mathcal{A},0} - p_{\mathcal{A},1}| \leq \mathrm{negl}(\lambda)$

Proof. This follows from the zero knowledgeness of NIZK. If there exists an adversary $\mathcal{A}$ which can distinguish between G_0, G_1 with non-negligible probability, we can create a reduction which can win against the zero-knowledge challenger for NIZK. Hence G_0 and G_1 are computationally indistinguishable □

$\mathbf{G_2}$: This is similar to game G_1, but we change to Garble instead of Sim and replace lab_i from Sim with lab_{i,s_i} from Garble. Note that $\mathsf{lab}_{i,1\oplus s_i}$ is not used anywhere in G_2 but will be used in further games. Here, $\forall (i,\gamma) \in [l_s] \times \{0,1\} : ct_{i,\gamma} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,\gamma}, \mathsf{lab}_{i,s_i}; r_{i,\gamma})$

Claim 11. *For any p.p.t adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : |p_{\mathcal{A},1} - p_{\mathcal{A},2}| \leq \mathrm{negl}(\lambda)$

Proof. This follows from the security of $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$. An adversary $\mathcal{A}$ which can distinguish between G_1, G_2 can be used to build a reduction which can win the security game for $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ □

$\mathbf{G_3}$: This is similar to game G_2, but there is a subtlety here. While $(\mathsf{sk}_{i,s_i})_{i\in[l_s]}$ is still present in other parts of the encryption, decryption algorithm, we however are not using $(\mathsf{sk}_{i,1\oplus s_i})_{i\in[l_s]}$ elsewhere. Hence, a reduction argument can be made for our transition to $\mathsf{ct}_{i,1\oplus s_i} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,1\oplus s_i}, \mathsf{lab}i, 1\oplus s_i; r_{i,1\oplus s_i})$ from$\mathsf{ct}_{i,1\oplus s_i} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,1\oplus s_i}, \mathsf{lab}i, s_i; r_{i,1\oplus s_i})$.

Claim 12. *For any p.p.t adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : |p_{\mathcal{A},2} - p_{\mathcal{A},3}| \leq \mathrm{negl}(\lambda)$

Proof. This follows from the Incompressible CPA PKE security for IncPKE. For each $i \in [l_s]$, we can sequentially transition from $\mathsf{ct}_{i,1\oplus s_i} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,1\oplus s_i}, \mathsf{lab}_{i,s_i}; r_{i,1\oplus s_i})$ to $\mathsf{ct}_{i,1\oplus s_i} \leftarrow \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,1\oplus s_i}, \mathsf{lab}_{i,1\oplus s_i}; r_{i,1\oplus s_i})$. If there is any adversary $\mathcal{A}$ which can differentiate between any of these transitions with non-negligble probability, we can create a reduction for the Incompressible PKE security game and win with a non-negligible advantage □

$\mathbf{G_4}$: This is similar to game G_3, but now we alter the decryption queries such that $\mathsf{lab}_i = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,0}, \mathsf{sk}_{i,0}, \mathsf{ct}_{i,0})$ instead of $\mathsf{IncPKE.Dec}(\mathsf{pk}_{i,s_i}, \mathsf{sk}_{i,s_i}, \mathsf{ct}_{i,s_i})$. After this transition, the decryption queries are also independent of s.

For the sequence of games G_3, G_4, G_5, G_6, G_7, we also define the following the bad decryption query events $\mathsf{BDQ}_i : i \in \{4, 5, 6, 7\}$ where BDQ_i is the following:

In game G_i, $\mathcal{A}$ makes a decryption query $\mathsf{ct} \notin \tau$ such that:

1. $\big((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}), \Pi\big) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct}), x = (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$ and
2. $\mathsf{NIZK.V}(\mathsf{crs}, x, \Pi) = 1$ and
3. $\exists i^* \in [l_s]$ such that $\mathsf{IncPKE.Dec}(\mathsf{pk}_{i,0}, \mathsf{sk}_{i,0}, \mathsf{ct}_{i,0}) = \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,1}, \mathsf{sk}_{i,1}, \mathsf{ct}_{i,1})$

Claim 13. *For any p.p.t adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : |p_{\mathcal{A},3} - p_{\mathcal{A},4}| \leq \Pr[\mathsf{BDQ}_4]$

Proof. The games G_3 and G_4 are identical if the event BDQ_4 does not occur. Hence, the above inequality must be valid. □

$\mathbf{G_5}$: At this point, we have completely removed s from the encryption and decryption queries. Hence, we can alter $\mathsf{ct}_{\mathsf{sk}}$ using the fact that $\mathsf{PRJ.Enc}$ is KDM-incompressible secure. In G_5, we transition from $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}(s, (\mathsf{sk}_{\mathsf{CCA}}, (\mathsf{sk}_{i,s_i})_{i\in[l_s]}))$ to $\mathsf{ct}_{\mathsf{sk}} \leftarrow \mathsf{PRJ.Enc}(s, \mathbf{0})$. We notice that $\mathsf{ct}_{\mathsf{sk}}$ in G_4 is an encryption of a projection on s as each of the bits of sk_{i,s_i} depends on s_i. Hence, for the reduction to work, we crucially require that PRJ be an Incompressible KDM secure scheme for the class of projective functions(where each bit of the message is dependent on at most a single bit of the encryption key s

Claim 14. *For any p.p.t adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : |p_{\mathcal{A},4} - p_{\mathcal{A},5}| \leq \mathrm{negl}(\lambda)$

Proof. This holds because of the Incompressible KDM secure property of PRJ. Using an adversary $\mathcal{A}$ which can distinguish between G_4, G_5, we can emulate a reduction that can win the Incompressible SKE game for PRJ. Note that this was transition was not possible in the earlier games due to the presence □

Claim 15. *For any p.p.t adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : |\Pr[\mathsf{BDQ}_4] - \Pr[\mathsf{BDQ}_5]| \leq \mathrm{negl}(\lambda)$

