

Tight Multi-Target Security for Key Encapsulation Mechanisms

by

Lewis Glabush

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2024

© Lewis Glabush 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The use of symmetric encryption schemes requires that the communicating parties have access to a shared secret key. A key encapsulation mechanism (KEM) is a cryptographic tool for the secure establishment of such a key. The KEMs most commonly used at this time are vulnerable to adversaries with access to a large quantum computer. This project concerns KEMs that are resistant to all known quantum attacks, such as lattice-based schemes.

A desirable property for any KEM is multi-target security, capturing the idea that security does not degrade below the targeted level as the number of users of a protocol or the amount of communication per user scales to a certain threshold. For schemes based on prime-order groups, multi-ciphertext security can be trivially reduced to single-ciphertext security using self reducibility arguments, but these arguments are not available for lattice-based schemes. Indeed, one of the alternates in NIST’s post-quantum cryptography standardization project, **FrodoKEM**, was susceptible to simple attacks in the multi-target setting.

The standard approach to building IND-CCA secure KEMs has been to start with an IND-CPA secure public key encryption scheme (PKE) and apply the Fujisaki-Okamoto transform (FO). In this paper, we introduce a new variant of the FO transform, called the salted FO transform (SFO) which adds a uniformly random salt to the generation of ciphertexts. We then show that the resulting KEM’s have much tighter security bounds compared to their generic counterparts. We then apply our results to **FrodoKEM** to resolve the aforementioned simple attacks.

Acknowledgements

I would like to thank my supervisor Douglas Stebila, who introduced me to the subject of this paper, and provided meticulous mentorship, invaluable insights, and consistent encouragement throughout the process. All novel results presented in this thesis were developed by myself and Douglas, with helpful consultation from Kathrin Hövelmanns. I would also like to thank Ray Perlner of NIST, who identified the attack on **FrodoKEM**, and suggested the salt based countermeasure, which became the motivation for this paper.

I am indebted to my parents, Sky Glabush and Julie Rogers-Glabush. I am eternally grateful to my wife Leili Rosen for her endless support and encouragement.

Table of Contents

Author’s Declaration	ii
Abstract	iii
Acknowledgements	iv
List of Figures	viii
1 Introduction	1
1.1 Motivation	1
1.2 Our Contributions	2
1.3 Related Work	3
2 Preliminaries	4
2.1 Notation and Terminology	4
2.2 Cryptography	6
2.2.1 Public Key Encryption Schemes	6
2.2.2 Security Definitions	8
2.2.3 Key Encapsulation Mechanisms	11
2.3 Reductions	13

3	Lattices and eFrodoKEM	15
3.1	Learning with Errors	15
3.1.1	Lattices	16
3.1.2	Learning with Errors	17
3.2	The FO Transform	18
3.2.1	T-transform : IND/OW-CPA to OW-PCA/PCVA	18
3.2.2	U-transform : OW-PCA to IND-CCA	20
3.3	eFrodoKEM	21
3.3.1	The Lindner-Peikert Cryptosystem	21
3.3.2	Auxiliary Algorithms for Frodo	23
3.3.3	FrodoPKE	27
3.3.4	eFrodoKEM Specifications	28
4	Multi-Target Security	31
4.1	Multi-Target Security Definitions	32
4.1.1	Security Definitions	32
4.1.2	Multi-Target Attack on eFrodoKEM	34
4.2	The FO/SFO Transform in the Multi-Target Setting	35
4.2.1	The T-transform: $\text{IND}_{n,u}\text{-CPA}$ to $\text{OW}_{n,u}\text{-PCVA}$	36
4.2.2	The ST-transform: $\text{IND}_{n,u}\text{-CPA}$ to $\text{OW}_{n,u}\text{-PCVA}$ tightly	42
4.2.3	The $\text{U}^\perp$ -transform: $\text{OW}_{n,u}\text{-PCVA}$ to $\text{IND}_{n,u}\text{-CCA}$ with implicit rejection	49
4.2.4	The $\text{U}^\perp$ -transform: $\text{OW}_{n,u}\text{-PCVA}$ to $\text{IND}_{n,u}\text{-CCA}$ with explicit rejection	64
4.2.5	Public Key Hashing	71
4.2.6	Prefix Hashing	73
4.2.7	Monolithic Security Bounds	73
4.2.8	Tightness for FrodoKEM Parameters	95
4.3	Modelling a Quantum Adversary	96
4.4	FrodoKEM	99
4.4.1	FrodoKEM Specifications	99
4.4.2	Efficiency Analysis	102

5 Conclusion	104
5.1 Comparison to Existing Work	104
5.2 Future Work	105
References	106

List of Figures

2.1	One-way security for a PKE.	9
2.2	IND-CPA security experiment.	10
2.3	Decryption oracle for IND-CCA security.	11
2.4	IND-CCA security experiment for a KEM.	12
3.1	Encryption and decryption for T[PKE].	19
3.2	Key encapsulation mechanism from $\text{FO}_{m,c}^\chi$	21
3.3	Lindner-Peikert cryptosystem.	22
3.4	Encoding function for Frodo.	24
3.5	Decoding function for Frodo.	24
3.6	Matrix packing function for Frodo.	25
3.7	Matrix unpacking function Frodo.	25
3.8	Sampling function for Frodo.	25
3.9	Matrix sampling function for Frodo.	26
3.10	Frodo.Gen() using AES128	26
3.11	Frodo.Gen() with SHAKE128.	26
3.12	Key generation algorithm for FrodoPKE.	27
3.13	Encryption algorithm for FrodoPKE.	28
3.14	Decryption algorithm for FrodoPKE.	28
3.15	Encapsulation for eFrodoKEM.	29
3.16	Decapsulation for eFrodoKEM.	30

4.1	Experiment for $\text{OW}_{n,u}$ -ATK-security.	33
4.2	Experiment for $\text{IND}_{n,u}$ -CCA-security.	34
4.3	Encryption and decryption for $\text{ST}[\text{PKE}]$.	36
4.4	$\mathbf{G}_0$ for the proof of Theorem 4.2.2	38
4.5	$\mathbf{G}_1$ for the proof of Theorem 4.2.2	39
4.6	$\mathbf{G}_2$ for the proof of Theorem 4.2.2	40
4.7	$\mathbf{G}_3$ for the proof of Theorem 4.2.2	41
4.8	Adversaries C and D for the proof of Theorem 4.2.2	42
4.9	$\mathbf{G}_0$ for the proof of Theorem 4.2.3	43
4.10	$\mathbf{G}_1$ for the proof of Theorem 4.2.3	44
4.11	$\mathbf{G}_2$ for the proof of Theorem 4.2.3	45
4.12	$\mathbf{G}_3$ for the proof of Theorem 4.2.3	46
4.13	Adversaries C and D against the $\text{OW}_{n,u}$ -CPA-security of PKE for the proof of Theorem 4.2.3	47
4.14	$\mathbf{G}_0$ for the proof of Theorem 4.2.4	50
4.15	$\mathbf{G}_1$ for the proof of Theorem 4.2.4	51
4.16	$\mathbf{G}_2$ for the proof of Theorem 4.2.4	52
4.17	$\mathbf{G}_3$ for the proof of Theorem 4.2.4	53
4.18	$\mathbf{G}_4$ for the proof of Theorem 4.2.4	55
4.19	An adversary A against the $\text{OW}_{n,u}$ -PCA-security of PKE_1 simulating $\mathbf{G}_4$ for B.	56
4.20	$\mathbf{G}_0$ for the proof of Theorem 4.2.5	58
4.21	$\mathbf{G}_1$ for the proof of Theorem 4.2.5	59
4.22	$\mathbf{G}_2$ for the proof of Theorem 4.2.5	60
4.23	$\mathbf{G}_3$ for the proof of Theorem 4.2.5	61
4.24	$\mathbf{G}_4$ for the proof of Theorem 4.2.5	63
4.25	An adversary A against the $\text{OW}_{n,u}$ -PCA-security of PKE_1 simulating $\mathbf{G}_4$ for B.	64
4.26	$\mathbf{G}_0$ for the proof of Theorem 4.2.6	66
4.27	$\mathbf{G}_1$ for the proof of Theorem 4.2.6	67

4.28	$\mathbf{G}_2$ for the proof of Theorem 4.2.6	68
4.29	$\mathbf{G}_3$ for the proof of Theorem 4.2.6	69
4.30	An adversary $\mathbf{A}$ against the $\text{OW}_{n,u}$ -PCA-security of PKE_1 simulating $\mathbf{G}_3$ for $\mathbf{B}$	70
4.31	The KEM $\text{SFO}_{\text{pk},m,c}^\chi[\text{PKE}, \mathbf{G}, \mathbf{H}, \text{len}_{\text{salt}}]$	71
4.32	$\mathbf{G}_0$ for the proof of Theorem 4.2.9	75
4.33	$\mathbf{G}_1$ for the proof of Theorem 4.2.9	76
4.34	$\mathbf{G}_2$ for the proof of Theorem 4.2.9.	77
4.35	$\mathbf{G}_3$ for the proof of Theorem 4.2.9.	78
4.36	$\mathbf{G}_4$ for the proof of Theorem 4.2.9.	80
4.37	$\mathbf{G}_5$ for the proof of Theorem 4.2.9.	82
4.38	Adversaries $\mathbf{B}_1$ and $\mathbf{B}_2$ for the proof of 4.2.9	83
4.39	$\mathbf{G}_0$ for the proof of Theorem 4.2.10	86
4.40	$\mathbf{G}_1$ for the proof of Theorem 4.2.10	87
4.41	$\mathbf{G}_2$ for the proof of Theorem 4.2.10.	88
4.42	$\mathbf{G}_3$ for the proof of Theorem 4.2.10.	89
4.43	$\mathbf{G}_4$ for the proof of Theorem 4.2.10.	90
4.44	$\mathbf{G}_5$ for the proof of Theorem 4.2.10.	92
4.45	$\mathbf{G}_6$ for the proof of Theorem 4.2.10.	93
4.46	Adversaries $\mathbf{B}_1$ and $\mathbf{B}_2$ for the proof of Theorem 4.2.10.	94
4.47	Security bounds for various versions of the FO transform	97
4.48	Encapsulation for FrodoKEM	100
4.49	Decapsulation for FrodoKEM	101
4.50	Size in bytes of inputs and outputs in each variant of FrodoKEM	102
4.51	Time in microseconds for an iteration of FrodoKEM for various versions, using SHAKE.	103

Chapter 1

Introduction

1.1 Motivation

Cryptography is the study of protocols and techniques for secure communication in the presence of adversaries. Cryptography often conjures the image of code breakers, intercepting cryptic messages and deciphering them to obtain intelligence. Today, cryptography is far more ubiquitous, and is used by nearly everyone, multiple times per day, when they access the internet, communicate using Whatsapp or Messenger, use their debit card, make transactions online, and so on. The security of these aforementioned interactions depends on cryptographic protocols — tools to achieve a particular functionality, preserving the safety of the user, even when they are targeted by malicious entities. To this end, computer scientists, engineers, and mathematicians, have developed rigorous mathematical notions of security, and apply them to prove security for the protocols that are employed by society.

Indistinguishability under chosen-ciphertext attack (IND-CCA) [RS92] is generally accepted as the standard notion of security in asymmetric cryptographic schemes. It entails that an adversary cannot determine, with high probability, which of two messages, selected by the adversary, has been encrypted under a public key. It captures the notion that no *partial* information should be leaked about a plaintext, by its encryption.

We can strengthen the notion of IND-CCA-security by capturing how security degrades with the number of users of a protocol, and the number of ciphertext prepared between all users. If, for instance, a protocol was secure for any given user, but with sufficiently many users, at least one would be compromised, this protocol would likely fall out of favour.

Multi-target IND-CCA security [BBM00] is parameterized by u , the number of public keys (users) and n the total number of challenge ciphertexts. We denote this by $\text{IND}_{n,u}\text{-CCA}$. In this paper, we consider multi-target security in the context of key agreement mechanisms (KEM's). KEM's are one of the most fundamental cryptographic protocols. They allow two parties, communicating over public channels, to establish a shared secret key, which can then be used in symmetric communication. The most widely used KEM's today do satisfy $\text{IND}_{n,u}\text{-CCA}$ security, however, they are not resistant to attacks from adversaries able to make use of large quantum computers.

In 2017, the National Institute of Standards and Technology (NIST), initiated the post-quantum cryptography standardization project [NIS17]. One of the major objectives was to standardize a KEM which was quantum resistant. The approach used in nearly all of the candidates in that competition was to build an IND-CPA-secure PKE, the hardness of which depends on a problem for which there is no known quantum algorithm, and then to apply the Fujisaki-Okamoto transform: which transforms a PKE to a KEM, and guarantees IND-CCA security provided the underlying PKE is IND-CPA secure.

FrodoKEM was one of the prominent candidates in the NIST post quantum cryptography standardization project, selected as a third round alternate. FrodoKEM satisfied the targeted security levels for IND-CCA security. However, with many challenge ciphertexts (large n), FrodoKEM exhibited security degradation, and was susceptible to an attack that only involved sampling sufficiently many ciphertexts, and looking for a collision. FrodoKEM is obtained by applying a variant of the FO to a PKE which was $\text{IND}_{n,u}\text{-CPA}$, assuming the hardness of Learning with Errors. As such, the vulnerability of FrodoKEM illustrated a consideration in constructing KEM's from the FO: under certain parameters a PKE being $\text{IND}_{n,u}\text{-CPA}$ does not guarantee that the KEM obtained by applying the FO to that PKE is $\text{IND}_{n,u}\text{-CCA}$.

1.2 Our Contributions

Chapters 2 of this thesis covers basic cryptographic background, which should be familiar to readers with some background in cryptography. These include notions of security, definitions of protocols, proof techniques, and an index of notation. Chapter 3 covers more specific background such as learning with errors, the Fujisaki Okamoto transform, and FrodoKEM. The definitions and theorems presented in these two chapters are drawn entirely from the literature.

The major results of this thesis are presented in chapter 4. There, we introduce the salted Fujisaki-Okamoto transform (SFO), a slightly modified version of the FO-transform,

which mitigates the security degradation from having large n , i.e., many challenge ciphertexts. Furthermore, we prove the security of the KEMs resulting from applying the **SFO** transform to $\text{IND}_{n,u}\text{-CPA}$ secure PKE's in the random oracle model. We apply these results to **FrodoKEM** to prove that the most recent version, which makes use of the **SFO**-transform, is $\text{IND}_{n,u}\text{-CCA}$ secure, and show experimentally that the overhead increase is negligible and easily mitigated. We briefly describe how one could obtain security bounds in the quantum random oracle model as well.

1.3 Related Work

This paper is a continuation [HHK17] by Hövelmanns, et al. That paper introduces modular versions of the **FO** transform, and proves security bounds for single-target security, building on [Den03, FO99]. This paper has been cited by nearly all of the NIST post quantum cryptography KEM proposals. Our paper builds on this by considering security in the multi-target setting, and by introducing salting to improve the tightness gap.

Another paper that considers multi-target security in the context of KEMs is [DHK⁺21] by Duman et. al. This paper considered the generic **FO** transform in the multi-target setting, as well as introduces, and characterizes a security-efficiency trade-off, called prefix hashing.

The central object of study for this thesis is **FrodoKEM**, an IND-CCA secure KEM that makes use of a variant of the **FO** transform, per [HHK17]. This thesis is therefore relating to [ABD⁺23b, ABD⁺23a, BCD⁺16, LP11, MR04, LPR10].

Chapter 2

Preliminaries

This chapter covers basic cryptographic definitions, notions of correctness and security, an overview of the random oracle model, and an overview of the general proof strategy used in later chapters.

2.1 Notation and Terminology

For convenience, we include a table of commonly used notation.

Table 2.1: Some generic symbols used throughout this thesis.

Symbol	Description	Defined
$m, \mathcal{M}$	A private message/plaintext, m belonging to a message space $\mathcal{M}$.	2.2.1
$c, \mathcal{C}$	A public ciphertext c belonging to a ciphertext space $\mathcal{C}$.	2.2.1
$r, \mathcal{R}$	A random bitstring r belonging to a randomness space $\mathcal{R}$.	2.2.1
$pk, \mathcal{PK}$	A public key pk belonging to a public key space $\mathcal{PK}$.	2.2.1
$sk, \mathcal{SK}$	A secret key sk belonging to a secret key space $\mathcal{SK}$.	2.2.1
$ss, \mathcal{K}$	A shared secret ss , belonging to a space $\mathcal{K}$.	2.2.1
$\text{salt}, \text{len}_{\text{salt}}$	A public uniformly random bitstring of length len_{salt} .	4.2

Symbol	Description	Defined
$\leftarrow$	Denotes sampling from a set, assignment of a variable, or running a deterministic algorithm.	-
$\overset{\$}{\leftarrow}$	Denotes sampling from a set uniformly at random, or running a probabilistic algorithm.	-
$\llbracket \text{statement} \rrbracket$	Returns 1 if statment is true, and 0 otherwise.	-
$\text{KeyGen}()$	A probabilistic key generation algorithm that outputs a public key pk and a secret key sk .	2.2.1
$\text{Enc}(pk, m)$	An encryption function, which takes as input a public key pk and message m , and outputs a public ciphertext.	2.2.1
$\text{Enc}(pk, m; r)$	A deterministic encryption function where randomness is an input, together with the public key pk and message m , and outputs a public ciphertext c .	2.2.1
$\text{Dec}(sk, c)$	A decryption function, which takes as input a secret key sk , and a ciphertext c and outputs a message m	2.2.1
$\text{Encaps}(pk)$	An encapsulation function, which takes as input a public key and outputs secret key ss and a public ciphertext c .	2.2.1
$\mathcal{O}_{\text{Dec}}(c)$	A decryption oracle, which takes a ciphertext c , and outputs a message $m \leftarrow \text{Dec}(sk, c)$.	2.2.2
$\mathcal{O}_{\text{Decaps}}(c)$	a decapsulation oracle, which takes as input a ciphertext and outputs a shared secret $ss := \text{Decaps}(sk, c)$.	2.2.2
$\text{Decaps}(sk, c)$	a decapsulation function, which takes as input a ciphertext and a secret key, and outputs a shared secret ss .	2.2.1
$\text{Decaps}(j, c)$	a decapsulation oracle, which takes as input a ciphertext and outputs a shared secret $ss := \text{Decaps}(sk_j, c)$, in a multi-target setting.	4.3
$\text{encode}(\mathbf{k})$	A function for representing bit strings as matrices.	3.3.1
$\text{decode}(\mathbf{K})$	The inverse of the encoding function, to convert matrices into bit strings.	3.3.1
$\text{pack}(\mathbf{C})$	A function for packing matrices into bit strings.	3.3.1
$\text{unpack}(\mathbf{b})$	The inverse of the packing function to convert bit strings into matrices.	3.3.1
$\text{sample}(\mathbf{r})$	An algorithm for sampling random error terms.	3.3.1
$\text{SampleMatrix}(\mathbf{r})$	An algorithm for sampling matrices of random error terms.	3.3.1

Table 2.2: More generic symbols and functions used throughout this thesis.

Symbol	Description	Defined
$\mathfrak{L}_M$	The private set of messages encrypted as challenge ciphertexts to an arbitrary public key in a single target setting.	4.2
$\mathfrak{L}_{M_j}$	The private set of messages encrypted as challenge ciphertexts for user j in a multi-target setting.	4.3
$\mathfrak{L}_C$	The public set of challenge ciphertexts with produced under an arbitrary public key, in a single target setting.	4.2
$\mathfrak{L}_{C_j}$	The public set of challenge ciphertexts with for user j in a multi-target setting.	4.3
$\mathfrak{L}_G, \mathfrak{L}_H$	Lists of (Q)ROM queries by an adversary in a single-target setting.	4.2
$\mathfrak{L}_{G_j}, \mathfrak{L}_{H_j}$	Lists of (Q)ROM queries by an adversary corresponding to a specific user in a multi-target setting.	4.3
$\mathfrak{L}_D$	The set of decapsulation oracle queries performed by an adversary in a single-target setting..	4.2
$\mathfrak{L}_{D_j}$	The set of decapsulation oracle queries performed by an adversary corresponding to a specific user in a multi-target setting.	4.3

Table 2.3: Lists used throughout this thesis.

2.2 Cryptography

Cryptography is the science concerning techniques and protocols for communicating in the presence of adversaries. There are two broad areas of cryptographic, symmetric and asymmetric (also public key) cryptography. The former assumes that the two communicating parties have access to a shared secret key, while the latter assumes no prior secret communication.

2.2.1 Public Key Encryption Schemes

A public key encryption scheme (PKE) is used when two parties with no prior secret communication wish to communicate over an insecure channel. Formally, we define a PKE as follows

Definition 2.2.1 (Public Key Encryption Scheme). A public key encryption scheme is a triple of algorithms $\text{PKE} = (\text{KeyGen}, \text{Enc}, \text{Dec})$, together with a message space $\mathcal{M}$:

- $\text{KeyGen}() \rightarrow (pk, sk)$: A probabilistic key generation algorithm that outputs a public key pk and a private key sk .
- $\text{Enc}(pk, m) \rightarrow c$: A probabilistic algorithm that takes as input a public key pk and a message $m \in \mathcal{M}$ and outputs a ciphertext c . Encryption makes use of random values, belonging to an *randomness space* $\mathcal{R}$. In some cases, the randomness used in an encryption will be treated as an explicit parameter so that, $c \leftarrow \text{Enc}(pk, m; r)$, in which case encryption is deterministic.
- $\text{Dec}(sk, c) \rightarrow m'$ or $\perp$: A deterministic algorithm that takes as input a ciphertext c and a private key sk and produces either a message $m' \in \mathcal{M}$ or an error symbol $\perp \notin \mathcal{M}$.

The properties of a PKE that are relevant to this project are *correctness* and *security*.

Certain public key encryption schemes, for example those based on lattice problems, exhibit correctness errors. These occur when, for a message $m \in \mathcal{M}$, and key pair (pk, sk) first encrypting, then decryption, does not return m . Correctness errors may leak some information, as shown in [Section 4.2.1](#). In [\[HHK17\]](#) Hofheinz, Hövelmanns, and Kiltz, came up with a notion of correctness sufficient to prove the security of modular FO transforms.

Definition 2.2.2 (δ -correctness of a PKE). A public key encryption scheme $\text{PKE} = (\text{KeyGen}, \text{Enc}, \text{Dec})$ with message space $\mathcal{M}$ is called δ -correct if

$$\mathbb{E}_{(pk, sk)} \left[\max_{m \in \mathcal{M}} \Pr[\text{Dec}(sk, c) \neq m | c \leftarrow \text{Enc}(pk, m)] \right] \leq \delta$$

where the expectation is taken over all $(pk, sk) \xleftarrow{\$} \text{KeyGen}()$

In settings where there are multiple users, we need a definition of multi-user correctness. We denote this as $\delta(u)$ which is the maximum δ , taken over all u users.

Definition 2.2.3 ($\delta(u)$ -correctness of a PKE). A public key encryption scheme $\text{PKE} = (\text{KeyGen}, \text{Enc}, \text{Dec})$ with message space $\mathcal{M}$ is called $\delta(u)$ -correct if

$$\mathbb{E} \left[\max_{j \in [u]} \max_{m \in \mathcal{M}} \Pr[\text{Dec}(sk_j, c) \neq m | c \leftarrow \text{Enc}(pk_j, m)] \right] \leq \delta(u)$$

where the expectation is taken over all $(pk_1, sk_1), \dots, (pk_u, sk_u) \xleftarrow{\$} \text{KeyGen}()$.

Another notion that comes up in security is the entropy of ciphertexts generated by a PKE. The idea is that for a PKE with higher entropy, there will be a more even distribution of ciphertexts, and the probability that a message encrypts to any particular ciphertext is low. Note that this notion is relevant because it is not necessarily the case that the output distribution of an encryption function will be uniform. We define entropy formally below.