Proof. This again holds due to the projective KDM Incompressible SKE security of PRJ. If there is an adversary that can make Bad Decryption Queries with non-negligible difference in probabilities for G_4, G_5, then I can build a reduction against the challenger for projective KDM Incompressible SKE game of PRJ that simulates the entire encryption/decryption oracle as all the terms including the keys $\mathsf{sk}_{\mathsf{CCA}}$ and $\mathsf{sk}_{i,0}$ are independent of s and if it sees a bad-decryption query, it guesses bit 0, else it guesses bit 1 and can distinguish between G_4, G_5 with non-negligible probability. Hence, the above claim holds by contradiction. □

$\mathbf{G_6}$: After G_5 we see that $\mathsf{sk}_{\mathsf{CCA}}$ has been eliminated from $\mathsf{ct}_{\mathsf{sk}}$. Hence, we can now replace $\mathsf{ct}_{\mathsf{CCA}}$ with an encryption of all zeros, and the games G_5, G_6 would be indistinguishable by the Incompressible CCA security of IncCCA. The transition from G_5 to G_6 replaces $\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, ((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_{f_\beta}), \Pi)\big)$ with $\mathsf{ct}_{\mathsf{CCA}} \leftarrow \mathsf{IncCCA.Enc}\big(\mathsf{pk}_{\mathsf{CCA}}, \mathbf{0}\big)$

Claim 16. *For any p.p.t adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : |p_{\mathcal{A},5} - p_{\mathcal{A},6}| \leq \mathrm{negl}(\lambda)$

Proof. G_5 and G_6 are indisinguishable due to the Incompressible CCA security of IncCCA. If there is an adversary $\mathcal{A}$ that can distinguish between G_5, G_6, we construct a reduction that simulates the $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ queries and queries the incompressible CCA challenger with decryption queries as required in the $\mathsf{IncKDMCCA.Dec}$ algorithm and for encryption queries it sends $\big(m_0, m_1\big) = \big(((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_{f_\beta}), \Pi)\big)$ and wins against the challenger if $\mathcal{A}$ guesses correctly for the $\mathsf{IncKDMCCA}$ game of G_5, G_6 □

Claim 17. *For any p.p.t adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : |\Pr[\mathsf{BDQ}_5] - \Pr[\mathsf{BDQ}_6]| \leq \mathrm{negl}(\lambda)$

Proof. The proof follows again due to the Incompressible CCA security of IncCCA. Let $\mathcal{A}$ be such that $|\Pr[\mathsf{BDQ}_5] - \Pr[\mathsf{BDQ}_6]|$ is non-negligible for $\mathcal{A}$. We can emulate the reduction adversary B against the Incompressible CCA challenger to which we send $\big(m_0, m_1\big) = \big(((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}_{f_\beta}), \Pi)\big)$ whenever encryption queries are made by $advA$ and decryption queries are made as usual. If $\mathcal{A}$ makes a bad-decryption query, we guess bit 0 for the Incompressible CCA game, else we guess bit 1 and we will win with a probability non-negligibly differing from $\frac{1}{2}$ □

$\mathbf{G_7}$: We finally revert to the non-simulated proof for NIZK. We crucially need to transition to this hybrid because we need the soundness property of NIZK to bound the probability of the bad decryption query (BDQ) event. Soundness for NIZKs holds only when the proof

is not simulated. We transition back to $(\mathsf{crs}) \leftarrow \mathsf{NIZK.K}(1^\lambda)$ from $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK}.S_1(1^\lambda)$ in $\mathsf{IncKDMCCA.KeyGen}$ and would have transitioned to$\Pi \leftarrow \mathsf{NIZK.P}(\mathsf{crs}, x, w)$ from $\Pi \leftarrow \mathsf{NIZK}.S_2(\mathsf{td}, x)$, but we have already eliminated $\mathsf{NIZK.P}(\mathsf{crs}, x, w)$ in G_6, hence we only see the change from $(\mathsf{crs}, \mathsf{td})$ to (crs) in our game.

Claim 18. *For any p.p.t adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : |p_{\mathcal{A},6} - p_{\mathcal{A},7}| \leq \mathrm{negl}(\lambda)$

Proof. This trivially follows from the zero-knowledgeness property of NIZK where we replace $\mathsf{NIZK}.S_1$ with $\mathsf{NIZK.K}$ and $\mathsf{NIZK}.S_2$ with $\mathsf{NIZK.P}$. However, note that we do not use $\mathsf{NIZK}.S_2$ in encryption queries for G_6 and hence, only the change from $\mathsf{NIZK.K}$ to $\mathsf{NIZK}.S_2$ is mentioned. The reduction is trivial. □

Claim 19. *For any p.p.t adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : |\Pr[\mathsf{BDQ}_6] - \Pr[\mathsf{BDQ}_7]| \leq \mathrm{negl}(\lambda)$

Proof. If there is an adversary $\mathcal{A}$ such that $|\Pr[\mathsf{BDQ}_6] - \Pr[\mathsf{BDQ}_7]|$ is not negligible, we can construct a reduction against the zero-knowledge game for NIZK, using the crs given by the challenger. On returning the bit the adversary $\mathcal{A}$ guesses, we win with an advantage $\frac{|\Pr[\mathsf{BDQ}_6] - \Pr[\mathsf{BDQ}_7]|}{2}$ □

Finally, we bound the probabilities $p_{\mathcal{A},7}$ and $\Pr[\mathsf{BDQ}_7]$

Claim 20. *For any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : p_{\mathcal{A},7} = \frac{1}{2}$

Proof. All the information in the view of adversary $\mathcal{A}$ is independent of the bit β chosen by the challenger in G_7. Hence, no adversary can guess the bit β with a probability other than $\frac{1}{2}$ □

Claim 21. *For any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : \Pr[\mathsf{BDQ}_7] \leq \mathrm{negl}(\lambda)$

Proof. The event BDQ_7 is the following:
In the game G_7, $\mathcal{A}$ makes a decryption query $\mathsf{ct} \notin \tau$ such that

$$\mathsf{NIZK.V}(\mathsf{crs}, (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \Pi) = 1$$

and $\exists i^* \in [l_s]$ such that $\mathsf{IncPKE.Dec}(\mathsf{pk}_{i,0}, \mathsf{sk}_{i,0}, \mathsf{ct}_{i,0}) \neq \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,1}, \mathsf{sk}_{i,1}, \mathsf{ct}_{i,1})$ where $\big((\mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, (\tilde{C}), \Pi\big) = \mathsf{IncCCA.Dec}(\mathsf{pk}_{\mathsf{CCA}}, \mathsf{sk}_{\mathsf{CCA}}, \mathsf{ct})$

Now, the condition that $\exists i^* \in [l_s] : \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,0}, \mathsf{sk}_{i,0}, \mathsf{ct}_{i,0}) \neq \mathsf{IncPKE.Dec}(\mathsf{pk}_{i,1}, \mathsf{sk}_{i,1}, \mathsf{ct}_{i,1})$ implies that that $(\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}$ is not in the NP language:

- $\mathcal{L} = \big\{(\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}} | \forall i \in [l_s] : \exists(\mathsf{lab}_i, r_{i,0}, r_{i,1}) :$
  $\mathsf{ct}_{i,0} = \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,0}, \mathsf{lab}_i; r_{i,0}), \mathsf{ct}_{i,1} = \mathsf{IncPKE.Enc}(\mathsf{pk}_{i,1}, \mathsf{lab}_i; r_{i,1})\big\}$

which is the language for which NIZK gives proofs. Hence, by the soundness property of NIZK, $\Pr[\mathsf{NIZK.V}(\mathsf{crs}, (\mathsf{pk}_{i,\gamma}, \mathsf{ct}_{i,\gamma})_{(i,\gamma)\in[l_s]\times\{0,1\}}, \Pi) = 1] \leq \text{negl}(\lambda)$.
Hence, the probability $\Pr[\mathsf{BDQ}_7] \leq \text{negl}(\lambda)$ □

Finally, we compile the above claims to show $p_{\mathcal{A},0} \leq \frac{1}{2} + \text{negl}(\lambda)$.
We observe that $\Pr[\mathsf{BDQ}_4] \leq \text{negl}(\lambda)$ as $\Pr[\mathsf{BDQ}_4] - \Pr[\mathsf{BDQ}_5]| \leq \text{negl}(\lambda)$, $\Pr[\mathsf{BDQ}_5] - \Pr[\mathsf{BDQ}_6]| \leq \text{negl}(\lambda)$, $\Pr[\mathsf{BDQ}_6] - \Pr[\mathsf{BDQ}_7]| \leq \text{negl}(\lambda)$, $\Pr[\mathsf{BDQ}_7] \leq \text{negl}(\lambda)$.

$$\text{Hence, } \Pr[\mathsf{BDQ}_4] \leq \text{negl}(\lambda)$$

Using the above, we get $|p_{\mathcal{A},3} - p_{\mathcal{A},4}| \leq \Pr[\mathsf{BDQ}_4] \leq \text{negl}(\lambda)$

Finally, we have $p_{\mathcal{A},7} = \frac{1}{2}, |p_{\mathcal{A},7} - p_{\mathcal{A},6}| \leq \text{negl}(\lambda), |p_{\mathcal{A},6} - p_{\mathcal{A},5}| \leq \text{negl}(\lambda), |p_{\mathcal{A},5} - p_{\mathcal{A},4}| \leq \text{negl}(\lambda), |p_{\mathcal{A},4} - p_{\mathcal{A},3}| \leq \text{negl}(\lambda), |p_{\mathcal{A},3} - p_{\mathcal{A},2}| \leq \text{negl}(\lambda), |p_{\mathcal{A},2} - p_{\mathcal{A},1}| \leq \text{negl}(\lambda), |p_{\mathcal{A},1} - p_{\mathcal{A},0}| \leq \text{negl}(\lambda)$. Using these, we can show that

$$|p_{\mathcal{A},0}| \leq \frac{1}{2} + \text{negl}(\lambda)$$

Hence, $\mathsf{IncKDMCCA}$ is Incompressible KDM CCA secure

□

Chapter 6

LR-KDM from Hash Proof Systems

In this chapter, we introduce a construction for Leakage-Resilient Key Dependent Message secure public key encryption for the class of affine functions from DDH and d-LIN assumptions via a cryptographic primitive known as Leakage-Resilient Smooth Hash Proof System. In the first section, we provide the preliminaries and required definitions for understanding our construction. In the subsequent section, we provide a construction for leakage-resilient smooth hash proof systems. Finally, we construct a Leakage-Resilient $\mathcal{F}$-KDM secure public key encryption scheme from DDH via Leakage-resilient smooth HPS where $\mathcal{F}$ refers to the class of affine functions

6.1 Definitions

Below we present a few preliminary definitions and notions which are pre-requisite to the construction

6.1.1 Leakage-Resilient Hash Proof System

A hash proof system $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$ associated with the 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 $\mathsf{k} \in \mathcal{L}'$
- $\mathsf{Decaps}(\mathsf{sk}, x)$: The decoding algorithm takes as input the secret key sk, an input string $x \in \mathcal{L} \cup \overline{\mathcal{L}}$ and string and outputs a string $\mathsf{k} \in \mathcal{L}'$.