Definition 2.2.4 (Min-entropy). For $(pk, sk) \leftarrow \text{KeyGen}$ and $m \in \mathcal{M}$, the min entropy of $\text{Enc}(pk, m)$ is defined as

$$\gamma(pk, m) := -\log \max_{c \in \mathcal{C}} \Pr_{r \in \mathcal{R}} [\text{Enc}(pk, m; r) = c]$$

Definition 2.2.5 (γ -spread). A PKE is γ -spread if for every key pair $(pk, sk) \leftarrow \text{KeyGen}()$ and every message $m \in \mathcal{M}$, $\gamma(pk, m) \geq \gamma$. Equivalently, we have that for any ciphertext c

$$\Pr_{r \in \mathcal{R}} [\text{Enc}(pk, m; r) = c] \leq 2^{-\gamma}$$

2.2.2 Security Definitions

There are a number of different notions for the security of a PKE. The standard in cryptographic literature is to express security notions in terms of an experiment being attempted by an adversary. The different notions relate to the objective of and powers afforded to the hypothetical adversary $\mathcal{A}$. The objective may be to break *one-way* (OW) security, meaning to recover a plaintext from its encryption. It may be to break *indistinguishability* (IND), meaning either to determine which of two messages has been encrypted under a given public key, or to distinguish a ciphertext from a random vector, depending on the context.

Definition 2.2.6 (OW-ATK Security for a PKE). Consider the game shown in Figure 2.3. where

$$\mathcal{O}_{\text{ATK}} := \begin{cases} - & \text{if ATK = CPA} \\ \text{PCO} & \text{if ATK = PCA} \\ \text{PCO, CPA} & \text{if ATK = PCVA} \end{cases}$$

Denote the advantage of an adversary $\mathcal{A}$ in the OW-ATK game as

$$\text{Adv}_{\text{PKE}}^{\text{OW-ATK}}(\mathcal{A}) := \Pr[\text{Exp}_{\text{PKE}}^{\text{OW-ATK}}(\mathcal{A}) \Rightarrow 1]$$

$\text{Exp}_{\text{PKE}}^{\text{OW-ATK}}(\mathbf{A})$	$\text{PCO}(m \in \mathcal{M}, c)$
1 : $(pk, sk) \xleftarrow{\$} \text{KeyGen}()$	6 : return $\llbracket \text{Dec}(sk, c) = m \rrbracket$
2 : $m^* \xleftarrow{\$} \mathcal{M}$	
3 : $c^* \leftarrow \text{Enc}(pk, m^*)$	$\text{CVO}(c \neq c^*)$
4 : $m' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ATK}}}(pk, c^*)$	7 : $m \leftarrow \text{Dec}(sk, c)$
5 : return $\llbracket m^* = m' \rrbracket$	8 : return $\llbracket m \in \mathcal{M} \rrbracket$

Figure 2.1: One-way security for a PKE.

A public key encryption scheme **PKE** is *one-way* against an attack $\text{ATK} \in \{\text{CPA}, \text{PCA}, \text{PCVA}\}$ if for any adversary **A** running an algorithm with time T and success probability epsilon, we have

$$\frac{T}{\epsilon} \geq 2^\kappa$$

where κ is the desired security level.

The above definition captures the idea that given a ciphertext/public key pair, and access to the relevant oracles, no adversary can produce the underlying plaintext. Here **OW** relates to the objective of a adversary: to find the underlying plaintext given a ciphertext. **ATK** describes the powers afforded to the adversary. Chosen plaintext attacks (**CPA**) only allow the adversary to encrypt messages using the public key. Plaintext checking attacks (**PCA**) allow the adversary to make use of a plaintext checking oracle (**PCO**). Plaintext checking and verifying attacks (**PCVA**) allow the adversary to make use of **PCO** and a ciphertext validity oracle **CVO**.

One-way security may seem like the natural notion of security to target for a PKE. However, an adversary might not be able to produce the plaintext underlying a particular ciphertext, but might be able to learn *partial* information. A strong notion of security should capture the idea that an adversary cannot leverage any information leaked by a ciphertext. Hence *ciphertext indistinguishability*, defined in figure, is considered the standard notion.

Definition 2.2.7 (IND-CPA Security of a PKE). Consider the experiment shown in Figure 2.2

$\text{Exp}_{\text{PKE}}^{\text{IND-CPA}}(\mathbf{A})$	
1 :	$(pk, sk) \xleftarrow{\$} \text{KeyGen}()$
2 :	$(m_1^*, m_2^*, st) \leftarrow \mathbf{A}(pk)$
3 :	$b \xleftarrow{\$} \{0, 1\}$
4 :	$c^* \leftarrow \text{Enc}(pk, m_b)$
5 :	$b' \leftarrow \mathbf{A}(pk, c^*, st)$
6 :	return $\llbracket b = b' \rrbracket$

Figure 2.2: IND-CPA security experiment.

Denote the advantage of an adversary $\mathbf{A}$ in the IND game as

$$\text{Adv}_{\text{PKE}}^{\text{IND-CPA}}(\mathbf{A}) := \left| \Pr[\text{Exp}_{\text{PKE}}^{\text{IND-CPA}}(\mathbf{A}) \Rightarrow 1] - \frac{1}{2} \right|$$

We say that a PKE is *indistinguishable* against an attack $\text{ATK} \in \{\text{CPA}, \text{CCA}\}$ if, for any adversary $\mathbf{A}$ running an algorithm with time T and success probability epsilon, we have

$$\frac{T}{\epsilon} \geq 2^\kappa$$

where κ is the desired security level.

Note that if a ciphertext leaked any information, even a single bit, of the plaintext, than this would allow an adversary to trivially win the IND-CCA game, by choosing two plaintexts which differ at every bit, in which case the leaked bit would reveal which plaintext had been encrypted.

The more standard, albeit difficult to achieve security notion for PKE's is *indistinguishability under chosen ciphertext attack* (IND-CCA). The definition of IND-CCA security is identical to that of IND-CPA security, except that the adversary is given access to a decryption oracle which they can query at any time.¹

¹There are two notions for IND-CCA security: non-adaptive and adaptive. The notion described above is *indistinguishability under adaptive chosen ciphertext attack* or IND-CCA2. The weaker notion would require that the adversary choose all ciphertexts to submit to the decryption oracle at the outset, rather than being able to make repeated queries. Adaptive is usually preferred because it is a stronger security notion, and reflects real life security risks from malicious adversaries.

$\mathcal{O}_{\text{Dec}}(c \neq c^*)$
$7 : m' \leftarrow \text{Dec}(sk, c)$
$8 : \text{return } m'$

Figure 2.3: Decryption oracle for IND-CCA security.

2.2.3 Key Encapsulation Mechanisms

Symmetric encryption schemes differ from PKE's, in that the communicating parties only make use of one shared secret key. In many settings, it is prohibitively computationally costly to perform all communications using a PKE. Thus a common approach to secure communication is *hybrid encryption* where public key cryptography is used for the establishment of a shared secret sk , which is then used in a symmetric encryption scheme. A key encapsulation mechanism (KEM) is a triple of algorithms used to establish a shared secret key between two parties. A standard way to build a KEM is to start with a passively secure PKE, and then apply the [Fujisaki-Okamoto transform](#), which is described in Chapter 3. A KEM is defined as follows

Definition 2.2.8 (Key Encapsulation Mechanism). A key encapsulation mechanism is a triple of algorithms $\text{KEM} = (\text{KeyGen}, \text{Encaps}, \text{Decaps})$, together with a finite key space $\mathcal{K}$:

- $\text{KeyGen}() \rightarrow (pk, sk)$: A probabilistic key generation algorithm that outputs a public key pk and a private key sk .
- $\text{Encaps}(pk) \rightarrow (c, ss)$: A probabilistic algorithm that takes as input a public key pk , and outputs an encapsulation c and a shared secret $ss \in \mathcal{K}$.
- $\text{Decaps}(sk, c) \rightarrow ss' \text{ or } \perp$: A deterministic algorithm which takes as input a secret key sk and an encapsulation c and produces a shared secret $ss' \in \mathcal{K}$ or an error symbol $\perp \notin \mathcal{K}$.

Note that it is possible to design KEMs for which decapsulation only ever returns an output $ss \in \mathcal{K}$. These KEMs have a property called *implicit rejection*. For these KEMs, if an encapsulation is rejected, the output of **Decaps** is something resembling a valid shared secret. A KEM for which **Decaps** outputs $\perp$ when an encapsulation is rejected has the property of *explicit rejection*. A formal description of these properties can be found in Section 3.1.2.

We can define *security* and *correctness* for KEMs similar to PKEs.

$\text{Exp}_{\text{KEM}}^{\text{IND-CCA}}(\mathbf{A})$	$\mathcal{O}_{\text{Decaps}}(c \neq c^*)$
1 : $(pk, sk) \xleftarrow{\$} \text{KeyGen}()$	7 : return $\text{Decaps}(sk, c)$
2 : $b \xleftarrow{\$} \{0, 1\}$	
3 : $(c^*, K_0) \xleftarrow{\$} \text{Encaps}(pk)$	
4 : $K_1 \xleftarrow{\$} U(\mathcal{K})$	
5 : $b' \leftarrow \mathbf{A}^{\mathcal{O}_{\text{Decaps}}}(pk, c^*, K_b)$	
6 : return $\llbracket b = b' \rrbracket$	

Figure 2.4: IND-CCA security experiment for a KEM.

Definition 2.2.9 (δ -correctness of a KEM). A key encapsulation mechanism $\text{KEM} = (\text{Encaps}, \text{Decaps}, \text{KeyGen})$ with key space $\mathcal{K}$ is called δ -correct if

$$\mathbb{E}_{m \in \mathcal{M}} [\max \Pr[\text{Decaps}(sk, c) \neq \text{ss} \mid (pk, sk) \xleftarrow{\$} \text{KeyGen}(); (c, K) \xleftarrow{\$} \text{Encaps}(pk)]] \leq \delta$$

where the expectation is taken over $(pk, sk) \leftarrow \text{KeyGen}$.

This definition captures the idea that a KEM should produce a shared secret between two parties, so that when $(c, K) \leftarrow \text{Encaps}(pk)$, then $\text{Decaps}(sk, c) \rightarrow K$, and K becomes the shared secret.

For KEM's the standard security notion is IND-CCA, defined below. Note that a notion of IND-CPA for a KEM can be obtained by disallowing access to a decapsulation oracle. Note that we do not present a notion of one-way security for a KEM, as this will not be relevant to our work.

Definition 2.2.10 (IND-CCA Security of a KEM). Consider the experiment shown in figure 2.4

Denote the advantage of an adversary $\mathbf{A}$ in the IND-CCA game as

$$\text{Adv}_{\text{KEM}}^{\text{IND-CCA}}(\mathbf{A}) := \left| \Pr[\text{Exp}_{\text{KEM}}^{\text{IND-CCA}}(\mathbf{A}) \Rightarrow 1] - \frac{1}{2} \right|$$

We say that a KEM is *indistinguishable under chosen-ciphertext attack*² if no adversary can win the above experiment with non-negligible probability.

²Removing access to $\mathcal{O}_{\text{Decaps}}()$ would define *indistinguishability under chosen plaintext attack* for a KEM

2.3 Reductions

In order to prove security for a modern public key cryptographic protocol one must always assume the hardness of some underlying conjectured-hard problem.³ A conjectured-hard problem is one for which there is no *known* algorithm which solves that problem in polynomial time, and it is considered reasonable to assume that no such algorithm will be found. For example: factoring a product of large primes has been assumed to be a hard problem for decades; no efficient classical algorithm has been found. However, prime factorization could be efficiently solved by quantum computers.. This section will provide definitions relevant to the security proofs used throughout this paper. Some familiarity with complexity classes is assumed.

Definition 2.3.1 (Polynomial-time reducibility of a protocol to a problem). Let $\mathcal{P}$ be a protocol, and $\mathcal{S}$ be a problem. Let $\mathbf{A}$ be an oracle that achieves the adversarial goal $\mathcal{P}$ characterizing the security of $\mathcal{P}$.

We say that the security of $\mathcal{P}$ reduces in polynomial time to the difficulty of $\mathcal{S}$ if given access to $\mathbf{A}$, one can design a polynomial-time algorithm $\mathbf{B}$, that utilizes $\mathbf{A}$ as a subroutine, that solves $\mathcal{S}$. This can be expressed as

$$\mathcal{S} \leq_P \mathcal{P}$$

If $\mathcal{S}$ is a conjectured-hard problem, then such a reduction would constitute a proof of security for $\mathcal{P}$. Examples of conjectured hard-problems that cryptographic protocols have been reduced to include, among many others, prime factorization, the discrete logarithm problem, and [learning with errors](#).

Definition 2.3.2 (Tight Reduction). Let $\mathcal{P}$ be a protocol, and $\mathbf{A}$ be an algorithm with an adversarial goal specified by the security definition for $\mathcal{P}$ with run time at most T and advantage at least ϵ . Suppose $\mathcal{S}$ is a problem, and $\mathbf{B}$ is an algorithm that solves $\mathcal{P}$ running $\mathbf{A}$ as a subroutine, with run time at most T' and advantage at least ϵ' . A reduction from $\mathcal{P}$ to $\mathcal{S}$ is said to be *tight* if

$$\left(\frac{T'}{\epsilon'}\right) \bigg/ \left(\frac{T}{\epsilon}\right) = \mathcal{O}(1)$$

where the value $\left(\frac{T'}{\epsilon'}\right) \bigg/ \left(\frac{T}{\epsilon}\right)$ is called the *tightness gap* of the reduction.

³This assertion does not apply to quantum key distribution.

Intuitively, large values of $\left(\frac{T}{\epsilon}\right)$ imply that the algorithm which solves the adversarial goal specified by the security definition of $\mathcal{P}$ either takes infeasible long, or has negligible probability of success. A security proof involving a reduction to a conjectured hard problem should be tight, since this demonstrates that breaking $\mathcal{P}$ is at least as hard as solving $\mathcal{S}$, whereas a non-tight reduction does not.

Definition 2.3.3 (Security level). Let $\mathcal{P}$ be a protocol with security parameter λ . We say that $\mathcal{P}$ has κ bits of security if any algorithm A with an adversarial goal specified by the security definition of $\mathcal{P}$, and has advantage *at most* ϵ and run time *at least* T satisfies the bound

$$\left(\frac{T}{\epsilon}\right) \geq 2^\kappa.$$

Chapter 3

Lattices and eFrodoKEM

In this chapter, we describe eFrodoKEM¹, a lattice-based KEM, that was selected as a third round alternate in the NIST post quantum cryptography standardization project [ABD⁺23b]. To this end, we also introduce the learning with errors problem (LWE), an encryption enabling problem, on which eFrodoKEM’s security depends, as well as the necessary lattice background.

3.1 Learning with Errors

Learning with errors is a problem introduced by Regev [Reg05, LPR10] which has turned out to be a very robust foundation for cryptographic protocols. In terms of hardness, the best known algorithms for LWE, whether quantum or classical, run in exponential time [MR04]. Furthermore there is a reduction between the hardness of LWE and the *worst-case hardness* of well-studied lattice problems [MP13].

In terms of functionality, LWE has been used most notably as the basis for post-quantum KEMs including eFrodoKEM/FrodoKEM, NewHope, and Kyber, and has been the only problem so far discovered which enabled fully homomorphic encryption. We present here some basic lattice definitions, as well as LWE and its variants [Pei16].

¹eFrodoKEM was simply called FrodoKEM in the NIST submission. The “e” stands for ephemeral, to distinguish an outdated version, which only establishes single-target security, from the updated version, presented in Chapter 4, which establishes multi-target security.

3.1.1 Lattices

In this subsection we describe some preliminary results and problems pertaining to lattices following [Pei16].

Definition 3.1.1 (Lattice ($\mathcal{L}$)). An n -dimensional lattice $\mathcal{L}$ is a subset of $\mathbb{R}^n$ that is both

1. An *additive subgroup*: $0 \in \mathcal{L}$ and $-x, x + y \in \mathcal{L}$ for each $x, y \in \mathcal{L}$.
2. *Discrete*: every $x \in \mathcal{L}$ has a neighborhood in $\mathbb{R}^n$ in which it is the only lattice point.

In this project vectors will be expressed in bold, so that $\mathbf{v} \in \mathcal{L}$ denotes a lattice point.

A lattice, while infinite, is always generated by a finite basis $\mathbb{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_k\}$. We say

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) := \left\{ \sum_{i=1}^k z_i \mathbf{b}_i : z_i \in \mathbb{Z} \right\}$$

There are many related conjectured-hard problems on lattices, that have proved to be highly practical for cryptographic applications. These problems, related constructions, and hardness results, will be summarized in this section.

Definition 3.1.2 (Minimum distance of a lattice). For a lattice $\mathcal{L}$ we define the *minimum distance* $\lambda_1(\mathcal{L})$, as

$$\lambda(\mathcal{L}) := \min_{\mathbf{v} \in \mathcal{L} \setminus \{0\}} \|\mathbf{v}\|$$

for some norm $\|\cdot\|$.

More generally, define $\lambda_i(\mathcal{L})$ to be the smallest r so that $\mathcal{L}$ has at most i linearly independent vectors of length at most r .

Definition 3.1.3 (Shortest vector problem (SVP)). Given a basis $\mathbf{B}$ of an n -dimensional lattice $\mathcal{L}$ find $\mathbf{v} \in \mathcal{L}$ with $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$

Definition 3.1.4 (Approximate shortest vector decision problem(GapSVP_γ)). Given a basis $\mathbf{B}$ of an n -dimensional lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ where $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) \geq \gamma(n)$ determine which is the case.

GapSVP_γ would be trivial if $\lambda_1(n)$ is known, and so it is clear that knowing the minimum distance of a lattice is an unreasonable assumption. Hence, the shortest vector problem is not trivially in NP, because a solution cannot be verified without knowing the minimum distance.

3.1.2 Learning with Errors

Now we describe LWE, which at the outset, does not appear to be a lattice problem directly, but which is linked to lattices through certain reductions to lattice problems. Intuitively, the LWE assumption states that distinguishing between linear equations with an error term, and truly random vectors, is difficult. Formal definitions for LWE assumptions are provided below.

Definition 3.1.5 (LWE sample). Let $\mathbf{s} \in \mathbb{Z}_q$ be a secret random vector. Let χ be an error distribution. Then $A_{\mathbf{s}, \chi}$ is an LWE distribution, sampled as $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $e \leftarrow \chi$ and returning $(\mathbf{a}, b := \mathbf{a}\mathbf{s} + e \bmod q)$.

Definition 3.1.6 (Decisional LWE). Given m samples which are either LWE samples of the form $v_i = \mathbf{a}\mathbf{s} + e_i$, or uniformly random vectors, determine which is the case.

Definition 3.1.7 (Search LWE). Given m LWE samples $v_1, \dots, v_m$ of the form $v_i = \mathbf{a}\mathbf{s} + e_i$, determine $\mathbf{s}$.

One of the essential hardness results relating to LWE is the reduction to the worst-case hardness of GapSVP

$$\text{LWE} \leq \text{GapSVP}_{\text{poly}(n)}$$

This result can be obtained the reductions in [MR04], [SSTX09]. An overview of the relations between lattice problems can be found in [LvdPdW12].

Remark. The hardness of LWE can be reduced to certain problems on *unstructured* lattices. There are variants of LWE which reduce to equivalent problems on lattices that have structure. This creates a security - efficiency trade-off. These variants are Ring-LWE and

Module-LWE, the latter of which is the basis for Kyber, the winner of the NIST post-quantum cryptography standardization project [BDK⁺18]. A detailed treatment of LWE variants is outside of the scope of this thesis, except in noting that **eFrodoKEM** had more conservative security assumptions than Kyber, and yet the former was not multi-target secure, while the latter was. This surprising result does not relate in any way to the hardness of LWE, but rather particular parameters of **eFrodoKEM**, especially the size of the message space. This subject is addressed in [chapter 4](#).

3.2 The FO Transform

To achieve an IND-CCA secure KEM, it suffices to start with the much easier task of constructing a PKE with weaker assumptions, and then apply a transform which maps that PKE into an IND-CCA secure KEM. Such a KEM can be used to establish a shared secret key between the communicating parties, who can then make use of symmetric encryption schemes, yielding a hybrid scheme that is IND-CCA, under certain standard assumptions on the symmetric encryption scheme.

The Fujisaki-Okamoto transform takes as input a passively secure PKE, with either OW-CPA or IND-CPA security, and outputs an IND-CCA KEM. The FO transform works by making it so that in order for an adversary to prepare a ciphertext, they would have to know the underlying plaintext, weakening adversarial advantage derived from access to a decapsulation oracle. In Section 3 of [HHK17] Hofheinz et al. establish four different versions of the FO transform, which vary in the assumptions placed on the PKE, as well as their use of either implicit or explicit rejection, building off of [FO99, Den03].

The FO transform is the composition of two components, the T transform, and the $\mathsf{U} \in \{\mathsf{U}^\perp, \mathsf{U}_m^\perp, \mathsf{U}^\times, \mathsf{U}_m^\times\}$. The resulting compositions are likewise called $\{\mathsf{FO}^\perp, \mathsf{FO}_m^\perp, \mathsf{FO}^\times, \mathsf{FO}_m^\times\}$.

3.2.1 T-transform : IND/OW-CPA to OW-PCA/PCVA

Let $\mathsf{PKE} = (\mathsf{Enc}, \mathsf{Dec}, \mathsf{KeyGen})$ be a generic passively secure public key encryption scheme. The T transform takes two arguments, PKE , and a hash function G , modelled as a random oracle. $\mathsf{T}[\mathsf{PKE}, \mathsf{G}] = \mathsf{PKE}_1 = (\mathsf{KeyGen}, \mathsf{Enc}_1, \mathsf{Dec}_1)$ is a new public key encryption scheme with Enc_1 and Dec_1 described in Figure 3.1.

$\text{Enc}_1(pk, m)$	$\text{Dec}_1(sk, c)$
1 : $c \leftarrow \text{Enc}(pk, m; G(m))$	3 : if $m' = \perp$ or $\text{Enc}(pk, m', G(m')) \neq c$
2 : return	4 : return $\perp$
	5 : else return m'

Figure 3.1: Encryption and decryption for $\text{T}[\text{PKE}]$.

Recall that encryption functions make use of an often unwritten random number r , i.e. $c = \text{Enc}(pk, m) = \text{Enc}(pk, m; r)$ where r is called the “random coins” used to generate c . PKE_1 does not use random coins but instead uses $G(m)$. Note that since G is modelled as a random oracle, $G(m)$ is indistinguishable from a random value. This transformation does, however, make PKE_1 deterministic. This means that, regardless of the security properties of PKE , $\text{T}[\text{PKE}, G]$ is not IND-CPA.

Observe that $\text{Dec}_1(sk, c)$ returns $\perp$ whenever $c \neq \text{Enc}(pk, m'; G(m'))$ for some m' . Consider an adversary with access to a decryption oracle Dec with $\text{Dec}(c) = \text{Dec}_1(sk, c)$. If c were sampled at random, then the probability that c decrypts to something other than $\perp$ is negligible. Informally, this means that in order to produce a ciphertext that decrypts to a message in the message space, the adversary would have to know the underlying plaintext in advance. This idea will be used in Section 3.3. The following theorems taken from [HHK17] describe the security properties of PKE_1 .

Theorem 3.2.1 ($\text{PKE OW-CPA} \xrightarrow{\text{ROM}} \text{T}[\text{PKE}, G] \text{ OW-PCVA}$). *Assume PKE is δ -correct and γ -spread. For any OW-PCVA adversary B issuing at most q_G queries to the random oracle G , q_P queries to a plaintext checking oracle PCO , and q_V queries to a ciphertext validity oracle CVO , there exists an IND-CPA adversary A such that*

$$\text{Adv}_{\text{PKE}_1}^{\text{OW-PCVA}}(B) \leq (q_G + q_P) \cdot \delta + q_V \cdot 2^{-\gamma} + (q_G + q_P + 1) \cdot \text{Adv}_{\text{PKE}}^{\text{OW-CPA}}(A)$$

Theorem 3.2.2 ($\text{PKE IND-CPA} \xrightarrow{\text{ROM}} \text{T}[\text{PKE}, G] \text{ OW-PCVA}$). *Assume PKE is δ -correct and γ -spread. For any OW-PCVA adversary B issuing at most q_G queries to the random oracle G , q_P queries to a plaintext checking oracle PCO , and q_V queries to a ciphertext validity oracle CVO , there exists an IND-CPA adversary A such that*

$$\text{Adv}_{\text{PKE}_1}^{\text{OW-PCVA}}(B) \leq (q_G + q_P) \cdot \delta + q_V \cdot 2^{-\gamma} + \frac{2q_G + 1}{\mathcal{M}} + 3 \cdot \text{Adv}_{\text{PKE}}^{\text{OW-CPA}}(A)$$

3.2.2 U-transform : OW-PCA to IND-CCA

As described in chapter 2.2, one-way security fails to capture the idea that no information about a plaintext should be leaked by a ciphertext. The notions of OW-PCA/PCVA are used as *intermediate* security notions. This section describes the several variations of the U transformation, which maps an OW-secure PKE to an IND-CCA KEM. Each version of the U transform maps a PKE to a KEM, but they vary in two ways:

1. Rejection of invalid ciphertexts:

- $U^\perp$ denotes the use of *explicit rejection* in handling invalid ciphertexts. This means that when $\text{Decaps}^\perp$ is queried on (sk, c) , and $\text{Dec}(sk, c) = \perp$, then $\text{Decaps}^\perp(sk, c) = \perp$
- $U^\times$ denotes the use of *implicit rejection* in handling invalid ciphertexts. This means that when a $\text{Decaps}^\times$ is queried on (sk, c) , and $\text{Dec}(sk, c) = \perp$, then a pseudorandom output is generated using a previously selected uniformly random s is used in the generation of the output of $\text{Decaps}^\times(sk, c)$.

2. Values passed to H:

- $U_{m,c}$ denotes that the output of Encaps and Decaps will be of the form $H(m, c)$, where H is a hash function.
- U_m denotes that the output of Encaps and Decaps will be of the form $H(m)$, where H is a hash function.
- It is standard practice to hash the public key used in the generation of ciphertexts together with the other inputs. [Section 4.3](#) describes the rationale behind this decision and introduces a technique called prefix hashing to achieve the same functionality with reduced overhead.

Four variants of the U transform $U_{m,c}^\perp, U_{m,c}^\times, U_m^\perp, U_m^\times$ are introduced in [\[HHK17\]](#). In practice, all KEM's that make use of the FO-transform also include some kind of public key hash. This inclusion makes no impact on single target security, but mitigates risk of security degradation occurring as the number of users scales.

As an example, let $\text{PKE} = (\text{KeyGen}, \text{Enc}, \text{Dec})$ and $\text{PKE}_1 := (\text{KeyGen}, \text{Enc}_1, \text{Dec}_1) = T[\text{PKE}, G]$ where G is a random oracle. Then $U_{m,c}^\times[\text{PKE}_1, H] := (\text{KeyGen}^\times, \text{Encaps}^\times, \text{Decaps}^\times)$ is a KEM with algorithms described in Figure 3.2.

$\text{KeyGen}^\chi(\cdot)$	$\text{Encaps}^\chi(pk)$	$\text{Decaps}^\chi(sk, c)$
1 : $(pk', sk') \xleftarrow{\$} \text{KeyGen}()$	5 : $m \xleftarrow{\$} \mathcal{M}$	9 : Parse $sk = (sk', s)$
2 : $s \xleftarrow{\$} \mathcal{M}$	6 : $c \leftarrow \text{Enc}_1(pk', m)$	10 : $m' := \text{Dec}_1(sk, c)$
3 : $sk := (sk', s)$	7 : $K := H(m, c)$	11 : if $m' \neq \perp$ return $K := H(m', c)$
4 : return (pk', sk)	8 : return (K, c)	12 : else return $H(s, c)$

Figure 3.2: Key encapsulation mechanism from $\text{FO}_{m,c}^\chi$.

3.3 eFrodoKEM

This section describes eFrodoKEM, an IND-CCA secure KEM, that relies on the (quantum) hardness or learning with errors, and is the central object of study for this paper. eFrodoKEM is a predecessor to FrodoKEM, which is discussed in chapter 4. eFrodoKEM is obtained by applying $\text{FO}_{m,c}^\chi$ to an IND-CPA PKE called FrodoPKE. FrodoKEM is also obtained from FrodoPKE, but applies a different transform to it, specifically $\text{SFO}_{m,c}^\chi$, which is described in Section 4.2. In turn, FrodoPKE, introduced in [BCD⁺16], is an instantiation of an LWE based PKE described in [LP11] by Lindner and Peikert, which is referred to in this document as the Lindner-Peikert cryptosystem, which generalizes cryptosystems presented in [Reg05, PV08, GPV08]. In this section we describe the algorithms comprising eFrodoKEM, and any algorithms that eFrodoKEM makes use of internally.

3.3.1 The Lindner-Peikert Cryptosystem

The Lindner-Peikert cryptosystem has the following parameters:

- Integers $q \geq 2$, $n_1, n_2 \geq 1$.
- Gaussian parameters s_e and s_k .
- Message length ℓ .
- A public uniformly random matrix $\overline{\mathbf{A}} \in \mathbb{Z}_q^{n_1 \times n_2}$.
- Generic error-resistant functions
 - $\text{encode} : \{0, 1\} \rightarrow \mathbb{Z}_q$ and

– **decode** : $\mathbb{Z}_q \rightarrow \{0, 1\}$

such that for some threshold t it holds that **decode**(**encode**(b) + $e \bmod q$) = b for any integer $e \in [-t, t]$.

The PKE is shown in figure 3.3.

KeyGen()	Enc(pk, m)	Dec(sk, c)
1 : $\mathbf{R}_1 \leftarrow D_{\mathbb{Z}, s_k}^{n_1 \times \ell}$	1 : parse $pk = (\overline{\mathbf{A}}, \mathbf{P})$	1 : parse $c^t = [c_1^t \ c_2^t], sk = \mathbf{R}_2$
2 : $\mathbf{R}_2 \leftarrow D_{\mathbb{Z}, s_k}^{n_2 \times \ell}$	2 : $e_1 \leftarrow D_{\mathbb{Z}, s_e}^{n_1}$	2 : $m' = c_1^t \cdot \mathbf{R}_2 + c_2^t$
3 : $\mathbf{P} := \mathbf{R}_1 - \overline{\mathbf{A}} \cdot \mathbf{R}_2$	3 : $e_2 \leftarrow D_{\mathbb{Z}, s_e}^{n_2}$	3 : return decode (m')
4 : $pk := (\overline{\mathbf{A}}, \mathbf{P})$	4 : $e_3 \leftarrow D_{\mathbb{Z}, s_e}^\ell$	
5 : $sk := \mathbf{R}_2$	5 : $e := (e_1, e_2, e_3)$	
6 : return (pk, sk)	6 : $\overline{m} := \text{encode}(m)$	
	7 : $c^t := [c_1^t \ c_2^t] =$	
	$[e_1^t \ e_2^t \ e_3^t + \overline{m}^t] \cdot \begin{bmatrix} \overline{\mathbf{A}} & \mathbf{P} \\ I & \\ & I \end{bmatrix}$	
	8 : return c	

Figure 3.3: Lindner-Peikert cryptosystem.

Intuitively, security follows from LWE, while correctness can be argued as follows: For $m \in \{0, 1\}^\ell$ we have that

$$\overline{m} = \text{encode}(m) \in \mathbb{Z}_q^\ell$$

$$c^t = [e_1^t \ e_2^t \ e_3^t + \overline{m}^t] \cdot \begin{bmatrix} \overline{\mathbf{A}} & \mathbf{P} \\ I & \\ & I \end{bmatrix}$$

$$= [e_1^t \cdot \overline{\mathbf{A}} + e_2^t \ e_1^t \cdot \mathbf{P} + e_3^t + \overline{m}^t]$$

$$\text{And so } c = \begin{bmatrix} e_1^t \cdot \overline{\mathbf{A}} + e_2^t \\ e_1^t \cdot \mathbf{R}_2 + e_3^t + \overline{m}^t \end{bmatrix} \in \mathbb{Z}_q^{(n_2 + \ell) \times 1}$$

Now observe that

$$\begin{aligned}
m' &= (e_1^t \cdot \overline{\mathbf{A}} + e_2^t) \cdot \mathbf{R}_2 + e_1^t \cdot \mathbf{P} + e_3^t + \overline{m}^t \\
&= e_1^t \cdot \overline{\mathbf{A}} \cdot \mathbf{R}_2 + e_2^t \cdot \mathbf{R}_2 + e_1^t \cdot \mathbf{P} + e_3^t + \overline{m}^t \\
&= e_1^t \cdot (\overline{\mathbf{A}}\mathbf{R}_2 + \mathbf{P}) + e_2^t \cdot \mathbf{R}_2 + e_3^t + \overline{m}^t \\
&= e_1^t \cdot \mathbf{R}_1 + e_2^t \cdot \mathbf{R}_2 + e_3^t + \overline{m}^t \\
&= (e^t + \begin{bmatrix} 0 & 0 & \overline{m}^t \end{bmatrix}) \cdot \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ I \end{bmatrix} \\
&= e_1 \mathbf{R} + \overline{m}^t,
\end{aligned}$$

where

$$\mathbf{R} = \begin{bmatrix} R_1 \\ R_2 \\ I \end{bmatrix}$$

This correctness argument generalizes to FrodoPKE A full argument for correctness can be found in [LP11].

3.3.2 Auxiliary Algorithms for Frodo

The algorithms shown in Figures 3.4 to 3.16 make use of these parameters:

- $q = 2^D$: an integer modulus with $D \leq 16$
- $n, \overline{m} : \overline{n}$ integer matrix dimensions with $n \equiv 0 \pmod{8}$
- $b \leq D$: the number of bits encoded in each matrix entry
- $\ell = B \cdot \overline{m} \cdot \overline{n}$
- len_m : the message length

- $\mathcal{M} = \{0, 1\}^{\text{len}_m}$: the message space
- $\text{len}_{\text{seed}_A}$: the length of the seed used for matrix generation
- $\text{len}_{\text{seed}_{SE}}$: the length of the seed used for error sampling
- T_χ : the distribution table for sampling

The algorithms in Figures 3.4 to 3.11 are used for FrodoPKE, as well as eFrodoKEM and FrodoKEM (in chapter 4). We begin with an encoder and decoder. The encoding function allows for bit strings to be represented as matrices over the integers mod q for the purpose of encryption and decryption with LWE. The decoding functions converts matrix representations back into bit strings.

We represent a bit string as $\mathbf{k}$ and a matrix as $\mathbf{K}$. We let $\overline{m}$ and $\overline{n}$ represent the dimensions of $\mathbf{K}$. If $\mathbf{k}$ were an $\overline{m}\overline{n}$ bitstring, representing $\mathbf{k}$ as a matrix would be trivial. More generally, strings of length $\ell = B \cdot \overline{m}\overline{n}$ are represented as matrices by first interpreting each B bit sub string as an integer k , then encoding by multiplying by $q/2^B = 2^{D-B}$.

Frodo.encode($\mathbf{k}$)

```

1 : for ( $i = 0; i < \overline{m}; i \leftarrow i + 1$ )
2 :   for ( $j = 0; j < \overline{n}; j \leftarrow j + 1$ )
3 :      $k \leftarrow \sum_{l=0}^{B-1} \mathbf{k}_{(i\overline{n}+j)B+l} \cdot 2^l$ 
4 :      $\mathbf{K}_{ij} \leftarrow k \cdot q/2^B$ 
5 : return  $\mathbf{K}$ 

```

Figure 3.4: Encoding function for Frodo.

Frodo.decode($\mathbf{K}$)

```

6 : for ( $i = 0; i < \overline{m}; i \leftarrow i + 1$ )
7 :   for ( $j = 0; j < \overline{n}; j \leftarrow j + 1$ )
8 :      $k \leftarrow \lfloor \mathbf{K}_{ij} \cdot 2^B / q \rfloor \bmod 2^B$ 
9 :      $k = \sum_{l=0}^{B-1} k_l \cdot 2^l$ 
10 :    for ( $l = 0; l < B; l \leftarrow l + 1$ )
11 :       $\mathbf{k}_{(i\overline{n}B+j)B+l} \leftarrow k_l$ 
12 : return  $\mathbf{k}$ 

```

Figure 3.5: Decoding function for Frodo.

FrodoPKE and eFrodoKEM also make use of algorithms for converting matrices to bit strings and vice versa shown in Figure 3.6 and Figure 3.7 respectively.

```

FrodoKEM.pack(C)
1 : for( $i = 0; i < n_1; i \leftarrow i + 1$ )
2 :   for( $j = 0; j < n_2; j \leftarrow j + 1$ )
3 :      $C_{i,j} = \sum_{l=1}^{D-1} c_l 2^l$ 
4 :     for( $l = 0; l < D; l \leftarrow l + 1$ )
5 :        $\mathbf{b}_{(i \cdot n_2 + j)D + l} \leftarrow \mathbf{c}_{D-1-l}$ 
6 :   return  $\mathbf{b}$ 

```

Figure 3.6: Matrix packing function for Frodo.

```

FrodoKEM.unpack(b)
7 : for( $i = 0; i < n_1; i \leftarrow i + 1$ )
8 :   for( $j = 0; j < n_2; j \leftarrow j + 1$ )
9 :      $C_{i,j} \leftarrow \sum_{l=0}^{D-1} \mathbf{b}_{(i \cdot n_2 + j)D + l} \cdot 2^{D-1-l}$ 
10 : return  $\mathbf{C}$ 

```

Figure 3.7: Matrix unpacking function Frodo.

FrodoKEM and eFrodoKEM sample errors from a discrete Gaussian distribution over the integers, centered at 0 with support S_χ . The sampling algorithm shown in Figure 3.8 achieves this task in constant time so as to avoid side-channel attacks. It takes as input a random bitstring $\mathbf{r} = (\mathbf{r}_0, \dots, \mathbf{r}_{\text{len}_\chi-1})$

```

Frodo.sample( $\mathbf{r} = (\mathbf{r}_0, \dots, \mathbf{r}_{\text{len}_\chi-1})$ ),  $T_\chi$ 
1 :  $t \leftarrow \sum_{i=1}^{\text{len}_\chi-1} \mathbf{r}_i \cdot 2^{i-1}$ 
2 :  $e \leftarrow 0$ 
3 : for( $z = 0; z < s; z \leftarrow z + 1$ )
4 :   if  $t > T_\chi(z)$ 
5 :      $e \leftarrow e + 1$ 
6 :    $e \leftarrow (-1)^{\mathbf{r}_0} \cdot e$ 
7 : return  $e$ 

```

Figure 3.8: Sampling function for Frodo.

Frodo.SampleMatrix ($\mathbf{r} = (\mathbf{r}^{(0)}, \dots, \mathbf{r}^{(n_1 n_2 - 1)}), T_X$)
<pre> 1 : for($i = 0; i < n_1; i \leftarrow i + 1$) 2 : for($j = 0; j < n_2; j \leftarrow j + 1$) 3 : $\mathbf{E}_{ij} \leftarrow \text{Frodo.sample}(\mathbf{r}^{(in_2+j)})$ 4 : return $\mathbf{E}$ </pre>

Figure 3.9: Matrix sampling function for Frodo.

Frodo.Gen (seed _A) using AES128
<pre> 1 : for ($i = 0; i < n; i \leftarrow i + 1$) do 2 : ($j = 0; j < n; i \leftarrow j + 8$) do 3 : $\mathbf{b} \leftarrow \langle i \rangle \langle j \rangle 0 \dots 0$ where $\langle i \rangle, \langle j \rangle \in \{0, 1\}^{16}$ 4 : $\langle c_{i,j} \rangle \langle c_{i,j+1} \rangle \dots \langle c_{i,j+7} \rangle \leftarrow \text{AES128}_{\text{seed}_A}(\mathbf{b})$ 5 : for($k = 0; k < 8; k \leftarrow k + 1$) do 6 : $\mathbf{A}_{i,j+k} \leftarrow c_{i,j+k} \pmod q$ 7 : return $\mathbf{A}$ </pre>

Figure 3.10: Frodo.Gen() using AES128

Frodo.Gen (seed _A) using SHAKE128
<pre> 1 : for ($i = 0; i < n; i \leftarrow i + 1$) do 2 : $\mathbf{b} \langle i \rangle \text{seed}_A \in \{0, 1\}^{16 + \text{len}_{\text{seed}_A}}$ 3 : $\langle c_{i,0} \rangle \langle c_{i,1} \rangle \dots \langle c_{i,n-1} \rangle \leftarrow \text{SHAKE128}(\mathbf{b}, 16n)$ 4 : for ($j = 0; j < n; j \leftarrow j + 1$) do 5 : $\mathbf{A}_{ij} \leftarrow c_{i,j} \pmod q$ 6 : return $\mathbf{A}$ </pre>

Figure 3.11: Frodo.Gen() with SHAKE128.

The algorithm shown in Figure 3.9 samples $n_1 \times n_2$ matrix of error terms.
The algorithms shown in Figures 3.10 and 3.11 are used to generate a random matrix given

a seed $\text{seed}_{\mathbf{A}}$. Two versions are presented, which each make use of a different hash function.

3.3.3 FrodoPKE

FrodoPKE is an IND-CPA-secure PKE, based on the hardness of LWE. In Figures 3.11-3.13 we provide a description of FrodoPKE with notation consistent with the rest of this document. For a thorough review, see the FrodoKEM specifications [ABD⁺23b]. The purpose of including eFrodoKEM specifications in this document is to show how KEM implementations that make use of SFO-transform (such as FrodoKEM) differ from implementations that make use of the FO-transform (such as eFrodoKEM.) FrodoKEM specifications are given in Section 4.4.

A proof of the IND-CCA security of FrodoPKE can be found in [ABD⁺23b] Section 2.2.7.

FrodoPKE.KeyGen()
1 : $\text{seed}_{\mathbf{A}} \xleftarrow{\$} \{0, 1\}^{\text{len}_{\text{seed}_{\mathbf{A}}}}$
2 : $\mathbf{A} \xleftarrow{\$} \text{Frodo.Gen}(\text{seed}_{\mathbf{A}})$
3 : $\text{seed}_{\mathbf{SE}} \xleftarrow{\$} \{0, 1\}^{\text{len}_{\text{seed}_{\mathbf{SE}}}}$
4 : $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \dots, \mathbf{r}^{(2n\bar{n}-1)}) \leftarrow \text{SHAKE}(0x5 \text{seed}_{\mathbf{SE}}, 2n\bar{n} \cdot \text{len}_{\chi})$
5 : $\mathbf{S}^T \leftarrow \text{Frodo.SampleMatrix}(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \dots, \mathbf{r}^{(n\bar{n}-1)}, \bar{n}, n, T_{\chi})$
6 : $\mathbf{E} \leftarrow \text{Frodo.SampleMatrix}(\mathbf{r}^{(n\bar{n})}, \mathbf{r}^{(n\bar{n})+1}, \dots, \mathbf{r}^{(2n\bar{n})-1}, \bar{n}, n, T_{\chi})$
7 : $\mathbf{B} \xleftarrow{\$} \mathbf{AS} + \mathbf{E}$
8 : $pk \leftarrow (\text{seed}_{\mathbf{A}}, \mathbf{B})$
9 : $sk \leftarrow \mathbf{S}^T$
10 : return (pk, sk)

Figure 3.12: Key generation algorithm for FrodoPKE.

FrodoPKE.Enc(pk, m)	
1 :	parse $pk = (\text{seed}_A, B)$
2 :	$A \xleftarrow{\$} \text{Frodo.Gen}(\text{seed}_A)$
3 :	$\text{seed}_{SE} \xleftarrow{\$} \{0, 1\}^{\text{len}_{\text{seed}_{SE}}}$
4 :	$(r^{(0)}, r^{(1)}, \dots, r^{(2\overline{m}n + \overline{m}n - 1)}) \leftarrow \text{SHAKE}(0x96 \text{seed}_{SE}, (2\overline{m}n + \overline{m}n - 1)\text{len}_{\chi})$
5 :	$S' \leftarrow \text{Frodo.SampleMatrix}(r^{(0)}, r^{(1)}, \dots, r^{(\overline{m}n - 1)}, \overline{m}, n, T_{\chi})$
6 :	$E' \leftarrow \text{Frodo.SampleMatrix}(r^{(\overline{m}n)}, r^{(\overline{m}n) + 1}, \dots, r^{(2\overline{m}n) - 1}, \overline{m}, n, T_{\chi})$
7 :	$E'' \leftarrow \text{Frodo.SampleMatrix}(r^{(2\overline{m}n)}, r^{(2\overline{m}n) + 1}, \dots, r^{(2\overline{m}n) + \overline{m}n - 1}, \overline{m}, n, T_{\chi})$
8 :	$B \leftarrow S'A + E'$
9 :	$V \leftarrow S'B + E''$
10 :	return $(C_1, C_2) := (B', V + \text{encode}(m))$

Figure 3.13: Encryption algorithm for FrodoPKE.

FrodoPKE.Dec(sk, c)	
1 :	parse $c = (C_1, C_2), sk = S^T$
2 :	$M \leftarrow C_2 - C_1 S$
3 :	return M

Figure 3.14: Decryption algorithm for FrodoPKE.

3.3.4 eFrodoKEM Specifications

eFrodoKEM is the name given to the NIST round 3 submission to distinguish it from the revised, salted version shown in chapter 4. It is constructed by applying $\text{FO}_{m,c}^{\chi}$ to FrodoPKE.

Remark. More accurately, eFrodoKEM makes use of a the $\text{FO}_{\text{pkh},m,c}^{\chi}$ -transform. This means that a hash of the public key is included as input into H when generating ciphertexts. The rationale behind this is described in [Section 4.2.3](#). In practice the public key hash **pkh**

is often omitted from the subscript in the notation, though almost universally used in implementations to prevent multi-user attacks.

<p>eFrodoKEM.Encaps(seed_A b)</p> <hr/> <pre> 1 : $m \xleftarrow{\\$} \{0, 1\}^{\text{len}_m}$ 2 : $\mathbf{pkh} \leftarrow \text{SHAKE}(pk, \text{len}_{\mathbf{pkh}})$ 3 : $\text{seed}_{\mathbf{SE}} \parallel \mathbf{k} \leftarrow \text{SHAKE}(\mathbf{pkh}, m, \text{len}_{\text{seed}_{\mathbf{SE}}} + \text{len}_{\mathbf{k}})$ 4 : $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \dots, \mathbf{r}^{(2\overline{m}n + \overline{m}\overline{n} - 1)}) \leftarrow \text{SHAKE}(0 \times 96 \parallel \text{seed}_{\mathbf{SE}}, (2\overline{m}n + \overline{m}\overline{n}) \cdot \text{len}_{\chi})$ 5 : $\mathbf{S}' \leftarrow \text{Frodo.SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \dots, \mathbf{r}^{\overline{m}n - 1}), \overline{m}, n, T_{\chi})$ 6 : $\mathbf{E}' \leftarrow \text{Frodo.SampleMatrix}((\mathbf{r}^{(\overline{m}n)}, \mathbf{r}^{(\overline{m}n + 1)}, \dots, \mathbf{r}^{2\overline{m}n - 1}), \overline{m}, n, T_{\chi})$ 7 : $\mathbf{A} \leftarrow \text{Frodo.Gen}(\text{seed}_{\mathbf{A}})$ 8 : $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$ 9 : $\mathbf{c}_1 \leftarrow \text{Frodo.pack}(\mathbf{B}')$ 10 : $\mathbf{E}'' \leftarrow \text{SampleMatrix}(\mathbf{r}^{(2\overline{m}n)}, \mathbf{r}^{(2\overline{m}n + 1)}, \dots, \mathbf{r}^{2\overline{m}n + \overline{m}\overline{n} - 1}), \overline{m}, \overline{n}, T_{\chi})$ 11 : $\mathbf{B} \leftarrow \text{Frodo.unpack}(\mathbf{b})$ 12 : $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$ 13 : $\mathbf{C} \leftarrow \mathbf{V} + \text{Frodo.encode}(m)$ 14 : $\mathbf{c} \leftarrow \text{Frodo.pack}(\mathbf{C})$ 15 : $\mathbf{ss} \leftarrow \text{SHAKE}(\mathbf{c}_1, \mathbf{c}_2 \parallel \mathbf{k})$ 16 : return ($\mathbf{c}_1 \parallel \mathbf{c}_2, \mathbf{ss}$) </pre>

Figure 3.15: Encapsulation for eFrodoKEM.

```

eFrodoKEM.Decaps( $\mathbf{c}_1 \parallel \mathbf{c}_2, (\mathbf{s} \parallel \text{seed}_{\mathbf{A}} \parallel \mathbf{b}, \mathbf{S}^T, \mathbf{pkh})$ )
1 :  $\mathbf{B}' \leftarrow \text{Frodo.unpack}(\mathbf{c}_1, \overline{m}, n)$ 
2 :  $\mathbf{C} \leftarrow \text{Frodo.unpack}(\mathbf{c}_1, \overline{m}, \overline{n})$ 
3 :  $\mathbf{M} \leftarrow \mathbf{C} - \mathbf{B}'\mathbf{S}$ 
4 :  $m' \leftarrow \text{Frodo.decode}(\mathbf{M})$ 
5 :  $\text{seed}'_{\mathbf{SE}} \parallel \mathbf{k}' \leftarrow \text{SHAKE}(\mathbf{pkh}, m', \text{len}_{\text{seed}_{\mathbf{SE}}} + \text{len}_{\mathbf{k}})$ 
6 :  $\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \dots, \mathbf{r}^{(2\overline{m}n + \overline{m}\overline{n} - 1)} \leftarrow \text{SHAKE}(0x96 \parallel \text{seed}_{\mathbf{SE}}, (2\overline{m}n + \overline{m}\overline{n}) \cdot \text{len}_{\chi})$ 
7 :  $\mathbf{S}' \leftarrow \text{Frodo.SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \dots, \mathbf{r}^{\overline{m}n - 1}), \overline{m}, n, T_{\chi})$ 
8 :  $\mathbf{E}' \leftarrow \text{Frodo.SampleMatrix}((\mathbf{r}^{(\overline{m}n)}, \mathbf{r}^{(\overline{m}n + 1)}, \dots, \mathbf{r}^{2\overline{m}n - 1}), \overline{m}, n, T_{\chi})$ 
9 :  $\mathbf{A} \leftarrow \text{Frodo.Gen}(\text{seed}_{\mathbf{A}})$ 
10 :  $\mathbf{B}'' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$ 
11 :  $\mathbf{E}'' \leftarrow \text{SampleMatrix}(\mathbf{r}^{(2\overline{m}n)}, \mathbf{r}^{(2\overline{m}n + 1)}, \dots, \mathbf{r}^{2\overline{m}n + \overline{m}\overline{n} - 1}), \overline{m}, \overline{n}, T_{\chi})$ 
12 :  $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$ 
13 :  $\mathbf{C}' \leftarrow \mathbf{V} + \text{Frodo.encode}(m')$ 
14 : if  $\mathbf{B}' \parallel \mathbf{C} = \mathbf{B}'' \parallel \mathbf{C}'$ 
15 :    $\overline{k} := k$ 
16 : else
17 :    $\overline{k} := s$ 
18 :  $\mathbf{ss} = \text{SHAKE}(\mathbf{c}_1 \parallel \mathbf{c}_2 \parallel \overline{k} \parallel \text{len}_{\mathbf{ss}})$ 
19 : return  $\mathbf{ss}$ 

```

Figure 3.16: Decapsulation for eFrodoKEM.