A leakage-resilient (homomorphic) hash proof system must satisfy the following properties:

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

$$\Pr[\mathsf{Encaps}(\mathsf{pk}, (x, w)) = \mathsf{Decaps}(\mathsf{sk}, x)] \geq 1 - \mathrm{negl}(\lambda)$$

Homomorphism For all $\lambda : \forall(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\forall x, y$ in the domain of $\mathsf{Decaps}(\mathsf{sk}, .)$,

$$\mathsf{Decaps}(\mathsf{sk}, x \odot y) = \mathsf{Decaps}(\mathsf{sk}, x) \odot \mathsf{Decaps}(\mathsf{sk}, y)$$

Leakage-Resilient Smoothness A hash proof system is $\ell(\lambda)$-smooth if for large enough λ, any (leakage) function f such that the size of the output of f is $\ell(\lambda)$

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

where the distribution is over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $x \leftarrow \overline{\mathcal{L}}$ and $t \leftarrow \mathcal{L}'$.

We also require that the language $\mathcal{L}$ is subgroup indisinguishable

Subgroup Indistinguishability We say that $\mathcal{L}$ is subgroup indistinguishable if the distributions $\mathcal{D}_1 = \{x : x \leftarrow \mathcal{L}\}$ and $\mathcal{D}_2 = \{x : x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}\}$ are computationally indistinguishable

6.1.2 Leakage-Resilient KDM PKE

Here, we briefly describe Leakage-Resilient KDM PKEs along with its formal security game. Leakage-Resilient KDM PKEs are encryption schemes which are $\mathcal{F}$-KDM secure for some class of functions $\mathcal{F}$ even in the presence of bounded adversarial leakage of the bits of the secret key. Note that by adversarial leakage, we do not restrict ourselves to the actual bits present in the secret key. The adversary also has the freedom to compute arbitrary functions over secret key, the only limitation the adversary should encounter is the bound on the amount of leakage.

6.2 Constructing LR-smooth homomorphic HPS from DDH

The language $\mathcal{L}$ Given a group generator g with group order p, let $\mathcal{G}$ be the corresponding group. Given a matrix $\mathbf{A} \in \mathbb{Z}_p^{m \times n}$, we define the language $\mathcal{L}_\mathbf{A}$ and its complement $\mathcal{L}_\mathbf{A}$ for our Hash Proof System as

$$\mathcal{L}_\mathbf{A} = \left\{ g^{\mathbf{Ax}} | \mathbf{x} \in \mathbb{Z}_p^n \right\}$$
$$\overline{\mathcal{L}}_\mathbf{A} = \left\{ g^{\mathbf{b}^\top} | \mathbf{b} \in \mathbb{Z}_p^m \right\} \setminus \mathcal{L}_\mathbf{A}$$

Now we define the $l(\lambda)$-smooth Hash Proof system $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$ below:

- $\mathsf{HPS.Setup}(1^\lambda, 1^{l(\lambda)})$:
  - Compute $n = n(\lambda), l = l(\lambda), p = p(\lambda) m = (n+2)\log(p) + l$ Sample $g \in \mathcal{G}, \mathbf{A} \leftarrow \mathbb{Z}_p^{m \times n}$.

  - Sample $\mathbf{s} \leftarrow \{0,1\}^m$. Compute $\mathsf{sk} = \mathbf{s}$
  - Compute $g^{\mathbf{s}^\top \mathbf{A}}$. Set $\mathsf{pk} = (g, g^{\mathbf{A}}, g^{\mathbf{s}^\top \mathbf{A}})$
  - Return $(\mathsf{pk}, \mathsf{sk})$

- $\mathsf{HPS.Encaps}(\mathsf{pk}, x = g^{\mathbf{Ax}}, w = \mathbf{x})$
  - Compute $(g, g^{\mathbf{A}}, g^{\mathbf{s}^\top \mathbf{A}}) = \mathsf{pk}$
  - Using $w = \mathbf{x}$ and $g^{\mathbf{s}^\top \mathbf{A}}$, compute and return $g^{\mathbf{s}^\top \mathbf{Ax}}$

- $\mathsf{HPS.Decaps}(\mathsf{sk}, x = g^{\mathbf{Ax}})$
  - Using $\mathbf{s} = \mathsf{sk}, x = g^{\mathbf{Ax}}$, compute and return $g^{\mathbf{s}^\top \mathbf{Ax}}$

Claim 22. *Correctness holds for* $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$ *as described above*

Proof. For any $x = g^{\mathbf{Ax}}, w = \mathbf{x}$: $\mathsf{Encaps}(\mathsf{pk}, x, w) = g^{\mathbf{s}^\top \mathbf{Ax}}$. $\mathsf{Decaps}(\mathsf{sk}, x = g^{\mathbf{Ax}}) = g^{\mathbf{s}^\top \mathbf{Ax}}$. Hence, correctness holds. □

Claim 23. *Homomorphism holds for* $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$ *as described above*

Proof. For any $y_1 = g^{\mathbf{Ax_1}}, y_2 = g^{\mathbf{Ax_2}}$:
$\mathsf{Decaps}(\mathsf{sk} = \mathbf{s}, y_1) = g^{\mathbf{s}^\top \mathbf{Ax_1}}, \mathsf{Decaps}(\mathsf{sk} = \mathbf{s}, y_2) = g^{\mathbf{s}^\top \mathbf{Ax_2}}$ and
$\mathsf{Decaps}(\mathsf{sk} = \mathbf{s}, y_1 \odot y_2) = \mathsf{Decaps}(\mathsf{sk} = \mathbf{s}, g^{\mathbf{Ax_1}} \odot g^{\mathbf{Ax_2}}) = \mathsf{Decaps}(\mathsf{sk} = \mathbf{s}, g^{\mathbf{A}(\mathbf{x_1}+\mathbf{x_2})}) = g^{\mathbf{s}^\top \mathbf{A}(\mathbf{x_1}+\mathbf{x_2})} = g^{\mathbf{s}^\top \mathbf{Ax_1}} \odot g^{\mathbf{s}^\top \mathbf{Ax_2}} = \mathsf{Decaps}(\mathsf{sk} = \mathbf{s}, y_1) \odot \mathsf{Decaps}(\mathsf{sk} = \mathbf{s}, y_2)$.
Hence, homomorphism holds true □