Chapter 4

Multi-Target Security

In this chapter, we introduce the salted Fujisaki-Okamoto transform (SFO), and bound the multi-target IND-CCA security of resulting KEMs. In the setting with one ciphertext per key pair, security bounds include some terms which come from the probability that an adversary queries some uniformly random value. These “guessing bounds” will be negligible. In the multi-target setting, these guessing bounds become replaced by birthday bounds, which may not be negligible. We see in the case of **eFrodoKEM**, that these birthday bounds induce a simple, albeit costly, attack that can win the multi-target security game with near certainty. The introduction of salting reduces the success probability of these attacks dramatically.

In section 4.1, we introduce security definitions for the multi-target setting, some of which we have not seen presented in the literature before. Furthermore, we show the multi-target attack in **eFrodoKEM**. In section 4.2, we show bounds for KEMs resulting from each of the 4 variants of the SFO-transform, carefully adapting the proofs of [HHK17] to the multi-target setting, using the game hopping technique described in [Sho04], and incorporating salts to improve birthday bounds. In section 4.3, we briefly discuss multi-target security, where we are concerned with how adversary advantage scales with the number of users, and we describe a technique from [DHK⁺21] called prefix hashing, that can dramatically reduce overhead for lattice-based KEMs, which induces slight variations of SFO. Finally, we apply our findings to FrodoKEM, and experimentally test the overhead, comparing **eFrodoKEM** [ABD⁺23b], FrodoKEM [ABD⁺23a], and $\text{FrodoKEM}_{F(pk),m,c}$, in order of increasing efficiency.

4.1 Multi-Target Security Definitions

In the Multi-target setting we consider two axes in which security can degrade: the number of users of a protocol, meaning the number of key pairs sampled from **KeyGen**, and the total number of ciphertexts issued. We make use of two parameters, n representing the total number of challenge ciphertexts, and u representing the total number of users. In this particular paper *multi-target security* refers to both *multi-user security* ($u > 1$), and *multi-ciphertext security* ($n > u$). In previous works, such as [DHK⁺21], multi-target security has been used to refer only to definitions in which $u > 1$, while $n = u$ (multiple users, but only one ciphertext per user).

If only a limited encapsulations may be produced, this would increase the computational cost of using said KEM, as key generation would have to be run multiple times to achieve the same functionality as a multi-target secure KEM. For lattice based KEMs in particular, key generation is about as computationally costly as encapsulation [LP11], see [section 4.2](#). Furthermore, the use-cases for said KEM would be reduced to those in which limiting the number of encapsulations produced under a given key is possible.

4.1.1 Security Definitions

The following are formal security definitions in the multi-target setting that will be used in this chapter. The variable n denotes the number of challenge ciphertexts, and u denotes the number of users. There is another axis for security bounds, which is the number of public keys.

Definition 4.1.1 (Multi-target OW-ATK security of a PKE ($\text{OW}_{n,u}\text{-ATK}$)). Let

$$\mathcal{O}_{\text{ATK}} := \begin{cases} - & \text{if ATK = CPA} \\ \text{PCO} & \text{if ATK = PCA} \\ \text{PCO, CVO} & \text{if ATK = PCVA} \end{cases}$$

and $\mathfrak{L}_M$ denotes the set of plaintexts underlying the challenge ciphertexts encrypted with pk . Consider the experiment shown in Figure 4.1.

$$\text{Adv}_{\text{PKE}}^{\text{OW}_{n,u}\text{-ATK}}(\mathbf{A}) := \Pr[\text{Exp}_{\text{PKE}}^{\text{OW}_{n,u}\text{-ATK}}(\mathbf{A}) \Rightarrow 1]$$

$\text{Exp}_{\text{PKE}}^{\text{OW}_{n,u}\text{-ATK}}(\mathcal{A})$	$\text{Chall}(j)$
1 : for $i = 1, \dots, u$	7 : $m_i^* \xleftarrow{\$} \mathcal{M}$
2 : $(pk, sk) \xleftarrow{\$} \text{KeyGen}()$	8 : $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m_i^*\}$
3 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	9 : $c^* \leftarrow \text{Enc}(pk_j, m_i^*)$
4 : $m' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ATK}}, \text{Chall}}(\vec{pk})$	10 : return c^*
5 : return $\llbracket m' \in \mathfrak{L}_{M_j} \rrbracket$	$\text{CVO}(j, c \neq c^*)$
$\text{PCO}(j, m \in \mathcal{M}, c)$	11 : $m \leftarrow \text{Dec}(sk_j, c)$
6 : return $\llbracket \text{Dec}(sk_j, c) = m \rrbracket$	12 : return $\llbracket m \in \mathcal{M} \rrbracket$

Figure 4.1: Experiment for $\text{OW}_{n,u}\text{-ATK}$ -security.

Where $\mathfrak{L}_{M_j}$ denotes the list of messages encrypted under pk_j , and $\mathfrak{L}_{C_j}$ denotes the corresponding list of ciphertexts. Both lists are initialized to be empty.

We say that a PKE is multi-target one way secure, if, for any adversary running an attack with time T , that has success probability ϵ , we have that

$$\frac{T}{\epsilon} \geq 2^\kappa$$

where κ is the desired security level.

Definition 4.1.2 (Multi-Target IND-CCA Security of a KEM ($\text{IND}_{n,u}\text{-CCA}$)). Consider the experiment shown in Figure 4.2

$$\text{Adv}_{\text{KEM}}^{\text{IND}_{n,u}\text{-CCA}}(\mathcal{A}) := \left| \Pr[\text{Exp}_{\text{KEM}}^{\text{IND}_{n,u}\text{-CCA}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right|$$

Where $\mathfrak{L}_{M_j}$ denotes the list of messages encrypted under pk_j and $\mathfrak{L}_{C_j}$ is the corresponding list of ciphertexts. Both lists are initialized to be empty.

We say that a KEM is indistinguishable under chosen multi-target attack ($\text{IND}_{n,u}\text{-CCA}$) if for any adversary running an attack with time T , that has success probability ϵ , we have that

$$\frac{T}{\epsilon} \geq 2^\kappa$$

$\text{Exp}_{\text{KEM}}^{\text{IND}_{n,u}\text{-CCA}}(\mathcal{A})$	$\text{Chall}(j)$
1 : $b \xleftarrow{\$} \{0, 1\}$	6 : $(c, \text{ss}_0) \xleftarrow{\$} \text{Encaps}(pk_j)$
2 : for $j = 1, \dots, u$	7 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	8 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c\}$
4 : $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Decaps}}, \text{Chall}}(\vec{pk})$	9 : return ss_b
5 : return $\llbracket b = b' \rrbracket$	$\mathcal{O}_{\text{Decaps}}(j, c \neq c^*)$
	10 : return $\text{Decaps}(sk_j, c)$

Figure 4.2: Experiment for $\text{IND}_{n,u}\text{-CCA}$ -security.

where κ is the desired security level.

In [BBM00], Bellare et al. prove what we will refer to in this paper as the trivial bound on multi-target security. Which is that

$$\text{Adv}^{\text{IND}_{n,u}\text{-CCA}} \leq (n \cdot u) \cdot \text{Adv}^{\text{IND-CCA}}$$

where u is the number of users and n is the total number of ciphertexts. In this chapter we improve on this bound significantly, for KEMs built from the FO transform.

4.1.2 Multi-Target Attack on eFrodoKEM

The following attack on eFrodoKEM is simple, involving nothing other than random brute force computation of sufficiently many encapsulations. Note that this attack assumes multiple challenge ciphertexts, and therefore falls out of the scope of IND-CCA security. Instead is is an attack on the $\text{IND}_{n,u}\text{-CCA}$ security of eFrodoKEM, Still, it exposes important vulnerabilities, which initiated a study into techniques for establishing multi-target security in KEMs.

We provide description of the attack for a generic $\text{KEM}^\mathcal{X} = \text{FO}^\mathcal{X}[\text{PKE}, \mathcal{G}, \mathcal{H}]$, with the parameters of eFrodoKEM640. Observe that in [eFrodoKEM.Encaps\(\$\cdot\$ \)](#) the value $c \leftarrow \text{Enc}_1(pk', m) = \text{Enc}(pk, m; \mathcal{G}(m))$ depends entirely on m and pkh . Consequently, the shared secret $\text{ss} \leftarrow \mathcal{H}(m, c)$ is completely determined by m . The adversary queries $\text{Chall}(j)$ n times,

using the same j ¹ The adversary prepares N ciphertexts of their own $c'_1, \dots, c'_N$, using pk_j . Then for each of the n challenges there is approximately an $\frac{N}{|\mathcal{M}|}$ chance that it uses an m that was also used for one of the $c'_1, \dots, c'_N$. Over n challenge ciphertexts, the probability of finding a collision, is

$$\Pr[\text{COLL}] \approx \frac{nN}{|\mathcal{M}|}.$$

In the specific case of **eFrodoKEM** in **eFrodoKEM.Encaps()** we see that the values **seed_{SE}** and **k** are entirely determined by the public key hash value **pkh**, and m . Likewise, in lines 4-6 the secret key **S'** and **E', E''** depend entirely on m and **pkh**. The probability that an adversary finds a collision between one of N ciphertexts that they compute, and one of the n challenges, is again

$$\Pr[\text{COLL}] \approx \frac{nN}{|\mathcal{M}|}.$$

We assume an adversary that could issue as many as 2^{64} queries to **Chall**. For **eFrodoKEM640**, $|\mathcal{M}| = 2^{128}$. With these parameters the adversary would be able to find a collision with probability approximately 1 and completely break the multi-target security of **eFrodoKEM**.²

Note that this attack does not relate to multi-target security of **PKE**. It only depends on $|\mathcal{M}|$ and n , and has to do with the fact that $\text{PKE}_1 := \text{T}[\text{PKE}, \text{G}]$ is deterministic. In Theorem 4.2.2 this attack is contextualized within the security bounds for generic **FO** based **KEMS**. This illustrates an important consideration in constructing post-quantum **KEMS**: multi-target security of a **PKE** is not enough to guarantee the multi-target security of the **KEM** obtained by applying the **FO** transform to that **PKE**.

4.2 The FO/SFO Transform in the Multi-Target Setting

In this subsection, we introduce and formalize a countermeasure to the previous attack, which has been described in [ABD⁺23a]. We showed in the previous subsection that even

¹This could be seen as a multi-ciphertext attack, with $n > 1$ and $u = 1$.

²For **eFrodoKEM976** $\mathcal{M} = 2^{192}$ then the success probability is approximately $\frac{1}{2^{64}}$, obtaining 128 bits of security, while targeting 192 bits of security. For **eFrodoKEM1344** $\mathcal{M} = 2^{256}$ then the success probability is approximately $\frac{1}{2^{128}}$ obtaining 192 bits of security, while targeting 256 bits of security.

if a PKE is $\text{IND}_{n,u}\text{-CPA}$ -secure, the KEM that results from applying the FO-transform to it may be vulnerable to a generic attack. The countermeasure is to introduce a uniformly random salt in the encryption process, so that the collision finding probability depends not only on $|\mathcal{M}|$ and n , but also on the length of the uniformly random salt. In this section we describe a modified version of the T transform called the ST-transform (Salted-T), and the corresponding salted SFO- transform (salted FO). We are unaware of any other way of addressing collision finding attacks in FO-based KEMs at this time.

Let $\text{PKE} = (\text{KeyGen}, \text{Enc}, \text{Dec})$ be a generic $\text{IND}\text{-CPA}$ -secure public key encryption scheme. The T transform takes two arguments, PKE, and a hash function G , modelled as a random oracle. $\text{ST}[\text{PKE}, G] = \text{PKE}_1 = (\text{Enc}_1, \text{Dec}_1, \text{KeyGen})$ is a new public key encryption scheme with Enc_1 and Dec_1 described in Figure 4.1. Changes from the generic T transform are highlighted in red. The SFO-transform is the composition of the ST trans-

$\text{ST}[\text{PKE}].\text{Enc}(pk, m)$	$\text{ST}[\text{PKE}].\text{Dec}(sk, c \text{salt})$
1 : $\text{salt} \xleftarrow{\$} \{0, 1\}^{\text{len}_{\text{salt}}}$	4 : if $m' = \perp$ or $\text{Enc}(pk, m', G(m' \text{salt})) \neq c$
2 : $c \leftarrow \text{Enc}(pk, m; G(m \text{salt}))$	5 : return $\perp$
3 : return $c \text{salt}$	6 : else return m'

Figure 4.3: Encryption and decryption for $\text{ST}[\text{PKE}]$.

form, with one of the four variants of the U-transform described in section 3.1.2.

In the following two sections we prove a number of results relating to the $\text{IND}_{n,u}\text{-CCA}$ security of KEMs obtained by applying the FO/SFO transform to an $\text{IND}_{n,u}\text{-CPA}$ secure PKE. Theorem 4.2.2 gives a bound relating to the generic T transform. The proofs will be formulated as a sequence of games and associated bounds in the style of [Sho04]. The sequence of changes to the games used will follow the proofs in [HHK17, DHK⁺21, BHH⁺19, LP11], however the underlying games will be multi-target versions of their equivalents in [HHK17], which leads to different bounds. Theorem 4.2.2 shows the impact of using SFO instead of FO, giving a much tighter bound than Theorem 4.2.2.

4.2.1 The T-transform: $\text{IND}_{n,u}\text{-CPA}$ to $\text{OW}_{n,u}\text{-PCVA}$

In the following proofs we make use of lemma 4.2.1 taken from [HHK17].

Lemma 4.2.1. For any $\text{OW}_{n,u}\text{-CPA}$ adversary $\mathbf{B}$, there exists an $\text{IND}_{n,u}\text{-CPA}$ adversary $\mathbf{A}$ with the same running time as $\mathbf{B}$ such that

$$\text{Adv}_{\text{PKE}}^{\text{OW}_{n,u}\text{-PCA}} \leq \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}} + \frac{n}{|\mathcal{M}|}$$

We begin with an intuitive overview of the forthcoming proofs. The following two proofs involve a sequence of game hops. One aspect of the proof strategy is to modify the oracles PCO and CVO so that they can be simulated by a CPA adversary, by removing any dependence on the secret key sk . Another aspect of the proof is to raise the flag CHAL to indicate that the adversary has queried a message underlying a challenge ciphertext, and to equate this event with an adversarial goal for a different game.

Theorem 4.2.2 ($\text{PKE } \text{IND}_{n,u}\text{-CPA} \xrightarrow{\text{ROM}} \text{T}[\text{PKE}] \text{OW}_{n,u}\text{-PCVA}$). *Suppose PKE is δ -correct and γ -spread. Then $\text{T}[\text{PKE}]$ is δ_1 -correct in the random oracle model with $\delta_1(q_G) = q_G \cdot \delta$. Moreover for any $\text{OW}_{n,u}\text{-PCVA}$ -adversary $\mathbf{B}$ issuing at most q_G queries to the random oracle $\mathbf{G}$, q_P queries to a plaintext checking oracle PCO , and q_V queries to a ciphertext validity oracle CVO , there exists an $\text{IND}_{n,u}\text{-CCA}$ adversary $\mathbf{A}$ such that*

$$\text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCVA}}(\mathbf{B}) \leq (q_G + q_P) \cdot \delta + q_V \cdot 2^{-\gamma} + \frac{n \cdot (2q_G + 1)}{|\mathcal{M}|} + 3 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$$

where $\mathbf{A}$ and $\mathbf{B}$ have similar run-time.

Proof. The correctness bound is trivial. To prove the security claim, consider the sequence of games starting from Figure 4.4. Lines in red represent changes from previous games.

Since $\mathbf{G}_0$ is the $\text{OW}_{n,u}\text{-PCVA}$ game we have

$$|\Pr[\mathbf{G}_0 \Rightarrow 1]| = \text{Adv}_{\text{T}[\text{PKE}]}^{\text{OW}_{n,u}\text{-PCVA}}(\mathbf{B})$$

In $\mathbf{G}_1$, the only change is to the oracle CVO which now returns 1 if and only if there was a previous query $r \leftarrow \mathbf{G}(j, m')$ such that $\text{Enc}(pk_j, m'; r) = c$. Consider a query $\text{CVO}(j, c)$ with $m' \leftarrow \text{Dec}(sk_j, c)$.

Case 1: $\text{CVO}(j, c) = 1$ in $\mathbf{G}_1$, then $(m', r) \in \mathcal{L}_G$, and $\text{Enc}(pk_j, m'; r) = c$, hence $\mathbf{G}(j, m') = r$ and $\text{Enc}(pk_j, m'; \mathbf{G}(j, m')) = c$. Thus $\text{CVO}(c) = 1$ in $\mathbf{G}_0$ as well.

Case 2: $\text{CVO}(j, c) = 0$ in $\mathbf{G}_1$. Here, we consider 2 sub-cases. Either $\text{CVO}(j, c) = 0$ in $\mathbf{G}_0$, in which case $\mathbf{G}_1$ and $\mathbf{G}_0$ are identical, or $\text{CVO}(j, c) = 1$ in $\mathbf{G}_0$. In the latter case,

$\mathbf{G}_0(\mathbf{B})$	$\mathbf{PCO}(j, m \in \mathcal{M}, c)$
1 : for $j = 1, \dots, u$	11 : $m' \leftarrow \text{Dec}(sk_j, c)$
2 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	12 : return $\llbracket m = m' \text{ and } \text{Enc}(pk_j, m'; \mathbf{G}(j, m')) = c \rrbracket$
3 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	$\mathbf{CVO}(j, c)$
4 : $(j, m') \leftarrow \mathbf{B}^{\text{Chall}, \mathbf{PCO}, \mathbf{CVO}, \mathbf{G}}(\vec{pk})$	13 : if $c \in \mathcal{L}_{C_j}$
5 : return $\llbracket m' \in \mathcal{L}_{M_j} \rrbracket$	14 : abort
$\mathbf{Chall}(j)$	15 : $m' \leftarrow \text{Dec}(sk_j, c)$
6 : $m \xleftarrow{\$} \mathcal{M}$	16 : return $\llbracket m' \in \mathcal{M} \text{ and } \text{Enc}(pk_j, m'; \mathbf{G}(j, m')) = c \rrbracket$
7 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	$\mathbf{G}(m)$
8 : $c = \text{Enc}(pk_j, m; \mathbf{G}(m))$	17 : if $\exists r \text{ s.t. } (m, r) \in \mathcal{L}_{\mathbf{G}}$
9 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c\}$	18 : return r
10 : return c	19 : $r \xleftarrow{\$} \mathcal{R}$
	20 : $\mathcal{L}_{\mathbf{G}} \leftarrow \mathcal{L}_{\mathbf{G}} \cup \{(m, r)\}$
	21 : return r

Figure 4.4: $\mathbf{G}_0$ for the proof of Theorem 4.2.2

we have that $\text{Enc}(pk_j, m'; \mathbf{G}(j, m')) = c$, and so for $\mathbf{CVO}(c) = 1$ in $\mathbf{G}_1$, it must be the case that $\mathbf{G}(j, m')$ has not been queried before. This means that the adversary was able to find r such that $\text{Enc}(pk_j, m; r) = \text{Enc}(pk_j, m, \mathbf{G}(m))$ and $\mathbf{G}(m) \neq r$.