Claim 24. *Leakage-Resilient smoothness holds for* $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$

Proof. For leakage-resilient smoothness, we want the following to hold true:

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

where the distribution is over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda), x \in \overline{\mathcal{L}}, t \in \mathcal{L}'$
$(\mathsf{pk}, f(\mathsf{sk}), x, \mathsf{Decaps}(\mathsf{sk}, x)) = ((g, g^{\mathbf{A}}, g^{\mathbf{s}^\top \mathbf{A}}), f(s), x, g^{\mathbf{s}^\top \mathbf{Ax}})$ where $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$
Now, $g^{\mathbf{s}^\top \mathbf{Ax}}$ is a universal hash function with seed x for the input $\mathbf{s}$. We can also see x separately in our view. Now, in our view:

$$\begin{aligned}
H_\infty(s|g^{\mathbf{s}^\top \mathbf{A}}, f(s)) &\geq H_\infty(s|g^{\mathbf{s}^\top \mathbf{A}}) - |f(s)| \\
&= H_\infty(s|g^{\mathbf{s}^\top \mathbf{A}}) - l \\
&\geq H_\infty(s) - |g^{\mathbf{s}^\top \mathbf{A}})| - l \\
&= H_\infty(s) - n\log_2(p) - l \\
&= (l + (n+2)\log_2(p)) - n\log_2(p) - l \\
&= 2\log_2(p)
\end{aligned}$$

But we know that the final output for the universal hash $h(\mathbf{s}) = g^{\mathbf{s}^\top \mathbf{A}\mathbf{x}}$ has at-most $\log_2(p)$ bits of entropy. Hence, by leftover hash lemma:
$(\mathsf{pk}, f(\mathsf{sk}), x, \mathsf{Decaps}(\mathsf{sk}, x)) \approx_{\frac{1}{\mathsf{poly}(p(\lambda))}} (\mathsf{pk}, f(\mathsf{sk}), x, \mathcal{L}')$ where $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}, (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
Note that increasing the dimension m will improve the bounds for smoothness. For $p(\lambda)$ being a super-polynomial in λ, the distributions $(\mathsf{pk}, f(\mathsf{sk}), x, \mathsf{Decaps}(\mathsf{sk}, x))$ and $(\mathsf{pk}, f(\mathsf{sk}), x, \mathcal{L}')$ are $\mathrm{negl}(\lambda)$ close
Hence, HPS is $l(\lambda)$ leakage-resilient smooth □

We note that $\mathcal{L}$ is also subgroup indistinguishable from d-LIN assumptions. Namely, it is hard to distinguish between the two distributions $\left\{g^{\mathbf{A}\mathbf{x}} | \mathbf{x} \leftarrow \mathbb{Z}_p^n\right\}$ and $\left\{g^{\mathbf{b}^\top} | \mathbf{b} \in \mathbb{Z}_p^m\right\}$ for sufficiently large λ. [Wee15]

6.3 From LR-smooth HPS to LR-KDM

In this section, we discuss how to construct Leakage Resilient KDM secure encryption for the class of affine functions over the bits of the secret key. Firstly, let us recall the LR-KDM game:

Consider the following experiment with an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{F}_{\lambda,S}$ is a class of functions $f : \mathcal{SK}_{\lambda,S} \to \mathcal{M}_{\lambda,S}$ and $\mathcal{H}_{\lambda,S}$ is a class of functions $h : \mathcal{SK}_{\lambda,S} \to \mathcal{L}_S$, and $\mathcal{L}_S$ is the class of Leakage strings of length at most S.

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$
- **Leakage Phase:** $\mathcal{A}_1$ receives (pk) and outputs a function $h \in \mathcal{H}_{\lambda,S}$. The challenger outputs leakage $h(\mathsf{sk})$ such that $|h(\mathsf{sk})| < S$. $\mathcal{A}_1$ on receiving $h(\mathsf{sk})$ outputs st and sends it to $\mathcal{A}_2$
- **First Response Phase:** $\mathcal{A}_2$ receives (pk, st) and chooses function $f \in \mathcal{F}_{\lambda,S}$, along with auxiliary information aux. The challenger randomly chooses $d \in \{0, 1\}$ and computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{pk}, \mathbf{0})$ if $d = 0$, else it computes $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{pk}, f(\mathsf{sk}))$. It sends ct^* to $\mathcal{A}_3$.
- **Second Response Phase:** $\mathcal{A}_2$ receives ct^* and outputs d'. $\mathcal{A}$ wins the experiment if $d = d'$.

Correctness: Correctness argument is similar to most other PKE schemes. Specifically, $\Pr[\mathsf{Dec}(\mathsf{sk}, x) = m | x \leftarrow \mathsf{Enc}(\mathsf{pk}, m)] \geq \mathrm{negl}(\lambda)$

Security: A PKE scheme is said to be leakage-resilient key-dependent message secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, S \in \mathbb{N}$,

$$\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

Now, we show how to construct a Leakage Resilient KDM secure scheme from Leakage-Resilient smooth Hash Proof Systems. Let $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$ be an $S(\lambda)$-leakage resilient smooth HPS for the language $\mathcal{L}$ which is subgroup indistinguishable, then consider the following PKE scheme $\mathsf{LRKdm} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$

- $\mathsf{LRKdm.KeyGen}(1^\lambda, 1^{S(\lambda)})$:
  - Compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HPS.Setup}(1^\lambda, 1^{S(\lambda)})$
  - Return $(\mathsf{pk}, \mathsf{sk})$
- $\mathsf{LRKdm.Enc}(\mathsf{pk}, m)$:
  - Sample $(x, w) \leftarrow \mathcal{L}$
  - Calculate and return $(x, \mathsf{Encaps}(\mathsf{pk}, x, w) \odot m)$
- $\mathsf{LRKdm.Dec}(\mathsf{sk}, \mathsf{ct})$:
  - Compute $(x, y) = \mathsf{ct}$
  - Compute and return $(\mathsf{Decaps}(\mathsf{sk}, x))^{-1} \odot y$