This is where we make use of the assumption that PKE is γ -spread, and so

$$\Pr_{r \in \mathcal{R}}[\text{Enc}(pk, m; r) = c] \leq 2^{-\gamma}.$$

Then the probability that $\text{Enc}(pk_j, m'; r) = \text{Enc}(pk_j, m'; \mathbf{G}(j, m'))$, is bounded by $2^{-\gamma}$. The $\mathbf{G}_0$ and $\mathbf{G}_1$ only differ in this case, and since there are q_V total $\mathbf{CVO}$ queries, we have

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq q_V \cdot 2^{-\gamma}$$

In $\mathbf{G}_2$, the $\mathbf{PCO}$ and $\mathbf{CVO}$ oracles no longer check whether $m = \text{Dec}(sk_j, c)$. In addition to the q_G $\mathbf{G}$ queries, each of the q_P $\mathbf{PCO}$ queries also make an implicit query to $\mathbf{G}$, for a maximum total of $q_G + q_P$ total $\mathbf{G}(j, m_i)$. Observe that $\mathbf{G}_2$ and $\mathbf{G}_1$ proceed identically, so

$\mathbf{G}_1(\mathbf{B})$	$\text{PCO}(j, m \in \mathcal{M}, c)$
1 : for $j = 1, \dots, u$	11 : $m' \leftarrow \text{Dec}(sk_j, c)$
2 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	12 : return $\llbracket m = m' \text{ and } \text{Enc}(pk_j, m'; \mathbf{G}(j, m')) = c \rrbracket$
3 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	$\text{CVO}(j, c)$
4 : $m' \leftarrow \mathbf{B}^{\text{Chall}, \text{PCO}, \text{CVO}, \mathbf{G}}(\vec{pk})$	13 : if $c \in \mathcal{L}_{C_j}$
5 : return $\llbracket m' \in \mathcal{L}_M \rrbracket$	14 : abort
$\text{Chall}(j)$	15 : $m' \leftarrow \text{Dec}(sk_j, c)$
6 : $m \xleftarrow{\$} \mathcal{M}$	16 : return $\llbracket (m', r) \in \mathcal{L}_{\mathbf{G}} \text{ and } \text{Enc}(pk_j, m'; r) = c \rrbracket$
7 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	$\mathbf{G}(m)$
8 : $c = \text{Enc}(pk_j, m; \mathbf{G}(m))$	17 : if $\exists r$ s.t. $(m, r) \in \mathcal{L}_{\mathbf{G}}$
9 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c\}$	18 : return r
10 : return c	19 : $r \xleftarrow{\$} \mathcal{R}$
	20 : $\mathcal{L}_{\mathbf{G}} \leftarrow \mathcal{L}_{\mathbf{G}} \cup \{(m, r)\}$
	21 : return r

Figure 4.5: $\mathbf{G}_1$ for the proof of Theorem 4.2.2

long as $\text{Dec}(sk, \text{Enc}(pk, m; \mathbf{G}(m_i))) = m_i$ for all i . Hence they only differ when $\mathbf{B}$ submits a query to PCO or CVO together with an m such that $\mathbf{G}(m)$ exhibits a correctness error. In that case $\mathbf{G}_1$ outputs 0 and $\mathbf{G}_2$ outputs 1. By $\delta(u)$ correctness, the probability that at least one $\mathbf{G}(j, m_i)$ exhibits a correctness error is bounded by $(q_{\mathbf{G}} + q_P) \cdot \delta(u)$. Hence we have

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq (q_P + q_{\mathbf{G}}) \cdot \delta(u)$$

$\mathbf{G}_3$ and $\mathbf{G}_2$ differ only when the flag CHAL is raised, and so

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \Pr[\text{CHAL}]$$

by the difference lemma. Now we establish two bounds, first we bound $\Pr[\mathbf{G}_3 \Rightarrow 1]$, by constructing an adversary $\mathbf{C}$ that against the $\text{OW}_{n,u}$ -CPA-security of PKE, then we bound $\Pr[\text{CHAL}]$ to complete the proof. $\mathbf{C}$ is an $\text{OW}_{n,u}$ -PCA adversary which simulates $\mathbf{G}_3$. With only the public key $\mathbf{C}$ can simulate PCO and CVO as described in $\mathbf{G}_3$. By construction, $\mathbf{C}$ winning the $\text{OW}_{n,u}$ -PCA is equivalent to $\mathbf{B}$ winning $\mathbf{G}_3$. We have

G₂(B)	PCO(j, m ∈ M, c)
1 : for j = 1, ..., u	11 : return $\llbracket \text{Enc}(pk_j, m; G(m)) = c \rrbracket$
2 : (pk _j , sk _j) $\xleftarrow{\$}$ KeyGen()	CVO(j, c)
3 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	12 : if c ∈ L _{C_j}
4 : m' $\leftarrow B^{\text{Chall}, \text{PCO}, \text{CVO}, G}(\vec{pk})$	13 : abort
5 : return $\llbracket m' \in L_M \rrbracket$	14 : return $\llbracket \exists (m, r) \in L_G \text{ s.t. } \text{Enc}(pk_j, m; r) = c \rrbracket$
Chall(j)	G(m)
6 : m $\xleftarrow{\$} M$	15 : if ∃ r s.t. (m, r) ∈ L _G
7 : L _{M_j} $\leftarrow L_{M_j} \cup \{m\}$	16 : return r
8 : c = Enc(pk _j , m; G(m))	17 : r $\xleftarrow{\$} R$
9 : L _{C_j} $\leftarrow L_{C_j} \cup \{c\}$	18 : L _G $\leftarrow L_G \cup \{(m, r)\}$
10 : return c	19 : return r

Figure 4.6: **G₂** for the proof of Theorem 4.2.2

$$\Pr[\mathbf{G}_3 \Rightarrow 1] = \text{Adv}_{\text{PKE}}^{\text{OW}_{n,u}\text{-CPA}}(\mathbf{C})$$

and by lemma 4.2.1, we have that

$$\Pr[\mathbf{G}_3 \Rightarrow 1] \leq \text{Adv}_{\text{PKE}}^{\text{IND-CPA}}(\mathbf{C}) + \frac{n}{|\mathcal{M}|}$$

D is an IND_{n,u}-CPA adversary against PKE, that is able to win based on the performance of C. Now, consider the advantage of D against the IND_{n,u}-CPA security of PKE. Let BADG be the event that B queries G on m ∈ L_{M_{j,1-b}}. Since L_{M_{j,1-b}} is a uniformly random list of length at most n, we have

$$\Pr[\text{BADG}] = \frac{n \cdot q_G}{|\mathcal{M}|}$$

D looks for random oracle queries containing m₀ ∈ L_{M_{j,0}} or m₁ ∈ L_{M_{j,1}}. If one is found, this indicates that either the flag CHAL or BADG was raised. If CHAL is raised, and BADG is not, then A detects a random oracle query containing m ∈ L_{M_{j,b}}. If neither CHAL or BADG is raised, A returns a random guess, as is the case when both CHAL and BADG are

G₃(B)	PCO(<i>j</i>, <i>m</i> ∈ <i>M</i>, <i>c</i>)
1 : for <i>j</i> = 1, ..., <i>u</i>	11 : return $\llbracket \text{Enc}(pk_j, m; G(m)) = c \rrbracket$
2 : (<i>pk_j</i> , <i>sk_j</i>) $\xleftarrow{\$}$ KeyGen()	CVO(<i>j</i>, <i>c</i>)
3 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	12 : if <i>c</i> ∈ <i>L_{C_j}</i>
4 : <i>m'</i> $\leftarrow B^{\text{Chall}, \text{PCO}, \text{CVO}, G}(\vec{pk})$	13 : abort
5 : return $\llbracket m' \in L_M \rrbracket$	14 : return $\llbracket \exists (m, r) \in L_G \text{ s.t. } \text{Enc}(pk_j, m; r) = c \rrbracket$
Chall(<i>j</i>)	G(<i>m</i>)
6 : <i>m</i> $\xleftarrow{\$} M$	15 : if $\exists r \text{ s.t. } (m, r) \in L_G$
7 : <i>L_{M_j}</i> $\leftarrow L_{M_j} \cup \{m\}$	16 : return <i>r</i>
8 : <i>c</i> = Enc(<i>pk_j</i> , <i>m</i> ; G(<i>m</i>))	17 : if <i>m</i> ∈ <i>L_M</i>
9 : <i>L_{C_j}</i> $\leftarrow L_{C_j} \cup \{c\}$	18 : CHAL = true
10 : return <i>c</i>	19 : abort
	20 : <i>r</i> $\xleftarrow{\$} \mathcal{R}$
	21 : <i>L_G</i> $\leftarrow L_G \cup \{(m, r)\}$
	22 : return <i>r</i>

Figure 4.7: **G₃** for the proof of Theorem 4.2.2

raised. Then provided **BADG** is not raised, **A** wins the game with probability 1 if **CHAL** is raised, and with probability $\frac{1}{2}$ otherwise. Hence

$$\begin{aligned}
\text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\text{D}) + \Pr[\text{BADG}] &\geq \left| \Pr[b = b' | \neg \text{BADG}] - \frac{1}{2} \right| \\
&= \left| \Pr[\text{CHAL}] + \frac{1}{2} \Pr[\neg \text{CHAL}] - \frac{1}{2} \right| \\
&= \frac{1}{2} \Pr[\text{CHAL}]
\end{aligned}$$

Since **C** and **D** are both **CPA** adversaries, we can represent (**C**, **D**) as a single **CPA** adversary **A**. Then we have

$C(\vec{pk})$	$D(\vec{pk})$
1 : $m' \leftarrow B^{\text{Chall}, G, \text{PCO}, \text{CVO}}(\vec{pk})$	1 : $b'' = C^{\text{Chall}, \text{PCO}, \text{CVO}, G}(\vec{pk})$
2 : $'$	2 : if $\exists m_0 \in \mathcal{L}_{M_{i,0}}$ and $((m, r) \in \mathcal{L}_{G_i})$
	3 : if $\exists m_1 \in \mathcal{L}_{M_{j,1}}$ and $((m, r) \in \mathcal{L}_G)$
	4 : $b' \xleftarrow{\$} \{0, 1\}$
	5 : else $b' := 0$
	6 : if $\exists m_1 \in \mathcal{L}_{M_{i,1}}$ and $((m, r) \in \mathcal{L}_{G_i})$
	7 : if $\exists m_0 \in \mathcal{L}_{M_{j,0}}$ and $((m, r) \in \mathcal{L}_G)$
	8 : $b' \xleftarrow{\$} \{0, 1\}$
	9 : else $b' := 1$
	10 : $b' \xleftarrow{\$} \{0, 1\}$
	11 : return $\llbracket b = b' \rrbracket$

Figure 4.8: Adversaries C and D for the proof of Theorem 4.2.2

$$\text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCVA}}(\mathbf{B}) \leq (q_G + q_P) \cdot \delta + q_V \cdot 2^{-\gamma} + \frac{n \cdot (2q_G + n)}{|\mathcal{M}|} + 3 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$$

as desired. $\square$

In the bound for Theorem 4.2.2, an especially problematic term for eFrodoKEM is $\frac{n \cdot (2q_G + n)}{|\mathcal{M}|}$. With a message length of 128 bits, 2^{64} challenge ciphertexts, and 2^{64} hash queries, this term is 1.

4.2.2 The ST-transform: IND_{n,u}-CPA to OW_{n,u}-PCVA tightly

In the Theorem 4.2.3, we introduce salting, which reduces the problematic term significantly.

Theorem 4.2.3 (PKE IND_{n,u}-CPA $\xrightarrow{\text{ROM}}$ ST[PKE, G, len_{salt}] OW_{n,u}-PCVA). *Suppose PKE is $\delta(u)$ -correct and γ -spread. Then ST[PKE] is $\delta(u)$ -correct in the random oracle model.*

Let $\mathcal{L}_S$ represent the list of all salts corresponding to challenge ciphertexts. For any $\text{OW}_{n,u}\text{-PCVA}$ -adversary $\mathbf{B}$ issuing at most q_G queries to the random oracle $\mathbf{G}$, q_P queries to a plaintext checking oracle PCO , and q_V queries to a ciphertext validity oracle CVO , there exists an $\text{IND}_{n,u}\text{-CCA}$ adversary $\mathbf{A}$ such that

$$\text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCVA}}(\mathbf{B}) \leq (q_G + q_P) \cdot \delta + q_V \cdot 2^{-\gamma} + \frac{n \cdot (2q_G + 1)}{|\mathcal{M}| |\mathcal{L}_S|} + 3 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$$

where $\mathbf{A}$ and $\mathbf{B}$ have similar run-time.

Proof. The correctness bound is trivial. To prove the security claim, consider the sequence of games starting from Figure 4.9. Lines in red represent changes from the previous game.

$\mathbf{G}_0(\mathbf{B})$	$\text{PCO}(j, m \in \mathcal{M}, c \parallel \text{salt})$
1 : for $j = 1, \dots, u$	12 : $m' = \text{Dec}(sk_j, c \parallel \text{salt})$
2 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	13 : return $\llbracket m = m' \text{ and } \text{Enc}(pk_j, m'; \mathbf{G}(j, m' \parallel \text{salt})) = c \rrbracket$
3 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	$\text{CVO}(j, c \parallel \text{salt})$
4 : $m' \leftarrow \mathbf{B}^{\text{Chall}, \text{PCO}, \text{CVO}, \mathbf{G}}(\vec{pk})$	14 : if $c \in \mathcal{L}_{C_j}$
5 : return $\llbracket m' \in \mathcal{L}_M \rrbracket$	15 : abort
$\text{Chall}(j)$	16 : $m' \leftarrow \text{Dec}(sk_j, c)$
6 : $m \xleftarrow{\$} \mathcal{M}$	17 : return $\llbracket m' \in \mathcal{M} \text{ s.t. } \text{Enc}(pk_j, m'; \mathbf{G}(j, m' \parallel \text{salt})) = c \rrbracket$
7 : $\text{salt} \xleftarrow{\$} \{0, 1\}^{\text{len}_{\text{salt}}}$	$\mathbf{G}(j, m \parallel \text{salt})$
8 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m \parallel \text{salt}\}$	18 : if $\exists r \text{ s.t. } (m \parallel \text{salt}, r) \in \mathcal{L}_G$
9 : $c \leftarrow \text{Enc}(pk_j, m; \mathbf{G}(j, m \parallel \text{salt}))$	19 : return r
10 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \parallel \text{salt}\}$	20 : $r \xleftarrow{\$} \mathcal{R}$
11 : return c	21 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \parallel \text{salt}, r)\}$
	22 : return r

Figure 4.9: $\mathbf{G}_0$ for the proof of Theorem 4.2.3

Since $\mathbf{G}_0$ is the $\text{OW}_{n,u}\text{-PCVA}$ game we have

$$|\Pr[\mathbf{G}_0 \Rightarrow 1]| = \text{Adv}_{\text{ST}[\text{PKE}]}^{\text{OW}_{n,u}\text{-PCVA}}(\mathbf{B})$$

$\mathbf{G}_1(\mathbf{B})$	$\text{PCO}(j, m \in \mathcal{M}, c \parallel \text{salt})$
1 : $(pk, sk) \xleftarrow{\$} \text{KeyGen}()$	13 : $m' \leftarrow \text{Dec}(sk_j, c)$
2 : for $j = 1, \dots, u$	14 : return $\llbracket m = m' \text{ and } \text{Enc}(pk_j, m'; \mathbf{G}(j, m' \parallel \text{salt})) = c \rrbracket$
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	$\text{CVO}(j, c \parallel \text{salt})$
4 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	15 : if $c \in \mathcal{L}_{C_j}$
5 : $m' \leftarrow \mathbf{B}^{\text{Chall}, \text{PCO}, \text{CVO}, \mathbf{G}}(\vec{pk})$	16 : abort
6 : return $\llbracket m' \in \mathcal{L}_M \rrbracket$	17 : $m' \leftarrow \text{Dec}(sk_j, c \parallel \text{salt})$
$\text{Chall}(j)$	18 : return $\llbracket (m' \parallel \text{salt}, r) \in \mathcal{L}_{\mathbf{G}} \text{ and } \text{Enc}(pk_j, m'; r \parallel \text{salt}) = c \rrbracket$
7 : $m \xleftarrow{\$} \mathcal{M}$	$\mathbf{G}(j, m \parallel \text{salt})$
8 : $\text{salt} \xleftarrow{\$} \{0, 1\}^{\text{len}_{\text{salt}}}$	19 : if $\exists r \text{ s.t. } (m \parallel \text{salt}, r) \in \mathcal{L}_{\mathbf{G}}$
9 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m \parallel \text{salt}\}$	20 : return r
10 : $c \leftarrow \text{Enc}(pk_j, m; \mathbf{G}(j, m \parallel \text{salt}))$	21 : $r \xleftarrow{\$} \mathcal{R}$
11 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \parallel \text{salt}\}$	22 : $\mathcal{L}_{\mathbf{G}} \leftarrow \mathcal{L}_{\mathbf{G}} \cup \{(m \parallel \text{salt}, r)\}$
12 : return c	23 : return r

Figure 4.10: $\mathbf{G}_1$ for the proof of Theorem 4.2.3

In $\mathbf{G}_1$, the only change is to the oracle CVO which now returns 1 if and only if there was a previous query $r \leftarrow \mathbf{G}(j, m')$ such that $\text{Enc}(pk_j, m'; r) = c$. Consider a query $\text{CVO}(j, c)$ with $m' \leftarrow \text{Dec}(sk_j, c)$.

Case 1: $\text{CVO}(j, c) = 1$ in $\mathbf{G}_1$, then $(m', r) \in \mathcal{L}_{\mathbf{G}}$, and $\text{Enc}(pk_j, m'; r) = c$, hence $\mathbf{G}(j, m') = r$ and $\text{Enc}(pk_j, m'; \mathbf{G}(j, m')) = c$. Thus $\text{CVO}(c) = 1$ in $\mathbf{G}_0$ as well.

Case 2: $\text{CVO}(j, c) = 0$ in $\mathbf{G}_1$. Here, we consider 2 sub-cases. Either $\text{CVO}(j, c) = 0$ in $\mathbf{G}_0$, in which case $\mathbf{G}_1$ and $\mathbf{G}_0$ are identical, or $\text{CVO}(j, c) = 1$ in $\mathbf{G}_0$. In the latter case, we have that $\text{Enc}(pk_j, m'; \mathbf{G}(j, m')) = c$, and so for $\text{CVO}(c) = 1$ in $\mathbf{G}_1$, it must be the case that $\mathbf{G}(j, m')$ has not been queried before. This means that the adversary was able to find r such that $\text{Enc}(pk_j, m; r) = \text{Enc}(pk_j, m; \mathbf{G}(m))$ and $\mathbf{G}(m) \neq r$.

This is where we make use of the assumption that PKE is γ -spread, and so

$$\Pr_{r \in \mathcal{R}}[\text{Enc}(pk, m; r) = c] \leq 2^{-\gamma}.$$

Then the probability that $\text{Enc}(pk_j, m'; r \parallel \text{salt}) = \text{Enc}(pk_j, m'; G(j, m' \parallel \text{salt}))$, is bounded by $2^{-\gamma}$. The $\mathbf{G}_0$ and $\mathbf{G}_1$ only differ in this case, and since there are q_V total CVO queries, we have

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq q_V \cdot 2^{-\gamma}$$

$\mathbf{G}_2(\mathbf{B})$	$\text{PCO}(j, m \in \mathcal{M}, c \parallel \text{salt})$
1 : for $j = 1, \dots, u$	12 : return $\llbracket \text{Enc}(pk_j, m; G(j, m \parallel \text{salt})) = c \rrbracket$
2 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	$\text{CVO}(j, c \parallel \text{salt})$
3 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	13 : if $c \in \mathcal{L}_{C_j}$
4 : $m' \leftarrow \mathbf{B}^{\text{Chall}, \text{PCO}, \text{CVO}, \mathbf{G}}(\vec{pk})$	14 : abort
5 : return $\llbracket m' \in \mathcal{L}_M \rrbracket$	15 : return $\llbracket \exists (m \parallel \text{salt}, r) \in \mathcal{L}_G \text{ s.t. } \text{Enc}(pk_j, m; r \parallel \text{salt}) = c \rrbracket$
$\text{Chall}(j)$	$G(j, m \parallel \text{salt})$
6 : $m \xleftarrow{\$} \mathcal{M}$	16 : if $\exists r \text{ s.t. } (m \parallel \text{salt}, r) \in \mathcal{L}_G$
7 : $\text{salt} \xleftarrow{\$} \{0, 1\}^{\text{len}_{\text{salt}}}$	17 : return r
8 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m \parallel \text{salt}\}$	18 : $r \xleftarrow{\$} \mathcal{R}$
9 : $c \leftarrow \text{Enc}(pk_j, m; G(j, m \parallel \text{salt}))$	19 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \parallel \text{salt}, r)\}$
10 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \parallel \text{salt}\}$	20 : return r
11 : return c	

Figure 4.11: $\mathbf{G}_2$ for the proof of Theorem 4.2.3

In $\mathbf{G}_2$, the PCO and CVO oracles no longer check whether $m = \text{Dec}(sk_j, c)$. In addition to the q_G $\mathbf{G}$ queries, each of the q_P PCO queries also make an implicit query to $\mathbf{G}$, for a maximum total of $q_G + q_P$ total $G(j, m_i \parallel \text{salt})$. Observe that $\mathbf{G}_2$ and $\mathbf{G}_1$ proceed identically, so long as $\text{Dec}(sk, \text{Enc}(pk, m; G(m_i \parallel \text{salt}))) = m_i$ for all i . Hence they only differ when $\mathbf{B}$ submits a query to PCO or CVO together with an m such that $G(j, m \parallel \text{salt})$ exhibits a correctness error, where $\mathbf{G}_1$ outputs 0 and $\mathbf{G}_2$ outputs 1. By $\delta(u)$ correctness, the probability that at least one $G(j, m_i \parallel \text{salt})$ exhibits a correctness error is bounded by $(q_G + q_P) \cdot \delta(u)$. Hence we have

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq (q_P + q_G) \cdot \delta(u)$$

$\mathbf{G}_3(\mathbf{B})$ <hr/> 1 : for $j = 1, \dots, u$ 2 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$ 3 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$ 4 : $m' \leftarrow \mathbf{B}^{\text{Chall}, \text{PCO}, \text{CVO}, \mathbf{G}}(\vec{pk})$ 5 : return $\llbracket m' \in \mathcal{L}_M \rrbracket$	$\text{PCO}(j, m \in \mathcal{M}, c \parallel \text{salt})$ <hr/> 12 : return $\llbracket \text{Enc}(pk_j, m; \mathbf{G}(j, m \parallel \text{salt})) = c \rrbracket$
$\text{Chall}(j)$ <hr/> 6 : $m \xleftarrow{\$} \mathcal{M}$ 7 : $\text{salt} \xleftarrow{\$} \{0, 1\}^{\text{len}_{\text{salt}}}$ 8 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m \parallel \text{salt}\}$ 9 : $c \leftarrow \text{Enc}(pk_j, m; \mathbf{G}(j, m \parallel \text{salt}))$ 10 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \parallel \text{salt}\}$ 11 : return c	$\text{CVO}(j, c \parallel \text{salt})$ <hr/> 13 : if $c \in \mathcal{L}_{C_j}$ 14 : abort 15 : return $\llbracket \exists (m \parallel \text{salt}, r) \in \mathcal{L}_G \text{ and } \text{Enc}(pk_j, m; r \parallel \text{salt}) = c \rrbracket$
	$\mathbf{G}(j, m \parallel \text{salt})$ <hr/> 16 : if $\exists r$ s.t. $(m \parallel \text{salt}, r) \in \mathcal{L}_G$ 17 : return r 18 : if $m \parallel \text{salt} \in \mathcal{L}_M$ 19 : $\text{CHAL} = \text{true}$ 20 : abort 21 : $r \xleftarrow{\$} \mathcal{R}$ 22 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \parallel \text{salt}, r)\}$ 23 : return r

Figure 4.12: $\mathbf{G}_3$ for the proof of Theorem 4.2.3

$\mathbf{G}_3$ and $\mathbf{G}_2$ differ only when the flag CHAL is raised, and so

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \Pr[\text{CHAL}]$$

by the difference lemma. Now we establish two bounds, first we bound $\Pr[\mathbf{G}_3 \Rightarrow 1]$, by constructing an adversary $\mathbf{C}$ that against the $\text{OW}_{n,u}$ -CPA-security of PKE, then we bound $\Pr[\text{CHAL}]$ to complete the proof. With only the public key $\mathbf{C}$ can simulate PCO and CVO as described in $\mathbf{G}_3$. By construction, $\mathbf{C}$ winning the $\text{OW}_{n,u}$ -PCA is equivalent to $\mathbf{B}$ winning $\mathbf{G}_3$. We have

$$\Pr[\mathbf{G}_3 \Rightarrow 1] = \text{Adv}_{\text{PKE}}^{\text{OW}_{n,u}\text{-CPA}}(\mathbf{C})$$

and by Lemma 4.2.1, we have that

$C(\vec{pk})$	$D(\vec{pk})$
1 : $m' \leftarrow B^{\text{Chall}, G, \text{PCO}, \text{CVO}}(\vec{pk})$	1 : $b'' = C^{\text{Chall}, \text{PCO}, \text{CVO}, G}(\vec{pk})$
2 : return m'	2 : if $\exists(m_0 \parallel \text{salt}) \in \mathcal{L}_{M_{i,0}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_{G_i})$
	3 : if $\exists(m_1 \parallel \text{salt}) \in \mathcal{L}_{M_{j,1}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_G)$
	4 : $b' \xleftarrow{\$} \{0, 1\}$
	5 : else $b' := 0$
	6 : if $\exists(m_1 \parallel \text{salt}) \in \mathcal{L}_{M_{i,1}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_{G_i})$
	7 : if $\exists(m_0 \parallel \text{salt}) \in \mathcal{L}_{M_{j,0}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_G)$
	8 : $b' \xleftarrow{\$} \{0, 1\}$
	9 : else $b' := 1$
	10 : else $b' \xleftarrow{\$} \{0, 1\}$
	11 : return $\llbracket b = b' \rrbracket$

Figure 4.13: Adversaries C and D against the $\text{OW}_{n,u}\text{-CPA}$ -security of PKE for the proof of Theorem 4.2.3

$$\Pr[\mathbf{G}_3 \Rightarrow 1] \leq \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{C}) + \frac{n}{|\mathcal{M}|}$$

D is an $\text{IND}_{n,u}\text{-CPA}$ adversary against PKE that is able to win based on the performance of C. Now, consider the advantage of D against the $\text{IND}_{n,u}\text{-CPA}$ security of PKE. Let BADG be the event that B queries G on $m \in \mathcal{L}_{M_{j,1-b}}$. Since $\mathcal{L}_{M_{j,1-b}}$ is a uniformly random list of length at most n , we have

$$\Pr[\text{BADG}] = \max_j \frac{n_j \cdot q_G}{|\mathcal{M} \parallel \mathcal{L}_{S_j}|}$$

where $\mathcal{L}_{S_j}$ represents the list of all distinct salts for user j , and n_j represents the number of queries to $\text{Chall}(j)$ for user j .