Claim 25. *Correctness for PKE encryption holds for* LRKdm

Proof. For any $m \in \mathcal{L}'$: $\mathsf{LRKdm.Dec}(\mathsf{sk}, (x, \mathsf{Encaps}(\mathsf{pk}, x, w) \odot m)) = (\mathsf{Decaps}(\mathsf{sk}, x))^{-1} \odot \mathsf{Encaps}(\mathsf{pk}, x, w) \odot m = m$ by the correctness of $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$ □

Theorem 6.3.1. *Given $S(\lambda)$-leakage resilient smooth Hash Proof System* $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$ *the encryption scheme* $\mathsf{LRKdm} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is $S(\lambda)$-leakage resilient smooth$\mathcal{F}$-KDM PKE secure for the following class of functions :*

$$\mathcal{F} = \{f_{x,y} : f_{x,y}(\mathsf{sk}) = \mathsf{Decaps}(\mathsf{sk}, x) \odot y\}$$

Proof. We prove the security through a series of hybrid games:
$\mathbf{G_0}$: This is the original LR-KDM PKE game. Consider an adversary $\mathcal{A}$:

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$. The challenger also chooses a bit $d \leftarrow \{0, 1\}$
- **Leakage Phase:** $\mathcal{A}_1$ receives (pk) and outputs a function $h \in \mathcal{H}_{\lambda,S}$. The challenger outputs leakage $h(\mathsf{sk})$ such that $|h(\mathsf{sk})| < S$.
- **Query Phase:**
  - $\mathcal{A}$ sends function query $(f_{a_0,a_1}, f_{a_1,b_1}) \in \mathcal{F}_{\lambda,S}$
  - The challenger samples $(x, w) \leftarrow \mathcal{L}$

– The challenger computes the encryption of $f_{a_d,b_d}(\mathsf{sk})$ as

$$\mathsf{ct}^* = (x, \mathsf{Encaps}(\mathsf{pk}, x, w) \odot f_{a_d,b_d}) = (x, \mathsf{Encaps}(\mathsf{pk}, x, w) \odot (\mathsf{Decaps}(\mathsf{sk}, a_d) \odot b_d))$$

– The challenger returns ct^* to $\mathcal{A}$

- **Challenge Phase:** $\mathcal{A}$ guesses bit d'. $\mathcal{A}$ wins the experiment if $d = d'$.

$\mathbf{G_1}$:

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$. The challenger also chooses a bit $d \leftarrow \{0, 1\}$
- **Leakage Phase:** $\mathcal{A}_1$ receives (pk) and outputs a function $h \in \mathcal{H}_{\lambda,S}$. The challenger outputs leakage $h(\mathsf{sk})$ such that $|h(\mathsf{sk})| < S$.
- **Query Phase:**
  – $\mathcal{A}$ sends function query $(f_{a_0,a_1}, f_{a_1,b_1}) \in \mathcal{F}_{\lambda,S}$
  – The challenger samples $(x, w) \leftarrow \mathcal{L}$
  – The challenger computes the encryption of $f_{a_d,b_d}(\mathsf{sk})$ as
  $\mathsf{ct}^* = (x, \mathsf{Decaps}(\mathsf{sk}, x) \odot \mathsf{Decaps}(\mathsf{sk}, a_d) \odot b_d)$
  – The challenger returns ct^* to $\mathcal{A}$
- **Challenge Phase:** $\mathcal{A}$ guesses bit d'. $\mathcal{A}$ wins the experiment if $d = d'$.

Claim 26. *Hybrids G_0, G_1 are statistically indistinguishable*

Proof. This follows from the correctness of HPS as for $(x, w) \leftarrow \mathcal{L}$, with $1 - \mathrm{negl}(\lambda)$ probability, $\mathsf{Encaps}(\mathsf{pk}, x, w) = \mathsf{Decaps}(\mathsf{sk}, x)$ □

$\mathbf{G_2}$:

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$. The challenger also chooses a bit $d \leftarrow \{0, 1\}$
- **Leakage Phase:** $\mathcal{A}_1$ receives (pk) and outputs a function $h \in \mathcal{H}_{\lambda,S}$. The challenger outputs leakage $h(\mathsf{sk})$ such that $|h(\mathsf{sk})| < S$.
- **Query Phase:**
  – $\mathcal{A}$ sends function query $(f_{a_0,a_1}, f_{a_1,b_1}) \in \mathcal{F}_{\lambda,S}$
  – The challenger samples $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$
  – The challenger computes the encryption of $f_{a_d,b_d}(\mathsf{sk})$ as
  $\mathsf{ct}^* = (x, \mathsf{Decaps}(\mathsf{sk}, x) \odot \mathsf{Decaps}(\mathsf{sk}, a_d) \odot b_d)$
  – The challenger returns ct^* to $\mathcal{A}$

- **Challenge Phase:** $\mathcal{A}$ guesses bit d'. $\mathcal{A}$ wins the experiment if $d = d'$.

Claim 27. *Hybrids G_1, G_2 are computationally indistinguishable*

Proof. This follows from the subgroup indistinguishability property of the language $\mathcal{L}$ which states that the distributions $\mathcal{D}_1 = \{x : x \in \mathcal{L}\}$ and $\mathcal{D}_2 = \{x : x \in \mathcal{L} \cup \overline{\mathcal{L}}\}$ are computationally indistinguishable. □

G_3 :

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$. The challenger also chooses a bit $d \leftarrow \{0, 1\}$
- **Leakage Phase:** $\mathcal{A}_1$ receives (pk) and outputs a function $h \in \mathcal{H}_{\lambda,S}$. The challenger outputs leakage $h(\mathsf{sk})$ such that $|h(\mathsf{sk})| < S$.
- **Query Phase:**
  - $\mathcal{A}$ sends function query $(f_{a_0,a_1}, f_{a_1,b_1}) \in \mathcal{F}_{\lambda,S}$
  - The challenger samples $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$
  - The challenger computes the encryption of $f_{a_d,b_d}(\mathsf{sk})$ as $\mathsf{ct}^* = (x, \mathsf{Decaps}(\mathsf{sk}, x \odot a_d) \odot b_d)$
  - The challenger returns ct^* to $\mathcal{A}$
- **Challenge Phase:** $\mathcal{A}$ guesses bit d'. $\mathcal{A}$ wins the experiment if $d = d'$.

Claim 28. *Hybrids G_2, G_3 are identical*

Proof. This follows from the homomorphic property of HPS. □

G_4 :

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$. The challenger also chooses a bit $d \leftarrow \{0, 1\}$
- **Leakage Phase:** $\mathcal{A}_1$ receives (pk) and outputs a function $h \in \mathcal{H}_{\lambda,S}$. The challenger outputs leakage $h(\mathsf{sk})$ such that $|h(\mathsf{sk})| < S$.
- **Query Phase:**
  - $\mathcal{A}$ sends function query $(f_{a_0,a_1}, f_{a_1,b_1}) \in \mathcal{F}_{\lambda,S}$
  - The challenger samples $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$
  - The challenger computes the encryption of $f_{a_d,b_d}(\mathsf{sk})$ as $\mathsf{ct}^* = (x \odot (a_d)^{-1}, \mathsf{Decaps}(\mathsf{sk}, x) \odot b_d)$
  - The challenger returns ct^* to $\mathcal{A}$

- **Challenge Phase:** $\mathcal{A}$ guesses bit d'. $\mathcal{A}$ wins the experiment if $d = d'$.

Claim 29. *Hybrids G_3, G_4 are identical*

Proof. Since the distributions $\mathcal{D}_1 = x : x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$, $\mathcal{D}_2 = x \odot (a_d)^{-1} : x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$ are identical, hybrids G_3, G_4 are identical. □

G_5 :

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$. The challenger also chooses a bit $d \leftarrow \{0, 1\}$
- **Leakage Phase:** $\mathcal{A}_1$ receives (pk) and outputs a function $h \in \mathcal{H}_{\lambda,S}$. The challenger outputs leakage $h(\mathsf{sk})$ such that $|h(\mathsf{sk})| < S$.
- **Query Phase:**
  - $\mathcal{A}$ sends function query $(f_{a_0,a_1}, f_{a_1,b_1}) \in \mathcal{F}_{\lambda,S}$
  - The challenger samples $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$
  - The challenger samples $r \leftarrow \mathcal{L}'$ and computes the encryption of $f_{a_d,b_d}(\mathsf{sk})$ as $\mathsf{ct}^* = (x \odot (a_d)^{-1}, r \odot b_d)$
  - The challenger returns ct^* to $\mathcal{A}$
- **Challenge Phase:** $\mathcal{A}$ guesses bit d'. $\mathcal{A}$ wins the experiment if $d = d'$.

Claim 30. *Hybrids G_4, G_5 are statistically indistinguishable*

Proof. This follows from Leakage-Resilient smoothness of HPS HPS. In the adversary's view, we have $(\mathsf{pk}, h(\mathsf{sk}), \mathsf{ct}^* = (x \odot (a_d)^{-1}, \mathsf{Decaps}(\mathsf{sk}, x) \odot b_d))$. We know that $\mathsf{pk}, h(\mathsf{sk}), x, \mathsf{Decaps}(\mathsf{sk}, x)$ is statistically indistinguishable from $(\mathsf{pk}, h(\mathsf{sk}), x, r)$ for some $r \leftarrow \mathcal{L}'$. Hence,$(\mathsf{pk}, h(\mathsf{sk}), \mathsf{ct}^* = (x \odot (a_d)^{-1}, \mathsf{Decaps}(\mathsf{sk}, x) \odot b_d))$ must be indistinguishable from $(\mathsf{pk}, h(\mathsf{sk}), \mathsf{ct}^* = (x \odot (a_d)^{-1}, r \odot b_d))$ else we can create a reduction which breaks the LR-smoothness of HPS. □

G_6 :

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$ and sends pk to $\mathcal{A}_1$. The challenger also chooses a bit $d \leftarrow \{0, 1\}$
- **Leakage Phase:** $\mathcal{A}_1$ receives (pk) and outputs a function $h \in \mathcal{H}_{\lambda,S}$. The challenger outputs leakage $h(\mathsf{sk})$ such that $|h(\mathsf{sk})| < S$.
- **Query Phase:**
  - $\mathcal{A}$ sends function query $(f_{a_0,a_1}, f_{a_1,b_1}) \in \mathcal{F}_{\lambda,S}$
  - The challenger samples $x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}$

  - The challenger samples $r \leftarrow \mathcal{L}'$ and computes the encryption of $f_{a_d,b_d}(\mathsf{sk})$ as $\mathsf{ct}^* = (x, r)$
  - The challenger returns ct^* to $\mathcal{A}$

- **Challenge Phase:** $\mathcal{A}$ guesses bit d'. $\mathcal{A}$ wins the experiment if $d = d'$.