Furthermore, we claim that

$$\max_j \frac{n_j \cdot q_G}{|\mathcal{M} \parallel \mathcal{L}_{S_j}|} \leq \frac{n}{|\mathcal{M} \parallel \mathcal{L}_S|}.$$

Consider

$$\max_{j \in [u]} \frac{n_j}{|\mathfrak{L}_{S_j}|}.$$

The expected number of distinct salt values for n_j queries to **Chall** can be obtained by the formula for the expected number of distinct results, when drawing n_j samples from a set without replacement:

$$\mathbb{E}(|\mathfrak{L}_{S_j}|) = 2^{\text{len}_{\text{salt}}} \left(1 - \left(1 - \frac{1}{2^{\text{len}_{\text{salt}}}} \right)^{n_j} \right).$$

Fix $\text{len}_{\text{salt}} = \lfloor \log_2(n_j) \rfloor$. The above formula yields

$$\max_{j \in [u]} \frac{n_j}{\mathbb{E}(|\mathfrak{L}_{S_j}|)} \approx \frac{1}{\left(1 - \left(1 - \frac{1}{2^{\text{len}_{\text{salt}}}} \right) \right)^{n_j}}.$$

which increases as $n_j \rightarrow n$.

D looks for random oracle queries containing $m_0 \in \mathfrak{L}_{M_{j,0}}$ or $m_1 \in \mathfrak{L}_{M_{j,1}}$. If one is found, this indicates that either the flag **CHAL** or **BADG** was raised. If **CHAL** is raised, and **BADG** is not, then A detects a random oracle query containing $m \in \mathfrak{L}_{M_{j,b}}$. If neither **CHAL** or **BADG** is raised, A returns a random guess, as is the case when both **CHAL** and **BADG** are raised. Then provided **BADG** is not raised, A wins the game with probability 1 if **CHAL** is raised, and with probability $\frac{1}{2}$ otherwise. Hence

$$\begin{aligned} \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\text{D}) + \Pr[\text{BADG}] &\geq \left| \Pr[b = b' | \neg \text{BADG}] - \frac{1}{2} \right| \\ &= \left| \Pr[\text{CHAL}] + \frac{1}{2} \Pr[\neg \text{CHAL}] - \frac{1}{2} \right| \\ &= \frac{1}{2} \Pr[\text{CHAL.}] \end{aligned}$$

Since C and D are both CPA adversaries, we can represent them as a single CPA adversary A. Then we have

$$\text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCVA}}(\text{B}) \leq (q_G + q_P) \cdot \delta + q_V \cdot 2^{-\gamma} + \frac{n \cdot (2q_G + n)}{|\mathcal{M}| |\mathfrak{L}_S|} + 3 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\text{A})$$

as desired. $\square$

Now we briefly consider what impact incorporating salts has made to the security bound. Note that the length of salts should be $\lceil \log_2(n) \rceil$. Then $n \leq 2^{\text{len}_{\text{salt}}}$. If $\text{len}_{\text{salt}} = 64$, and $n = 2^{64}$, and $|\mathcal{M}| \approx 2^{128}$ then $\text{Adv}_{\text{KEM}}^{\text{IND}_{n,u}\text{-CCA}}(\mathcal{B}) \approx 1$ when $\text{KEM} = \text{T}[\text{PKE}, \mathcal{G}]$. With these same parameters, $\mathbb{E}(|\mathcal{L}_S|) \approx 0.63 \cdot 2^{64} > \frac{n}{2}$. Recall that in the bound shown in Theorem 4.2.2 there is a problematic term $\frac{n \cdot (2q_{\mathcal{G}} + 1)}{|\mathcal{M}|}$. For messages of bit length 128, and 2^{64} challenges, this value greater than 1, trivializing the upper bound. When $\text{KEM} = \text{ST}[\text{PKE}, \mathcal{G}, \text{len}_{\text{salt}} = 64]$ the corresponding term is $\frac{n \cdot (2q_{\mathcal{G}} + 1)}{|\mathcal{M}||\mathcal{L}_S|}$, a reduction by a factor greater than 2^{63} .

4.2.3 The $\mathcal{U}^\chi$ -transform: $\text{OW}_{n,u}$ -PCVA to $\text{IND}_{n,u}$ -CCA with implicit rejection

In this section we consider the bounds relating to the 4 variants of the U transform defined in [HHK17]. By changing the T transform to ST, you also change the hash functions, encapsulation and decapsulation oracles accordingly. This change is demonstrated in the differences between the proof of Theorem 4.2.5, in which PKE_1 is the result of applying the T transform to PKE, and Theorem 4.2.6, in which ST is used.

Intuitive overview: the following proofs involve a sequence of game hops. One aspect of the proof strategy is to modify the oracle $\mathcal{O}_{\text{Decaps}^\chi}$ so that it can be simulated by a PCA/PCVA adversary. Another aspect of the proof is to raise the flag CHAL to indicate that the adversary has queried a message underlying a challenge ciphertext, and to equate this event with an adversarial goal for the $\text{OW}_{n,u}$ -PCA/ $\text{OW}_{n,u}$ -PCVA game. Finally, the games are modified so that in the final game, the adversary can do no better than making a random guess, by making their view independent of the bit b .

Theorem 4.2.4 ($\text{T}[\text{PKE}, \mathcal{G}] \text{ OW}_{n,u}\text{-PCA} \xrightarrow{\text{ROM}} \text{FO}^\chi[\text{PKE}, \mathcal{G}, \mathcal{H}] \text{ IND}_{n,u}\text{-CCA}$). *If $\text{PKE}_1 := \text{T}[\text{PKE}, \mathcal{G}]$ is δ -correct, then $\text{KEM}^\chi := \mathcal{U}_{m,c}^\chi[\text{PKE}_1, \mathcal{H}]$ is δ -correct in the random oracle model. Moreover for any $\text{IND}_{n,u}$ -CCA adversary $\mathcal{B}$ against KEM^χ , issuing at most q_D queries to the decapsulation oracle $\mathcal{O}_{\text{Decaps}^\chi}$ and at most q_H queries to the random oracle $\mathcal{H}$ and q_G queries to the random oracle $\mathcal{G}$, there exists an $\text{OW}_{n,u}$ -PCA adversary $\mathcal{A}$ against PKE_1 issuing at most q_H queries to the PCO oracle such that*

$$\text{Adv}_{\text{KEM}^\chi}^{\text{IND}_{n,u}\text{-CCA}}(\mathcal{B}) \leq \frac{q_H + n^2}{|\mathcal{M}|} + \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\mathcal{A})$$

where A and B have similar run-time.

Proof. The proof consists of a sequence of games, following the technique shown in [Sho04] and an analysis of the adversary B 's advantage in each, adapted from [HHK17] and [DHK⁺21], with careful consideration of the impact of adding uniformly random salts.

The correctness bound is trivial. To prove the security claim, consider the sequence of games starting from Figure 4.2.4, which is simply the $\text{IND}_{n,u}\text{-CCA}$ game against $\text{FO}_{m,c}^\times[\text{PKE}, G, H]$.

$G_0(B)$	$H(j, m, c)$
1 : $b \xleftarrow{\$} \{0, 1\}$	16 : if $\exists \text{ss}$ s.t. $(m, c, \text{ss}) \in \mathcal{L}_{H_j}$
2 : for $j = 1, \dots, u$	17 : return ss
3 : $(pk_j, sk'_j) \xleftarrow{\$} \text{KeyGen}()$	18 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
4 : $s_j \xleftarrow{\$} \mathcal{M}$	19 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(m, c, \text{ss})\}$
5 : $sk_j \leftarrow (sk'_j, s_j)$	20 : return ss
6 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	$G(m)$
7 : $b' \leftarrow B^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, H, G}(\vec{pk})$	17 : if $\exists r$ s.t. $(m, r) \in \mathcal{L}_G$
8 : return $\llbracket b = b' \rrbracket$	18 : return r
$\text{Chall}(j)$	19 : $r \xleftarrow{\$} \mathcal{R}$
9 : $m \xleftarrow{\$} \mathcal{M}$	20 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m, r)\}$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	21 : return r
11 : $c \leftarrow \text{Enc}(pk_j, m; G(m))$	$\mathcal{O}_{\text{Decaps}^\times}(j, c \notin \mathcal{L}_{C_j})$
12 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c\}$	21 : $m' \leftarrow \text{Dec}(sk'_j, c)$
13 : $\text{ss}_0 \leftarrow H(j, m, c)$	22 : if $m' = \perp$ or $c \neq \text{Enc}(pk_j, m'; G(m'))$
14 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	23 : return $\text{ss} \leftarrow H(j, s, c)$
15 : return (c, ss_b)	24 : return $\text{ss} \leftarrow H(j, m', c)$

Figure 4.14: G_0 for the proof of Theorem 4.2.4

G_0 is the $\text{IND}_{n,u}\text{-CCA}$ game against $\text{FO}_{m,c}^\times[\text{PKE}, G, H]$, and so

$$\left| \Pr[G_0 \Rightarrow 1] - \frac{1}{2} \right| = \text{Adv}_{\text{KEM}^\times}^{\text{IND}_{n,u}\text{-CCA}}(B)$$

$\mathbf{G}_1(\mathbf{B})$	$\mathbf{H}(j, m, c)$
1 : $b \xleftarrow{\$} \{0, 1\}$	16 : if $\exists \mathbf{ss}$ s.t. $(m, c, \mathbf{ss}) \in \mathcal{L}_{\mathbf{H}_j}$
2 : for $j = 1, \dots, u$	17 : return $\mathbf{ss}$
3 : $(pk_j, sk'_j) \xleftarrow{\$} \text{KeyGen}()$	18 : $\mathbf{ss} \xleftarrow{\$} \mathcal{K}$
4 : $s_j \xleftarrow{\$} \mathcal{M}$	19 : $\mathcal{L}_{\mathbf{H}_j} \leftarrow \mathcal{L}_{\mathbf{H}_j} \cup \{(m, c, \mathbf{ss})\}$
5 : $sk_j \leftarrow (sk'_j, s_j)$	20 : return $\mathbf{ss}$
6 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	$\mathbf{G}(m)$
7 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, \mathbf{H}, \mathbf{G}}(\vec{pk})$	17 : if $\exists r$ s.t. $(m, r) \in \mathcal{L}_{\mathbf{G}}$
8 : return $\llbracket b = b' \rrbracket$	18 : return r
$\text{Chall}(j)$	19 : $r \xleftarrow{\$} \mathcal{R}$
9 : $m \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_j}$	20 : $\mathcal{L}_{\mathbf{G}} \leftarrow \mathcal{L}_{\mathbf{G}} \cup \{(m, r)\}$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	21 : return r
11 : $c \leftarrow \text{Enc}(pk_j, m; \mathbf{G}(m))$	$\mathcal{O}_{\text{Decaps}^\times}(j, c \notin \mathcal{L}_{C_j})$
12 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c\}$	21 : $m' \leftarrow \text{Dec}(sk'_j, c)$
13 : $\mathbf{ss}_0 \leftarrow \mathbf{H}(j, m, c)$	22 : if $m' = \perp$ or $c \neq \text{Enc}(pk_j, m'; \mathbf{G}(m'))$
14 : $\mathbf{ss}_1 \xleftarrow{\$} \mathcal{K}$	23 : return $\mathbf{ss} \leftarrow \mathbf{H}(j, s, c)$
15 : return $(c, \mathbf{ss}_b)$	24 : return $\mathbf{ss} \leftarrow \mathbf{H}(j, m', c)$

Figure 4.15: $\mathbf{G}_1$ for the proof of Theorem 4.2.4

In $\mathbf{G}_1$, challenge messages are sampled without replacement. This change is necessary because if $m_i = m_j$ for $i \neq j$, then $c_i = c_j$ due to the fact that encryption is deterministic. Then if $b = 0$, $\mathbf{ss}_0 = \mathbf{ss}_1$ while whenever $b = 1$ $\mathbf{ss}_0$ and $\mathbf{ss}_1$ are independent, which trivializes the $\text{IND}_{n,u}\text{-CCA}$ game. By the birthday bound and the difference lemma we have that

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq \frac{n^2}{|\mathcal{M}|}.$$

In $\mathbf{G}_2$ we introduce two changes. If $\mathbf{H}$ is queried on s_j , the flag QUERY is raised, and the protocol is aborted. The other change is that decapsulation queries are made perfectly random when an input is rejected, making use of a separate internal random oracle $\mathbf{H}'$. We observe that $\mathbf{G}_2$ and $\mathbf{G}_1$ are identical unless the adversary queries $\mathbf{H}$ on s_j , which is a uniformly secret value. Thus by the difference lemma

G₂(B)	H(j, m, c)
1 : $b \xleftarrow{\$} \{0, 1\}$	16 : if $\exists \text{ss s.t. } (m, c, \text{ss}) \in \mathfrak{L}_{H_j}$
2 : for $j = 1, \dots, u$	17 : return ss
3 : $(pk_j, sk'_j) \xleftarrow{\$} \text{KeyGen}()$	18 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
4 : $s_j \xleftarrow{\$} \mathcal{M}$	19 : if $m = s_j$
5 : $sk_j \leftarrow (sk'_j, s_j)$	20 : QUERY = true; abort
6 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	21 : $\mathfrak{L}_{H_j} \leftarrow \mathfrak{L}_{H_j} \cup \{(m, c, \text{ss})\}$
7 : $b' \leftarrow \text{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, \text{H}, \text{G}}(\vec{pk})$	22 : return ss
8 : return $\llbracket b = b' \rrbracket$	G(m)
Chall(j)	23 : if $\exists r \text{ s.t. } (m, r) \in \mathfrak{L}_G$
9 : $m \xleftarrow{\$} \mathcal{M} \setminus \mathfrak{L}_{M_j}$	24 : return r
10 : $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m\}$	25 : $r \xleftarrow{\$} \mathcal{R}$
11 : $c \leftarrow \text{Enc}(pk_j, m; \text{G}(m))$	26 : $\mathfrak{L}_G \leftarrow \mathfrak{L}_G \cup \{(m, r)\}$
12 : $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c\}$	27 : return r
13 : $\text{ss}_0 \leftarrow \text{H}(j, m, c)$	$\mathcal{O}_{\text{Decaps}^\times}(j, c \notin \mathfrak{L}_{C_j})$
14 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	28 : $m' \leftarrow \text{Dec}(sk'_j, c)$
15 : return (c, ss_b)	29 : if $m' = \perp$ or $c \neq \text{Enc}(pk_j, m'; \text{G}(m'))$
	30 : return $\text{ss} \leftarrow \text{H}(j, s, c)$
	31 : return $\text{ss} \leftarrow \text{H}(j, m', c)$

Figure 4.16: **G₂** for the proof of Theorem 4.2.4

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \frac{q_H}{|\mathcal{M}|}$$

$G_3(B)$	$H(j, m, c)$
1 : $b \xleftarrow{\$} \{0, 1\}$	16 : if $\exists ss$ s.t. $(m, c, ss) \in \mathcal{L}_{H_j}$
2 : for $j = 1, \dots, u$	17 : return ss
3 : $(pk_j, sk'_j) \xleftarrow{\$} \text{KeyGen}()$	18 : $ss \xleftarrow{\$} \mathcal{K}$
4 : $s_j \xleftarrow{\$} \mathcal{M}$	19 : if $m = s_j$
5 : $sk_j \leftarrow (sk'_j, s_j)$	20 : $\text{QUERY} = \text{true}; \text{abort}$
6 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	21 : if $\text{Dec}(sk'_j, c) = m$ and $\text{Enc}(pk_j, m; G(m)) = c$
7 : $b' \leftarrow B^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, H, G}(\vec{pk})$	22 : if $\exists ss' \text{ s.t. } (c, ss') \in \mathcal{L}_{D_j}$
8 : return $\llbracket b = b' \rrbracket$	23 : $ss := ss'$
$\text{Chall}(j)$	24 : else
9 : $m \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_j}$	25 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c, ss\}$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	26 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(m, c, ss)\}$
11 : $c \leftarrow \text{Enc}(pk_j, m; G(m))$	27 : return ss
12 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c\}$	$\mathcal{O}_{\text{Decaps}^\times}(j, c \notin \mathcal{L}_{C_j})$
13 : $ss_0 \leftarrow H(j, m, c)$	28 : if $\exists ss$ s.t. $(c, ss) \in \mathcal{L}_{D_j}$
14 : $ss_1 \xleftarrow{\$} \mathcal{K}$	29 : return ss
15 : return (c, ss_b)	30 : else
	31 : $ss \xleftarrow{\$} \mathcal{K}$
	32 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c, ss\}$
	33 : return ss

Figure 4.17: G_3 for the proof of Theorem 4.2.4

In Figure 4.17 we omit G , since it doesn't change from Figure 4.16, and would cause the figure to be too large to fit on the page. In G_3 lists $\mathcal{L}_{D_j}$ and $\mathcal{L}_{H_j}$ are introduced for book-keeping, and the oracles are modified so as to no longer make use of the secret keys, except to test if $\text{Dec}_1(sk'_j, c) = m$ on line 27. In G_3 and G_2 the view of the adversary is identical. To see this, consider a fixed plaintext ciphertext pair $(c, m' \leftarrow \text{Dec}_1(sk'_j, c))$. Consider the following cases

Case 1: $m \in \{\perp, s\}$.

in $\mathbf{G}_3$ $\mathbf{H}$ takes as input a message $m \in \mathcal{M}$, and $c \in \mathcal{C}$. $\mathbf{H}(j, \perp, c)$ cannot be queried. Querying $\mathbf{H}(j, s, c)$ raises the flag QUERY on line 25. Hence querying $\mathbf{H}$ can never append a tuple $(c, \mathbf{ss})$ to the list $\mathfrak{L}_{D_j}$. $\mathcal{O}_{\text{Decaps}^\times}(j, c)$, the output will be a uniformly random value $\mathbf{ss}$.

Case 2: $m \notin \{\perp, s\}$

We aim to show that $\mathcal{O}_{\text{Decaps}^\times}(j, c)$ returns $\mathbf{H}(j, m', c)$ for valid ciphertexts with $\text{Dec}_1(sk'_j, c) = m'$. There are two possibilities here: $\mathbf{H}(j, m', c)$ is queried by $\mathbf{B}$ first, then $\mathcal{O}_{\text{Decaps}^\times}(j, c)$, or $\mathcal{O}_{\text{Decaps}^\times}(j, c)$ is queried first, then $\mathbf{H}(j, m', c)$.

Suppose $\mathbf{B}$ queries $\mathbf{H}(j, m', c)$ first. Then on line 31, since $\mathcal{O}_{\text{Decaps}^\times}$ has not yet been queried on c , $(c, \mathbf{ss})$ will be appended to $\mathfrak{L}_{D_j}$. Then on line 32 $(m, c, \mathbf{ss})$ will be appended to $\mathfrak{L}_{H_j}$ and $\mathbf{H}(j, m, c)$ will output $\mathbf{ss}$. Then when $\mathcal{O}_{\text{Decaps}^\times}(j, c)$ is queried, line 34 will recognize $(c, \mathbf{ss}) \in \mathfrak{L}_{D_j}$ and $\mathcal{O}_{\text{Decaps}^\times}(j, c)$ will return $\mathbf{ss}$.

Now suppose that $\mathbf{B}$ first queries $\mathcal{O}_{\text{Decaps}^\times}(j, c)$. Then $(c, \mathbf{ss})$ will be appended to $\mathfrak{L}_{D_j}$. When $\mathbf{H}$ is queried, line 28 will recognize $(c, \mathbf{ss}) \in \mathfrak{L}_{D_j}$, then $(m, c, \mathbf{ss})$ will be appended to $\mathfrak{L}_{H_j}$, and $\mathbf{H}(j, m', c)$ will output $\mathbf{ss}$.

Hence, in $\mathbf{G}_3$ and $\mathbf{G}_2$, the view of $\mathbf{B}$ is identical, so

$$\Pr[\mathbf{G}_3 \Rightarrow 1] = \Pr[\mathbf{G}_2 \Rightarrow 1].$$

$\mathbf{G}_4(\mathbf{B})$	$\mathbf{H}(j, m, c)$
1 : $b \xleftarrow{\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk'_j) \xleftarrow{\$} \text{KeyGen}()$ 4 : $s_j \xleftarrow{\$} \mathcal{M}$ 5 : $sk_j \leftarrow (sk'_j, s_j)$ 6 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$ 7 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, \mathbf{H}, \mathbf{G}}(\vec{pk})$ 8 : return $\llbracket b = b' \rrbracket$	21 : if $\exists \text{ss s.t. } (m, c, \text{ss}) \in \mathcal{L}_{\mathbf{H}_j}$ 22 : return ss 23 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 24 : if $m = s_j$ 25 : $\text{QUERY} = \text{true}$ 26 : abort 27 : if $\text{Dec}(sk'_j, c) = m$ and $\text{Enc}(pk_j, m; \mathbf{G}(m)) = c$ 28 : if $c \in \mathcal{L}_{C_j}$ 29 : $\text{CHAL} = \text{true}$ 30 : abort 31 : if $\exists \text{ss}' \text{ s.t. } (c, \text{ss}') \in \mathcal{L}_{D_j}$ 32 : $\text{ss} \leftarrow \text{ss}'$ 33 : else 34 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c, \text{ss}\}$ 35 : $\mathcal{L}_{\mathbf{H}_j} \leftarrow \mathcal{L}_{\mathbf{H}_j} \cup \{(m, c, \text{ss})\}$ 36 : return ss
Chall (j) 9 : $m \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_j}$ 10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$ 11 : $c \leftarrow \text{Enc}(pk_j, m; \mathbf{G}(m))$ 12 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c\}$ 13 : $\text{ss}_0 \xleftarrow{\$} \mathcal{K}$ 14 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$ 15 : return (c, ss_b)	$\mathcal{O}_{\text{Decaps}^\times}(j, c \notin \mathcal{L}_{C_j})$ 34 : if $\exists \text{ss s.t. } (c, \text{ss}) \in \mathcal{L}_{D_j}$ 35 : return ss 36 : else 37 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 38 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c, \text{ss}\}$ 39 : return ss
G (m) 16 : if $\exists r \text{ s.t. } (m, r) \in \mathcal{L}_{\mathbf{G}}$ 17 : return r 18 : $r \xleftarrow{\$} \mathcal{R}$ 19 : $\mathcal{L}_{\mathbf{G}} \leftarrow \mathcal{L}_{\mathbf{G}} \cup \{(m, r)\}$ 20 : return r	

Figure 4.18: $\mathbf{G}_4$ for the proof of Theorem 4.2.4

In $\mathbf{G}_4$, we introduce a flag CHAL that is raised whenever $\mathbf{H}$ is queried on a challenge ciphertext. The view of the adversary is identical to $\mathbf{G}_3$ unless, $\mathbf{H}(j, m, c)$ is queried on a

message (m, c) with $c \in \mathfrak{L}_C$ and $m = \text{Dec}_1(sk'_j, c)$. By the difference lemma

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \Pr[\text{CHAL}]$$

Now observe that $\Pr[\mathbf{G}_4 \Rightarrow 1] = \frac{1}{2}$, since the view of $\mathbf{B}$ is entirely independent of the bit b , as now both $\mathbf{ss}_0$ and $\mathbf{ss}_1$ are perfectly random, and $\mathbf{B}$ cannot gain useful information by querying some $m \in \mathfrak{L}_M$. Now, to bound $\Pr[\text{CHAL}]$ we construct a multi-target $\text{OW}_{n,u}\text{-PCA}$ adversary $\mathbf{A}$ against PKE_1 so that $\Pr[\text{CHAL}] = \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\mathbf{A})$. $\mathbf{A}$ is shown in [Figure 4.19](#).

$\mathbf{A}^{\text{PCO}}(\vec{pk})$ <hr style="border: 0; border-top: 1px solid black; margin: 5px 0;"/> <pre style="margin: 0;"> 1 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, \mathbf{H}, \mathbf{G}}(\vec{pk})$ 2 : if $\exists (m, c, \mathbf{ss}) \in \mathfrak{L}_{\mathbf{H}_j}$ s.t. $\text{PCO}(m, c) = 1$ and $c \in \mathfrak{L}_{C_j}$ 3 : return m 4 : else 5 : abort</pre>
--

Figure 4.19: An adversary $\mathbf{A}$ against the $\text{OW}_{n,u}\text{-PCA}$ -security of PKE_1 simulating $\mathbf{G}_4$ for $\mathbf{B}$.

Note that $\mathbf{A}$ perfectly simulates $\mathbf{G}_3$ for $\mathbf{B}$. In particular, $\mathcal{O}_{\text{Decaps}^\times}$ has no dependence on the secret key, and line 27 is equivalent to the check $\text{PCO}(m, c) = 1$. Observe that CHAL implies that $\mathbf{B}$ queried $\mathbf{H}(j, m, c)$ where $m \in \mathfrak{L}_{M_j}$ and $c \leftarrow \text{Enc}(pk_j, m; \mathbf{G}(m))$ hence $\mathbf{A}$ wins the game and returns m' . Thus

$$\Pr[\text{CHAL}] = \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\mathbf{A})$$

Adding up all the inequalities gives

$$\text{Adv}_{\text{KEM}^\times}^{\text{IND}_{n,u}\text{-CCA}}(\mathbf{B}) \leq \frac{q_{\mathbf{H}} + n^2}{|\mathcal{M}|} + \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\mathbf{A})$$

as desired. □

The bound given in Theorem 4.2.4 is meaningless when n is large relative to $|\mathcal{M}|$. If the message size is 128 bits or fewer, then for 2^{64} challenge ciphertexts, $\frac{n^2}{|\mathcal{M}|} = 1$. When salts are used, collisions in messages do not imply a collision in ciphertexts, and therefore to KEM keys when $b = 0$. The following theorem addresses this issue by incorporating salts, dramatically reducing the significance of that term.

Theorem 4.2.5 ($\text{ST}[\text{PKE}, \text{G}] \text{ OW}_{n,u}\text{-PCA} \xrightarrow{\text{ROM}} \text{SFO}_{m,c}^\chi[\text{PKE}, \text{G}, \text{H}, \text{len}_{\text{salt}}] \text{ IND}_{n,u}\text{-CCA}$). *If $\text{PKE}_1 \leftarrow \text{ST}[\text{PKE}, \text{G}]$ is δ -correct, then $\text{KEM}^\chi := \text{SFO}_{m,c}^\chi[\text{PKE}, \text{G}, \text{H}, \text{len}_{\text{salt}}]$ is δ -correct in the random oracle model. Moreover for any $\text{IND}_{n,u}\text{-CCA}$ adversary B against KEM^χ , issuing at most q_D queries to the decapsulation oracle $\mathcal{O}_{\text{Decaps}^\chi}$ and at most q_H queries to the random oracle H , there exists an $\text{OW}_{n,u}\text{-PCA}$ adversary A against PKE_1 issuing at most q_H queries to the PCO oracle such that*

$$\text{Adv}_{\text{KEM}^\chi}^{\text{IND}_{n,u}\text{-CCA}}(\text{B}) \leq \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}} + \frac{q_H}{|\mathcal{M}|} + \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\text{A})$$

where A and B have similar run-time.

Proof. The correctness bound is trivial. To prove the security claim, consider the sequence of games starting from Figure 4.20 which is the $\text{IND}_{n,u}\text{-CCA}$ game against $\text{FO}_{m,c}^\chi[\text{PKE}, \text{G}, \text{H}]$. G_0 is the $\text{IND}_{n,u}\text{-CCA}$ game against $\text{FO}_{m,c}^\chi[\text{PKE}, \text{G}, \text{H}]$, and so

$$\left| \Pr[\text{G}_0 \Rightarrow 1] - \frac{1}{2} \right| = \text{Adv}_{\text{KEM}^\chi}^{\text{IND}_{n,u}\text{-CCA}}(\text{B})$$

In G_1 , messages/salt pairs are sampled without replacement. This change is necessary because if $m_i \parallel \text{salt} = m_j \parallel \text{salt}$ for $i \neq j$, then $c_i = c_j$ due to the fact that encryption is deterministic. Then if $b = 0$, $\text{ss}_0 = \text{ss}_1$ while whenever $b = 1$ ss_0 and ss_1 are independent, which trivializes the $\text{IND}_{n,u}\text{-CCA}$ game. By the birthday bound and the difference lemma we have that

$$|\Pr[\text{G}_1 \Rightarrow 1] - \Pr[\text{G}_0 \Rightarrow 1]| \leq \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}}.$$

$G_0(B)$	$H(j, m, c \text{salt})$
1 : $b \xleftarrow{\$} \{0, 1\}$	16 : if $\exists \text{ss}$ s.t. $(m, c \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$
2 : for $j = 1, \dots, u$	17 : return ss
3 : $(pk_j, sk'_j) \xleftarrow{\$} \text{KeyGen}()$	18 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
4 : $s_j \xleftarrow{\$} \mathcal{M}$	19 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(m, c \text{salt}, \text{ss})\}$
5 : $sk_j \leftarrow (sk'_j, s_j)$	20 : return ss
6 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	$G(m \text{salt})$
7 : $b' \leftarrow B^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, H, G}(\vec{pk})$	21 : if $\exists r$ s.t. $(m \text{salt}, r) \in \mathcal{L}_G$
8 : return $\llbracket b = b' \rrbracket$	22 : return r
$\text{Chall}(j)$	23 : $r \xleftarrow{\$} \mathcal{R}$
9 : $(m, \text{salt}) \xleftarrow{\$} \mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}$	24 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \text{salt}, r)\}$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m \text{salt}\}$	25 : return r
11 : $c \leftarrow \text{Enc}(pk_j, m, G(m \text{salt}))$	$\mathcal{O}_{\text{Decaps}^\times}(j, c \text{salt})$
12 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \text{salt}\}$	26 : if $c \in \mathcal{L}_{C_j}$
13 : $\text{ss}_0 \leftarrow H(j, m, c \text{salt})$	27 : abort
14 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	28 : $m' \leftarrow \text{Dec}(sk'_j, c)$
15 : return $(c \text{salt}, \text{ss}_b)$	29 : if $m' = \perp$ or $c \neq \text{Enc}(pk_j, m'; G(m' \text{salt}))$
	30 : return $\text{ss} \leftarrow H(j, s, c \text{salt})$
	31 : return $\text{ss} \leftarrow H(j, m', c \text{salt})$

Figure 4.20: G_0 for the proof of Theorem 4.2.5

G₁(B)	H(j, m, c salt)
1 : $b \xleftarrow{\$} \{0, 1\}$	16 : if $\exists \text{ss s.t. } (m, c \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$
2 : for $j = 1, \dots, u$	17 : return ss
3 : $(pk_j, sk'_j) \xleftarrow{\$} \text{KeyGen}()$	18 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
4 : $s_j \xleftarrow{\$} \mathcal{M}$	19 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(m, c \text{salt}, \text{ss})\}$
5 : $sk_j \leftarrow (sk'_j, s_j)$	20 : return ss
6 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	G(m salt)
7 : $b' \leftarrow \text{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}, \text{H}, \text{G}}(\vec{pk})$	17 : if $\exists r \text{ s.t. } (m \text{salt}, r) \in \mathcal{L}_G$
8 : return $\llbracket b = b' \rrbracket$	18 : return r
Chall(j)	19 : $r \xleftarrow{\$} \mathcal{R}$
9 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}} \setminus (\mathcal{L}_{M_j} \times \mathcal{L}_{S_j}))$	20 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \text{salt}, r)\}$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m \text{salt}\}$	21 : return r
11 : $c \leftarrow \text{Enc}(pk_j, m, \text{G}(m \text{salt}))$	$\mathcal{O}_{\text{Decaps}}(j, c \text{salt})$
12 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \text{salt}\}$	21 : if $c \in \mathcal{L}_{C_j}$
13 : $\text{ss}_0 \leftarrow \text{H}(j, m, c \text{salt})$	22 : abort
14 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	23 : $m' \leftarrow \text{Dec}(sk'_j, c)$
15 : return $(c \text{salt}, \text{ss}_b)$	24 : if $m' = \perp$ or $c \neq \text{Enc}(pk_j, m'; \text{G}(m' \text{salt}))$
	25 : return $\text{ss} \leftarrow \text{H}(j, s, c \text{salt})$
	26 : return $\text{ss} \leftarrow \text{H}(j, m', c \text{salt})$

Figure 4.21: **G₁** for the proof of Theorem 4.2.5

In figure 4.2.2 we omit **G** since it doesn't change from Figure 4.2.1, and would cause the figure to be too large to fit on the page. In **G₂** we introduce two changes. If **H** is queried on s_j , the flag **QUERY** is raised, and the protocol is aborted. The other change is that decapsulation queries are made perfectly random when an input is rejected, making use of a separate internal random oracle **H'**. We observe that **G₂** and **G₁** are identical unless the adversary queries **H** on s_j , which is a uniformly secret value. Thus by the difference lemma

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{q_H}{|\mathcal{M}|}.$$

G₂(B)	H(j, m, c salt)
1 : $b \xleftarrow{\$} \{0, 1\}$	16 : if $\exists \text{ss s.t. } (m, c \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$
2 : for $j = 1, \dots, u$	17 : return ss
3 : $(pk_j, sk'_j) \xleftarrow{\$} \text{KeyGen}()$	18 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
4 : $s_j \xleftarrow{\$} \mathcal{M}$	19 : if $m = s_j$
5 : $sk_j \leftarrow (sk'_j, s_j)$	20 : QUERY = true
6 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	21 : abort
7 : $b' \leftarrow \mathcal{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, H, G}(\vec{pk})$	22 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(m, c \text{salt}, \text{ss})\}$
8 : return $\llbracket b = b' \rrbracket$	23 : return ss
Chall(j)	$\mathcal{O}_{\text{Decaps}^\times}(j, c \text{salt})$
9 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}} \setminus (\mathcal{L}_{M_j} \times \mathcal{L}_{S_j}))$	24 : if $c \in \mathcal{L}_{C_j}$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m \text{salt}\}$	25 : abort
11 : $c \leftarrow \text{Enc}(pk_j, m, G(m \text{salt}))$	26 : $m' \leftarrow \text{Dec}(sk'_j, c)$
12 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \text{salt}\}$	27 : if $m' = \perp$ or $c \neq \text{Enc}(pk_j, m'; G(m' \text{salt}))$
13 : $\text{ss}_0 \leftarrow H(j, m, c \text{salt})$	28 : return $\text{ss} \leftarrow H(j, s, c \text{salt})$
14 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	29 : return $\text{ss} \leftarrow H(j, m', c \text{salt})$
15 : return $(c \text{salt}, \text{ss}_b)$	

Figure 4.22: **G₂** for the proof of Theorem 4.2.5

In **G₃** lists $\mathcal{L}_{D_j}$ and $\mathcal{L}_{H_j}$ are introduced for bookkeeping, and the oracles are modified so as to no longer make use of the secret keys, except to test if $\text{Dec}_1(sk'_j, c||\text{salt}) = m$ on line 27. In **G₃** and **G₂** the view of the adversary is identical. To see this, consider a fixed plaintext ciphertext and salt ($c||\text{salt}, m' \leftarrow \text{Dec}_1(sk'_j, c||\text{salt})$). Consider the following cases

$\mathbf{G}_3(\mathcal{B})$	$\mathbf{H}(j, m, c \parallel \text{salt})$
1 : $b \xleftarrow{\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk'_j) \xleftarrow{\$} \text{KeyGen}()$ 4 : $s_j \xleftarrow{\$} \mathcal{M}$ 5 : $sk_j \leftarrow (sk'_j, s_j)$ 6 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$ 7 : $b' \leftarrow \mathcal{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, \mathbf{H}, \mathbf{G}}(\vec{pk})$ 8 : return $\llbracket b = b' \rrbracket$	21 : if $\exists \text{ss s.t. } (m, c \parallel \text{salt}, \text{ss}) \in \mathcal{L}_{\mathbf{H}_j}$ 22 : return ss 23 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 24 : if $m = s_j$ 25 : $\text{QUERY} = \text{true}; \text{abort}$ 26 : if $\text{Dec}(sk'_j, c) = m$ and $\text{Enc}(pk_j, m; \mathbf{G}(m \parallel \text{salt})) = c$ 27 : if $\exists \text{ss}' \text{ s.t. } (c \parallel \text{salt}, \text{ss}') \in \mathcal{L}_{D_j}$ 28 : $\text{ss} := \text{ss}'$ 29 : else 30 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$ 31 : $\mathcal{L}_{\mathbf{H}_j} \leftarrow \mathcal{L}_{\mathbf{H}_j} \cup \{(m, c \parallel \text{salt}, \text{ss})\}$ 32 : return ss
$\text{Chall}(j)$	$\mathcal{O}_{\text{Decaps}^\times}(j, c \parallel \text{salt})$
9 : $m \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_j}$ 10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m \parallel \text{salt}\}$ 11 : $c \leftarrow \text{Enc}(pk_j, m, \mathbf{G}(m \parallel \text{salt}))$ 12 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \parallel \text{salt}\}$ 13 : $\text{ss}_0 \leftarrow \mathbf{H}(j, m, c \parallel \text{salt})$ 14 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$ 15 : return $(c \parallel \text{salt}, \text{ss}_b)$	33 : if $c \in \mathcal{L}_{C_j}$ 34 : abort 35 : if $\exists \text{ss s.t. } (c \parallel \text{salt}, \text{ss}) \in \mathcal{L}_{D_j}$ 36 : return ss 37 : else 38 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 39 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$ 40 : return ss
$\mathbf{G}(m \parallel \text{salt})$	
16 : if $\exists r \text{ s.t. } (m \parallel \text{salt}, r) \in \mathcal{L}_{\mathbf{G}}$ 17 : return r 18 : $r \xleftarrow{\$} \mathcal{R}$ 19 : $\mathcal{L}_{\mathbf{G}} \leftarrow \mathcal{L}_{\mathbf{G}} \cup \{(m \parallel \text{salt}, r)\}$ 20 : return r	

Figure 4.23: $\mathbf{G}_3$ for the proof of Theorem 4.2.5

Case 1: $m \in \{\perp, s\}$.

in $\mathbf{G}_3$ $\mathbf{H}$ takes as input a message $m \in \mathcal{M}$, and $c \in \mathcal{C}$. $\mathbf{H}(j, \perp, c)$ cannot be queried. Querying $\mathbf{H}(j, s, c \parallel \text{salt})$ raises the flag QUERY on line 25. Hence querying $\mathbf{H}$ can never

append a tuple $(c\|\text{salt}, \text{ss})$ to the list $\mathcal{L}_{D_j}$. $\mathcal{O}_{\text{Decaps}^\times}(j, c\|\text{salt})$, the output will be a uniformly random value ss .

Case 2: $m \notin \{\perp, s\}$

We aim to show that $\mathcal{O}_{\text{Decaps}^\times}(j, c\|\text{salt})$ returns $\text{H}(j, m', c\|\text{salt})$ for valid ciphertexts with $\text{Dec}_1(sk'_j, c\|\text{salt}) = m'$. There are two possibilities here: $\text{H}(j, m', c\|\text{salt})$ is queried by B first, then $\mathcal{O}_{\text{Decaps}^\times}(j, c\|\text{salt})$; or $\mathcal{O}_{\text{Decaps}^\times}(j, c\|\text{salt})$ is queried first then $\text{H}(j, m', c\|\text{salt})$.

Suppose B queries $\text{H}(j, m', c\|\text{salt})$ first. Then on line 31, since $\mathcal{O}_{\text{Decaps}^\times}$ has not yet been queried on $c\|\text{salt}$, $(c\|\text{salt}, \text{ss})$ will be appended to $\mathcal{L}_{D_j}$. Then on line 32 $(m, c\|\text{salt}, \text{ss})$ will be appended to $\mathcal{L}_{H_j}$ and $\text{H}(j, m, c\|\text{salt})$ will output ss . Then when $\mathcal{O}_{\text{Decaps}^\times}(j, c\|\text{salt})$ is queried, line 34 will recognize $(c\|\text{salt}, \text{ss}) \in \mathcal{L}_{D_j}$ and $\mathcal{O}_{\text{Decaps}^\times}(j, c\|\text{salt})$ will return ss .

Now suppose that B first queries $\mathcal{O}_{\text{Decaps}^\times}(j, c\|\text{salt})$. Then $(c\|\text{salt}, \text{ss})$ will be appended to $\mathcal{L}_{D_j}$. When H is queried, line 28 will recognize $(c, \text{ss}) \in \mathcal{L}_{D_j}$, then $(m, c\|\text{salt}, \text{ss})$ will be appended to $\mathcal{L}_{H_j}$, and $\text{H}(j, m'\|\text{salt}, c)$ will output ss .

Hence, in $\mathbf{G}_3$ and $\mathbf{G}_2$, the view of B is identical, so

$$\Pr[\mathbf{G}_3 \Rightarrow 1] = \Pr[\mathbf{G}_2 \Rightarrow 1].$$

In $\mathbf{G}_4$, we introduce a flag CHAL that is raised whenever H is queried on a challenge ciphertext. The view of the adversary is identical to $\mathbf{G}_3$ unless, $\text{H}(j, m, c)$ is queried on a message (m, c) with $c \in \mathcal{L}_C$ and $m = \text{Dec}_1(sk'_j, c)$. By the difference lemma

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \Pr[\text{CHAL}]$$

Now observe that $\Pr[\mathbf{G}_4 \Rightarrow 1] = \frac{1}{2}$, since the view of B is entirely independent of the bit b , as now, for all i , both ss_0 and ss_1 are perfectly random, and B cannot gain useful information by querying some $m \in \mathcal{L}_M$.

$\mathbf{G}_4(\mathbf{B})$	$\mathbf{H}(j, m, c \text{salt})$
1 : $b \xleftarrow{\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk'_j) \xleftarrow{\$} \text{KeyGen}()$ 4 : $s_j \xleftarrow{\$} \mathcal{M}$ 5 : $sk_j \leftarrow (sk'_j, s_j)$ 6 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$ 7 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, \mathbf{H}, \mathbf{G}}(\vec{pk})$ 8 : return $\llbracket b = b' \rrbracket$	21 : if $\exists \text{ss s.t. } (m, c \text{salt}, \text{ss}) \in \mathcal{L}_{\mathbf{H}_j}$ 22 : return ss 23 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 24 : if $m = s_j$ 25 : $\text{QUERY} = \text{true}$ 26 : abort 27 : if $c \in \mathcal{L}_{C_j}$ 28 : $\text{CHAL} = \text{true}$ 29 : abort 30 : if $\text{Dec}(sk'_j, c) = m$ and $\text{Enc}(pk_j, m; \mathbf{G}(m \text{salt})) = c$ 31 : if $\exists \text{ss}' \text{ s.t. } (c \text{salt}, \text{ss}') \in \mathcal{L}_{D_j}$ 32 : $\text{ss} := \text{ss}'$ 33 : else 34 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \text{salt}, \text{ss}\}$ 35 : $\mathcal{L}_{\mathbf{H}_j} \leftarrow \mathcal{L}_{\mathbf{H}_j} \cup \{(m, c \text{salt}, \text{ss})\}$ 36 : return ss
$\text{Chall}(j)$	$\mathcal{O}_{\text{Decaps}^\times}(j, c \text{salt})$
9 : $m \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_j}$ 10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m \text{salt}\}$ 11 : $c \leftarrow \text{Enc}(pk_j, m, \mathbf{G}(m \text{salt}))$ 12 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \text{salt}\}$ 13 : $\text{ss}_0 \xleftarrow{\$} \mathcal{K}$ 14 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$ 15 : return $(c \text{salt}, \text{ss}_b)$	37 : if $c \in \mathcal{L}_{C_j}$ 38 : abort 39 : if $\exists \text{ss s.t. } (c \text{salt}, \text{ss}) \in \mathcal{L}_{D_j}$ 40 : return ss 41 : else 42 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 43 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \text{salt}, \text{ss}\}$ 44 : return ss
$\mathbf{G}(m \text{salt})$	
16 : if $\exists r \text{ s.t. } (m \text{salt}, r) \in \mathcal{L}_{\mathbf{G}}$ 17 : return r 18 : $r \xleftarrow{\$} \mathcal{R}$ 19 : $\mathcal{L}_{\mathbf{G}} \leftarrow \mathcal{L}_{\mathbf{G}} \cup \{(m \text{salt}, r)\}$ 20 : return r	

Figure 4.24: $\mathbf{G}_4$ for the proof of Theorem 4.2.5

Now, to bound $\Pr[\text{CHAL}]$ we construct a multi-target $\text{OW}_{n,u}$ -PCA adversary $\mathbf{A}$ against

PKE_1 so that $\Pr[\text{CHAL}] = \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\mathbf{A})$. $\mathbf{A}$ is shown in Figure 4.19.

$\mathbf{A}^{\text{PCO}}(\vec{pk})$ <hr/> <pre> 1 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, \mathbf{H}, \mathbf{G}}(\vec{pk})$ 2 : if $\exists (m, c \parallel \text{salt}, \text{ss}) \in \mathfrak{L}_{\mathbf{H}_j}$ s.t. $\text{PCO}(m, c \parallel \text{salt}) = 1$ and $c \in \mathfrak{L}_{C_j}$ 3 : return m 4 : else 5 : abort</pre>
--

Figure 4.25: An adversary $\mathbf{A}$ against the $\text{OW}_{n,u}$ -PCA-security of PKE_1 simulating $\mathbf{G}_4$ for $\mathbf{B}$.

Note that $\mathbf{A}$ perfectly simulates $\mathbf{G}_3$ for $\mathbf{B}$. In particular, $\mathcal{O}_{\text{Decaps}^\times}$ has no dependence on the secret key, and line 27 is equivalent to the check $\text{PCO}(m, c) = 1$. Observe that CHAL implies that $\mathbf{B}$ queried $\mathbf{H}(j, m, c \parallel \text{salt})$ where $m \in \mathfrak{L}_{M_j}$ and $c \parallel \text{salt} \leftarrow \text{Enc}(pk_j, m; \mathbf{G}(m \parallel \text{salt}))$ hence $\mathbf{A}$ wins the game and returns m' . Thus

$$\Pr[\text{CHAL}] = \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\mathbf{A})$$

Adding up all the inequalities gives

$$\text{Adv}_{\text{KEM}^\times}^{\text{IND}_{n,u}\text{-CCA}}(\mathbf{B}) \leq \frac{q_{\mathbf{H}} + n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}} + \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\mathbf{A})$$

as desired. □

4.2.4 The $\mathbf{U}^\perp$ -transform: $\text{OW}_{n,u}$ -PCVA to $\text{IND}_{n,u}$ -CCA with explicit rejection

In the previous theorems, we showed the bound for both the salted and unsalted KEM. The motivation for this was specific comparison between **FrodoKEM** and **eFrodoKEM**. For the remaining Theorems, we present only the bound for KEMs obtained from the **SFO**-transform, since the proofs are largely identical. You may obtain bounds for the generic **FO**-transform by taking $\text{len}_{\text{salt}} = 0$, and $|\mathfrak{L}_{\mathcal{S}}| = 1$.

Theorem 4.2.6 ($\text{ST}[\text{PKE}, \text{G}] \text{ OW}_{n,u}\text{-PCVA} \xrightarrow{\text{ROM}} \text{SFO}_{m,c}^\perp[\text{PKE}, \text{G}, \text{H}, \text{len}_{\text{salt}}] \text{ IND}_{n,u}\text{-CCA}$). *If $\text{PKE}_1 := \text{ST}[\text{PKE}, \text{G}, \text{len}_{\text{salt}}]$ is δ -correct, then $\text{KEM}^\perp := \text{SFO}_{m,c}^\perp[\text{PKE}_1, \text{H}, \text{len}_{\text{salt}}]$ is δ -correct in the random oracle model. Moreover for any $\text{IND}_{n,u}\text{-CCA}$ adversary B against $\text{KEM}^\perp$, issuing at most q_D queries to the decapsulation oracle $\mathcal{O}_{\text{Decaps}}^\perp$ and at most q_H queries to the random oracle H , there exists an $\text{OW}_{n,u}\text{-PCVA}$ adversary A against PKE_1 issuing at most q_H queries to the PCO oracle such that*

$$\text{Adv}_{\text{KEM}^\perp}^{\text{IND}_{n,u}\text{-CCA}}(\text{B}) \leq \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}} + \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\text{A})$$

where A and B have similar run-time.

Proof. The correctness bound is trivial. To prove the security claim, consider the sequence of games starting from Figure 4.2.7 which the $\text{IND}_{n,u}\text{-CCA}$ game against $\text{FO}_{m,c}^\perp[\text{PKE}, \text{G}, \text{H}]$.