Claim 31. *Hybrids G_5, G_6 are identical*

Proof. This follows as the distributions $\{(x, r) : x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}, r \leftarrow \mathcal{L}'\}$ and $\{(x \odot (a_d)^{-1}, r \odot (b_d)) : x \leftarrow \mathcal{L} \cup \overline{\mathcal{L}}, r \leftarrow \mathcal{L}'\}$ are identical □

Claim 32. *No adversary can win G_6 with prob. $> \frac{1}{2}$*

Proof. In game G_6, the bit d is information theoretically hidden from the adversary's view as ct^* is sampled independently. Hence for all adversaries $\mathcal{A}$, prob. that $\mathcal{A}$ wins G_6 is exactly $\frac{1}{2}$ □

From the above claims, we can see that the probability of any p.p.t $\mathcal{A}$ winning in G_0 is less than $\frac{1}{2} + \mathrm{negl}(\lambda)$. Hence, LRKdm is a Leakage-Resilient KDM secure PKE schem. □

Finally we show that the HPS constructed in the previous section can be used to build LR-KDM secure PKE for affine functions. We know that HPS defined over $\mathcal{L}_{\mathbf{A}}$ is LR-smooth and homomorphic. Hence, LRKdm is leakage-resilient $\mathcal{F}-$KDM secure for the following function class $\mathcal{F}$:

$$\mathcal{F} = \{f_{x,y} : f_{x,y}(\mathsf{sk}) = \mathsf{Decaps}(\mathsf{sk}, x) \odot y\}$$

$$\mathcal{F} = \{f_{x,y} : f_{x,y}(\mathsf{sk}) = g^{s^\top \mathbf{x}} \odot g^{\mathbf{y}}\}$$

$$\mathcal{F} = \{f_{x,y} : f_{x,y}(\mathsf{sk}) = g^{s^\top \mathbf{x}+\mathbf{y}}\}$$

which is the class of affine functions from $\{0,1\}^m$ to $\mathcal{G}$. Hence, we have shown a transformation from DDH to LR-KDM HPS to a secure LR-KDM PKE scheme over the class of affine functions.

Bibliography

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous Hardcore Bits and Cryptography against Memory Attacks. In Omer Reingold, editor, Theory of Cryptography, Lecture Notes in Computer Science, pages 474–495, Berlin, Heidelberg, 2009. Springer.

[BDD22] Pedro Branco, Nico Döttling, and Jesko Dujmović. Rate-1 Incompressible Encryption from Standard Assumptions. In Eike Kiltz and Vinod Vaikuntanathan, editors, Theory of Cryptography, Lecture Notes in Computer Science, pages 33–69, Cham, 2022. Springer Nature Switzerland.

[BDJR97] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In Proceedings 38th Annual Symposium on Foundations of Computer Science, pages 394–403, 1997.

[BHHO08] Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In David Wagner, editor, Advances in Cryptology – CRYPTO 2008, pages 108–125, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs #1. In Hugo Krawczyk, editor, Advances in Cryptology — CRYPTO '98, pages 1–12, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[BRS02] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. Cryptology ePrint Archive, Paper 2002/100, 2002. `https://eprint.iacr.org/2002/100`.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. volume 2045/2001, pages 93–118, 05 2001.

[CS01] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. Cryptology ePrint Archive, Paper 2001/085, 2001. `https://eprint.iacr.org/2001/085`.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23, pages 523–540. Springer, 2004.

[Dzi06] Stefan Dziembowski. On Forward-Secure Storage. In Cynthia Dwork, editor, Advances in Cryptology - CRYPTO 2006, Lecture Notes in Computer Science, pages 251–270, Berlin, Heidelberg, 2006. Springer.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. volume 9, pages 169–178, 05 2009.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. J. Comput. Syst. Sci., 28:270–299, 1984.

[Gof23] Igol Goffman. The mgm breach and the role of idp in modern cyber attacks. 2023.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. J. ACM, 59(3):11, 2012.

[Gro10] Jens Groth. Short non-interactive zero-knowledge proofs. In Masayuki Abe, editor, Advances in Cryptology - ASIACRYPT 2010, pages 341–358, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[JW16] Zahra Jafargholi and Daniel Wichs. Adaptive security of yao's garbled circuits. In Theory of Cryptography Conference, pages 433–458. Springer, 2016.

[JWZ22] Guan Jiaxin, Daniel Wichs, and Mark Zhandry. Incompressible Cryptography, pages 700–730. 01 2022.

[KKRS24] Venkata Koppula, Abhinav Kumar, Mahesh Sreekumar Rajasree, and Harihar Swaminathan. Incompressible encryption beyond cpa/cca security. Manuscript submitted, 2024.

[KM19a] Fuyuki Kitagawa and Takahiro Matsuda. CPA-to-CCA transformation for KDM security. Cryptology ePrint Archive, Paper 2019/609, 2019. https://eprint.iacr.org/2019/609.

[KM19b] Fuyuki Kitagawa and Takahiro Matsuda. CPA-to-CCA transformation for KDM security. Cryptology ePrint Archive, Paper 2019/609, 2019. https://eprint.iacr.org/2019/609.

[KMT19] Fuyuki Kitagawa, Takahiro Matsuda, and Keisuke Tanaka. Simple and efficient KDM-CCA secure public key encryption. Cryptology ePrint Archive, Paper 2019/1012, 2019. https://eprint.iacr.org/2019/1012.

[KW18] Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. Cryptology ePrint Archive, Paper 2018/847, 2018. https://eprint.iacr.org/2018/847.

[Nis90] N. Nisan. Pseudorandom generators for space-bounded computations. In Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, STOC '90, page 204–212, New York, NY, USA, 1990. Association for Computing Machinery.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. Cryptology ePrint Archive, Paper 2009/105, 2009. https://eprint.iacr.org/2009/105.

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

[Riv97] Ronald L. Rivest. All-or-nothing encryption and the package transform. In Eli Biham, editor, Fast Software Encryption, pages 210–218, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[RS91] Charles Rackoff and Daniel Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. volume 576, pages 433–444, 08 1991.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 21(2):120–126, 1978.

[SGV11] Leo Reyzin Boaz Barak Shafi Goldwasser, Yael Kalai and Salil Vadhan. Extractors and the leftover hash lemma, 2011.

[Wee15] Hoeteck Wee. KDM-security via homomorphic smooth projective hashing. Cryptology ePrint Archive, Paper 2015/721, 2015. `https://eprint.iacr.org/2015/721`.

[Yao86] Andrew Yao. How to generate and exchange secrets. In FOCS, 1986.