$\mathbf{G}_0$ is the $\text{IND}_{n,u}\text{-CCA}$ game against $\text{FO}_{m,c}^\perp[\text{PKE}, \text{G}, \text{H}]$, and so

$$\left| \Pr[\mathbf{G}_0 \Rightarrow 1] - \frac{1}{2} \right| = \text{Adv}_{\text{KEM}^\perp}^{\text{IND}_{n,u}\text{-CCA}}(\text{B})$$

In $\mathbf{G}_1$, messages/salt pairs are sampled without replacement. This change is necessary because if $m_i \parallel \text{salt} = m_j \parallel \text{salt}$ for $i \neq j$, then $c_i = c_j$ due to the fact that encryption is deterministic. Then if $b = 0$, $\text{ss}_0 = \text{ss}_1$ while whenever $b = 1$ ss_0 and ss_1 are independent, which trivializes the $\text{IND}_{n,u}\text{-CCA}$ game. By the birthday bound and the difference lemma we have that

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}}.$$

In $\mathbf{G}_2$ lists $\mathfrak{L}_{D_j}$ and $\mathfrak{L}_{H_j}$ are introduced for bookkeeping, and the oracles are modified so as to no longer make use of the secret key. We claim that in $\mathbf{G}_2$ and $\mathbf{G}_1$ the view of the adversary is identical. To see this, consider a fixed plaintext ciphertext pair $(c \parallel \text{salt}, m' \leftarrow \text{Dec}_1(sk_j, c' \parallel \text{salt}))$, where salt is public. Consider the following cases

$G_0(B)$	$H(j, m, c \text{salt})$
1 : $b \xleftarrow{\$} \{0, 1\}$	14 : if $\exists \text{ss}$ s.t. $(m, c \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$
2 : for $j = 1, \dots, u$	15 : return ss
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	16 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
4 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$	17 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(m, c \text{salt}, \text{ss})\}$
5 : $b' \leftarrow B^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\mathcal{L}}, H, G}(\vec{pk})$	18 : return ss
6 : return $\llbracket b = b' \rrbracket$	$G(m \text{salt})$
$\text{Chall}(j)$	19 : if $\exists r$ s.t. $(m \text{salt}, r) \in \mathcal{L}_G$
7 : $(m, \text{salt}) \xleftarrow{\$} \mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}$	20 : return r
8 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m \text{salt}\}$	21 : $r \xleftarrow{\$} \mathcal{R}$
9 : $c \leftarrow \text{Enc}(pk_j, m, G(m \text{salt}))$	22 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \text{salt}, r)\}$
10 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \text{salt}\}$	23 : return r
11 : $\text{ss}_0 \leftarrow H(j, m, c \text{salt})$	$\mathcal{O}_{\text{Decaps}}^\perp(j, c \text{salt})$
12 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	24 : if $c \in \mathcal{L}_{C_j}$
13 : return $(c \text{salt}, \text{ss}_b)$	25 : abort
	26 : $m' \leftarrow \text{Dec}(sk_j, c)$
	27 : if $m' = \perp$ or $c \neq \text{Enc}(pk_j, m'; G(m' \text{salt}))$
	28 : return $\perp$
	29 : return $\text{ss} \leftarrow H(j, m', c \text{salt})$

Figure 4.26: G_0 for the proof of Theorem 4.2.6

Case 1: $m \notin \mathcal{M}$.

If $m \notin \mathcal{M}$, this is equivalent to $m = \perp$. Then $\mathcal{O}_{\text{Decaps}}^\perp(j, c||\text{salt})$ returns $\perp$ in either game.

Case 2: $m \in \mathcal{M}$.

We aim to show that $\mathcal{O}_{\text{Decaps}^\mathcal{L}}(j, -)$ returns $H(j, m', c||\text{salt})$ for valid ciphertexts with $\text{Dec}_1(sk_j, c||\text{salt}) = m'$. There are two possibilities here: $H(j, m', c||\text{salt})$ is queried by B

first, then $\mathcal{O}_{\text{Decaps}}^\perp(j, c\|\text{salt})$; or $\mathcal{O}_{\text{Decaps}}^\perp(j, c\|\text{salt})$ is queried first, then $H(j, m', c\|\text{salt})$.

$\mathbf{G}_1(\mathbf{B})$ <hr/> 1 : $b \xleftarrow{\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$ 4 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$ 5 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^\perp, \mathbf{H}, \mathbf{G}}(\vec{pk})$ 6 : return $\llbracket b = b' \rrbracket$ $\text{Chall}(j)$ <hr/> 7 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}} \setminus (\mathcal{L}_{M_j} \times \mathcal{L}_{S_j}))$ 8 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\ \text{salt}\}$ 9 : $c \leftarrow \text{Enc}(pk_j, m, \mathbf{G}(m\ \text{salt}))$ 10 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c\ \text{salt}\}$ 11 : $\text{ss}_0 \leftarrow H(j, m, c\ \text{salt})$ 12 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$ 13 : return $(c\ \text{salt}, \text{ss}_b)$	$H(j, m, c\ \text{salt})$ <hr/> 14 : if $\exists \text{ss s.t. } (m, c\ \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$ 15 : return ss 16 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 17 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(m, c\ \text{salt}, \text{ss})\}$ 18 : return ss $\mathbf{G}(m\ \text{salt})$ <hr/> 19 : if $\exists r \text{ s.t. } (m\ \text{salt}, r) \in \mathcal{L}_G$ 20 : return r 21 : $r \xleftarrow{\$} \mathcal{R}$ 22 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m\ \text{salt}, r)\}$ 23 : return r $\mathcal{O}_{\text{Decaps}}^\perp(j, c\ \text{salt})$ <hr/> 24 : if $c \in \mathcal{L}_{C_j}$ 25 : abort 26 : $m' \leftarrow \text{Dec}(sk'_j, c)$ 27 : if $m' = \perp$ or $c \neq \text{Enc}(pk_j, m'; \mathbf{G}(m'\ \text{salt}))$ 28 : return $\perp$ 29 : return $\text{ss} \leftarrow H(j, m', c\ \text{salt})$
--	--

Figure 4.27: $\mathbf{G}_1$ for the proof of Theorem 4.2.6

$\mathbf{G}_2(\mathbf{B})$ <hr/> 1 : $b \xleftarrow{\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$ 4 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$ 5 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^\perp, \mathbf{H}, \mathbf{G}}(\vec{pk})$ 6 : return $\llbracket b = b' \rrbracket$ $\text{Chall}(j)$ <hr/> 7 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}} \setminus (\mathfrak{L}_{M_j} \times \mathfrak{L}_{S_j}))$ 8 : $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m \parallel \text{salt}\}$ 9 : $c \leftarrow \text{Enc}(pk_j, m, \mathbf{G}(m \parallel \text{salt}))$ 10 : $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c \parallel \text{salt}\}$ 11 : $\text{ss}_0 \leftarrow \mathbf{H}(j, m, c \parallel \text{salt})$ 12 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$ 13 : return $(c \parallel \text{salt}, \text{ss}_b)$ $\mathbf{G}(m)$ <hr/> 14 : if $\exists r$ s.t. $(m \parallel \text{salt}, r) \in \mathfrak{L}_{\mathbf{G}}$ 15 : return r 16 : $r \xleftarrow{\$} \mathcal{R}$ 17 : $\mathfrak{L}_{\mathbf{G}} \leftarrow \mathfrak{L}_{\mathbf{G}} \cup \{(m \parallel \text{salt}, r)\}$ 18 : return r	$\mathbf{H}(j, m, c \parallel \text{salt})$ <hr/> 19 : if $\exists \text{ss}$ s.t. $(m, c \parallel \text{salt}, \text{ss}) \in \mathfrak{L}_{\mathbf{H}_j}$ 20 : return ss 21 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 22 : if $\text{Enc}(pk, m, \mathbf{G}(m \parallel \text{salt})) = c$ 23 : if $\exists \text{ss}'$ s.t. $(c \parallel \text{salt}, \text{ss}') \in \mathfrak{L}_{D_j}$ 24 : $\text{ss} := \text{ss}'$ 25 : else 26 : $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$ 27 : $\mathfrak{L}_{\mathbf{H}_j} \leftarrow \mathfrak{L}_{\mathbf{H}_j} \cup \{(m, c \parallel \text{salt}, \text{ss})\}$ 28 : return ss $\mathcal{O}_{\text{Decaps}}^\perp(j, c \parallel \text{salt})$ <hr/> 29 : if $c \in \mathfrak{L}_{C_j}$ 30 : abort 31 : if $\exists \text{ss}$ s.t. $(c \parallel \text{salt}, \text{ss}) \in \mathfrak{L}_{D_j}$ 32 : return ss 33 : if $\text{Dec}_1(sk, c \parallel \text{salt}) \notin \mathcal{M}$ 34 : return $\perp$ 35 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 36 : $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$ 37 : return ss
---	---

Figure 4.28: $\mathbf{G}_2$ for the proof of Theorem 4.2.6

Suppose $\mathbf{B}$ queries $\mathbf{H}(j, m', c \parallel \text{salt})$ first. Then on line 26, since $\mathcal{O}_{\text{Decaps}}^\perp$ has not yet been queried on $(j, c \parallel \text{salt})$, $(c \parallel \text{salt}, \text{ss})$ will be appended to $\mathfrak{L}_{D_j}$. Then on line 27 $(m, c \parallel \text{salt}, \text{ss})$ will be appended to $\mathfrak{L}_{\mathbf{H}}$ and finally $\mathbf{H}(m, c \parallel \text{salt})$ will output ss . Then when $\mathcal{O}_{\text{Decaps}}^\perp(c \parallel \text{salt})$ is queried, on line 31 the oracle finds $(c \parallel \text{salt}, \text{ss}) \in \mathfrak{L}_D$ and return $\text{ss} = \mathbf{H}(m', c \parallel \text{salt})$.

Now suppose that $\mathbf{B}$ first queries $\mathcal{O}_{\text{Decaps}}^\perp(j, c\|\text{salt})$. Then on line 36, $(c\|\text{salt}, \text{ss})$ will be appended to $\mathfrak{L}_{D_j}$. When $\mathbf{H}$ is queried, line 23 will find $(c\|\text{salt}, \text{ss}) \in \mathfrak{L}_{D_j}$, then $(m, c\|\text{salt}, \text{ss})$ will be appended to $\mathfrak{L}_{H_j}$, and $\mathbf{H}(j, m', c\|\text{salt})$ will output $\text{ss} = \text{Decaps}^\perp(j, c\|\text{salt})$.

$\mathbf{G}_3(\mathbf{B})$	$\mathbf{H}(j, m, c\ \text{salt})$
<hr/> 1 : $b \xleftarrow{\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$ 4 : $\vec{pk} \leftarrow (pk_1, \dots, pk_u)$ 5 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^\perp, \mathbf{H}, \mathbf{G}}(\vec{pk})$ 6 : return $\llbracket b = b' \rrbracket$	<hr/> 19 : if $\exists \text{ss}$ s.t. $(m, c\ \text{salt}, \text{ss}) \in \mathfrak{L}_{H_j}$ 20 : return ss 21 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 22 : if $\text{Enc}(pk, m, \mathbf{G}(m\ \text{salt})) = c$ 23 : if $c \in \mathfrak{L}_{C_j}$ 24 : $\text{CHAL} = \text{true}$ 25 : abort 26 : if $\exists \text{ss}'$ s.t. $(c\ \text{salt}, \text{ss}') \in \mathfrak{L}_{D_j}$ 27 : $\text{ss} := \text{ss}'$ 28 : else 29 : $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c\ \text{salt}, \text{ss}\}$ 30 : $\mathfrak{L}_{H_j} \leftarrow \mathfrak{L}_{H_j} \cup \{(m, c\ \text{salt}, \text{ss})\}$ 31 : return ss
<hr/> Chall (j) 7 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}) \setminus (\mathfrak{L}_{M_j} \times \mathfrak{L}_{S_j})$ 8 : $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m\ \text{salt}\}$ 9 : $c \leftarrow \text{Enc}(pk_j, m, \mathbf{G}(m\ \text{salt}))$ 10 : $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c\ \text{salt}\}$ 11 : $\text{ss}_0 \xleftarrow{\$} \mathcal{K}$ 12 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$ 13 : return $(c\ \text{salt}, \text{ss}_b)$	<hr/> $\mathcal{O}_{\text{Decaps}}^\perp(j, c\ \text{salt})$ 32 : if $c \in \mathfrak{L}_{C_j}$ 33 : abort 34 : if $\exists \text{ss}$ s.t. $(c\ \text{salt}, \text{ss}) \in \mathfrak{L}_{D_j}$ 35 : return ss 36 : if $\text{Dec}_1(sk, c\ \text{salt}) \notin \mathcal{M}$ 37 : return $\perp$ 38 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 39 : $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c\ \text{salt}, \text{ss}\}$ 40 : return ss
<hr/> G (m) 14 : if $\exists r$ s.t. $(m\ \text{salt}, r) \in \mathfrak{L}_G$ 15 : return r 16 : $r \xleftarrow{\$} \mathcal{R}$ 17 : $\mathfrak{L}_G \leftarrow \mathfrak{L}_G \cup \{(m\ \text{salt}, r)\}$ 18 : return r	

Figure 4.29: $\mathbf{G}_3$ for the proof of Theorem 4.2.6

Hence, in $\mathbf{G}_1$ and $\mathbf{G}_2$, the view of $\mathbf{B}$ is identical, so

$$\Pr[\mathbf{G}_2 \Rightarrow 1] = \Pr[\mathbf{G}_1 \Rightarrow 1].$$

In $\mathbf{G}_3$ we introduce a flag CHAL that is raised whenever $\mathbf{H}$ is queried on a challenge ciphertext. The view of the adversary is identical to $\mathbf{G}_3$ unless, $\mathbf{H}(j, m, c)$ is queried on a message (m, c) with $c \in \mathfrak{L}_C$ and $m = \text{Dec}_1(sk'_j, c)$. By the difference lemma

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \Pr[\text{CHAL}]$$

Now observe that $\Pr[\mathbf{G}_3 \Rightarrow 1] = \frac{1}{2}$, since the view of $\mathbf{B}$ is entirely independent of the bit b , as now both $\mathbf{ss}_0$ and $\mathbf{ss}_1$ are perfectly random, and $\mathbf{B}$ cannot gain useful information by querying some $m \in \mathfrak{L}_M$. Now, to bound $\Pr[\text{CHAL}]$ we construct a multi-target $\text{OW}_{n,u}\text{-PCA}$ adversary $\mathbf{A}$ against PKE_1 so that $\Pr[\text{CHAL}] = \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\mathbf{A})$. $\mathbf{A}$ is shown in Figure 4.19.

$\mathbf{A}^{\text{PCO}, \text{CVO}}(\vec{pk})$ <hr style="border: 0.5px solid black;"/> <pre style="margin: 0; font-family: monospace;"> 1 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^\perp, \mathbf{H}, \mathbf{G}}(\vec{pk})$ 2 : if $\exists(m, c \parallel \text{salt}, \mathbf{ss}) \in \mathfrak{L}_{\mathbf{H}_j}$ s.t. $\text{PCO}(m, c \parallel \text{salt}) = 1$ and $c \in \mathfrak{L}_{C_j}$ 3 : return m 4 : else 5 : abort</pre>
--

Figure 4.30: An adversary $\mathbf{A}$ against the $\text{OW}_{n,u}\text{-PCA}$ -security of PKE_1 simulating $\mathbf{G}_3$ for $\mathbf{B}$.

Note that $\mathbf{A}$ perfectly simulates $\mathbf{G}_3$ for $\mathbf{B}$. In particular, $\mathcal{O}_{\text{Decaps}}^\perp$ has no dependence on the secret key, and line 36 is equivalent to the check $\text{CVO}(c \parallel \text{salt}) = 1$. Observe that CHAL implies that $\mathbf{B}$ queried $\mathbf{H}(j, m, c \parallel \text{salt})$ where $m \in \mathfrak{L}_{M_j}$ and $c \parallel \text{salt} \leftarrow \text{Enc}(pk_j, m; \mathbf{G}(m \parallel \text{salt}))$ hence $\mathbf{A}$ wins the game and returns m' . Thus

$$\Pr[\text{CHAL}] = \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\mathbf{A})$$

Adding up all the inequalities gives

$$\text{Adv}_{\text{KEM}^\perp}^{\text{IND}_{n,u}\text{-CCA}}(\mathbf{B}) \leq \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}} + \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(\mathbf{A})$$

as desired. □

4.2.5 Public Key Hashing

In practice, schemes are designed to mitigate increased risk induced by having a large number of users (large u) by including a hash of the public key into the generation of encapsulations. Though it had been standard practice to make use of the public key in the generation of keys in all FO-based KEMs, until [DHK⁺21], there had been no formal analysis of this technique in the literature. In [DHK⁺21] Duman et al establish the connection between public key hashing and multi-target attacks.

Public key hashing entails including pk as an input into the hash function for generating encapsulations. For example, $\text{SFO}_{pk,m,c}^\chi[\text{PKE}, G, H]$ is shown in Figure 4.31.

$\text{SFO}_{m,c,pk}^\chi[\text{PKE}, G, H, \text{len}_{\text{salt}}].\text{KeyGen}()$	$\text{SFO}_{m,c,pk}^\chi[\text{PKE}].\text{Decaps}(sk, c \text{salt})$
1 : $(pk', sk') \leftarrow \text{KeyGen}()$	4 : $m' \leftarrow \text{Dec}(sk, c)$
2 : $s \xleftarrow{\$} \mathcal{M}$	5 : if $m' = \perp$ or $\text{Enc}(pk, m; G(m \text{salt})) \neq c$
3 : $sk \leftarrow (sk', s)$	6 : return $ss \leftarrow H(s, c \text{salt}, pk)$
4 : return (pk, sk)	7 : else return $ss \leftarrow H(m', c \text{salt}, pk)$
$\text{SFO}_{m,c,pk}^\chi[\text{PKE}, G, H, \text{len}_{\text{salt}}].\text{Encaps}(pk)$	
1 : $m \xleftarrow{\$} \mathcal{M}$	
2 : $\text{salt} \xleftarrow{\$} \{0, 1\}^{\text{len}_{\text{salt}}}$	
3 : $c \leftarrow \text{Enc}(pk, m; G(m \text{salt}))$	
4 : $ss \leftarrow H(m, c \text{salt}, pk)$	
5 : return $(ss, c \text{salt})$	

Figure 4.31: The KEM $\text{SFO}_{pk,m,c}^\chi[\text{PKE}, G, H, \text{len}_{\text{salt}}]$

The following two Theorems relate to KEMs that differ from shown in Theorem 4.24, 4.25 and 4.26 in their use of the hash function H . Hashing can be computationally costly, and each inclusion into the hash function should be justified by some necessary functionality. In [BHH⁺19] Bindel et al. show that hashing $H(m)$ is equivalent to hashing $H(m, c)$ in that the IND-CCA security of either KEM is equivalent. To briefly summarize, the argument provided there is that for $\text{KEM}_1 = \text{U}_{m,c}[\text{PKE}, H_1]$ and $\text{KEM}_2 = \text{U}_m[\text{PKE}, H_2]$, an adversary against the IND-CCA security of KEM_2 can perfectly simulate the IND-CCA game for KEM_1 by sampling a new random oracle H and setting

$$H_1(m, c) = \begin{cases} H_2(m), & \text{if } c = \text{Enc}(pk, m), \\ H(m, c), & \text{otherwise.} \end{cases}$$

In the other direction, an adversary against the IND-CCA security of KEM_1 can simulate the IND-CCA game for KEM_2 by letting $H_2(m) \leftarrow H_1(m, \text{Enc}(pk, m))$.

This argument does not extend to the multi-target setting, as there would be u public keys to consider. In fact, $H(m)$ is in no way connected to the public key of any user, while $H(m, c \leftarrow \text{Enc}(pk_j(m)))$ is. However, if the public key hash is included, the argument does hold, with the simulation being

$$H_1(pk_j, m, c) = \begin{cases} H_2(pk_j, m), & \text{if } c = \text{Enc}(pk_j, m), \\ H(pk_j, m, c), & \text{otherwise.} \end{cases}$$

The difference being that now the adversary is required to have knowledge of the public key in either case.

Theorem 4.2.7 ($\text{ST}[\text{PKE}, G] \text{ OW}_{n,u}\text{-PCA} \xrightarrow{\text{ROM}} \text{SFO}_{pk,m}^\perp[\text{PKE}, G, H, \text{len}_{\text{salt}}] \text{ IND}_{n,u}\text{-CCA}$). *If $\text{PKE}_1 := \text{ST}[\text{PKE}, G]$ is δ -correct, then $\text{KEM}^\perp := \text{SFO}_{pk,m}^\perp[\text{PKE}_1, H, \text{len}_{\text{salt}}]$ is δ -correct in the random oracle model. Moreover for any $\text{IND}_{n,u}\text{-CCA}$ adversary B against $\text{KEM}^\perp$, issuing at most q_D queries to the decapsulation oracle $\mathcal{O}_{\text{Decaps}}^\perp$ and at most q_H queries to the random oracle H , there exists an $\text{OW}_{n,u}\text{-PCVA}$ adversary A against PKE_1 issuing at most q_H queries to the PCO oracle such that*

$$\text{Adv}_{\text{KEM}^\perp}^{\text{IND}_{n,u}\text{-CCA}}(B) \leq \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}} + \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(A)$$

where A and B have similar run-time.

Proof. Combine Theorem 5 in [BHH⁺19] with the bound from Theorem 4.2.5. Theorem 5 of [BHH⁺19] shows that in the proof of Theorem 4.2.5 we can replace each instance of $\text{KEM}^\perp := \text{SFO}_{m,c}^\perp[\text{PKE}, G, H]$, with $\text{KEM}'^\perp := \text{SFO}_m^\perp[\text{PKE}, G, H]$ without impacting security. $\square$

Theorem 4.2.8 ($\text{ST}[\text{PKE}, G] \text{ OW}_{n,u}\text{-PCVA} \xrightarrow{\text{ROM}} \text{SFO}_{pk,m}^\perp[\text{PKE}, G, H, \text{len}_{\text{salt}}] \text{ IND}_{n,u}\text{-CCA}$). *If $\text{PKE}_1 := \text{ST}[\text{PKE}, G]$ is δ -correct, then $\text{KEM}^\perp := \text{SFO}_{pk,m}^\perp[\text{PKE}_1, H, \text{len}_{\text{salt}}]$ is δ -correct in the random oracle model. Moreover for any $\text{IND}_{n,u}\text{-CCA}$ adversary B against $\text{KEM}^\perp$,*

issuing at most q_D queries to the decapsulation oracle $\mathcal{O}_{\text{Decaps}}^\times$ and at most q_H queries to the random oracle H , there exists an $\text{OW}_{n,u}$ -PCVA adversary A against PKE_1 issuing at most q_H queries to the PCO oracle such that

$$\text{Adv}_{\text{KEM}^\perp}^{\text{IND}_{n,u}\text{-CCA}}(B) \leq \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}} + \text{Adv}_{\text{PKE}_1}^{\text{OW}_{n,u}\text{-PCA}}(A)$$

where A and B have similar run-time.

Proof. Combine Theorem 5 in [BHH⁺19] with the bound from Theorem 4.2.6. Theorem 5 of [BHH⁺19] shows that in the proof of Theorem 4.2.6 we can replace each instance of $\text{KEM}^\perp := \text{SFO}_{m,c}^\perp[\text{PKE}, G, H]$, with $\text{KEM}'^\perp := \text{SFO}_m^\perp[\text{PKE}, G, H]$ without impacting security. $\square$

4.2.6 Prefix Hashing

Hashing pk can be a costly measure, especially in lattice-based schemes where public keys are large matrices, rather than bit strings. To reach the target security level, it may be sufficient to hash only a bit string that uniquely identifies the public key. In [DHK⁺21], Duman et al show that for an identifying function with sufficient entropy, one can get the same security level, with a significant decrease in overhead. This amounts to hashing a small, unpredictable part of pk , by using an identifying function $\text{ID} : \mathcal{PK} \rightarrow \{0, 1\}^\ell$. This technique is known as prefix hashing.

In the security analysis, when prefix hashing is used, a game hop can be added, which introduces a flag COLL , and aborts if there is a collision in public key identifiers meaning there are indices i, j with $i \neq j$ and $F(pk_i) = F(pk_j)$. By the birthday bound, the probability of COLL is $\frac{u^2}{2^\ell}$, where u is the number of users. Provided the flag COLL is not raised, there exists a unique identifier pointing to each public key. The analysis proceeds identically, and the bound differs only by the term $\frac{u^2}{2^\ell}$, while in [DHK⁺21] authors experimentally showed that, even with a high entropy identifying function, overhead could be reduced by over 30% for certain lattice based KEMs.

4.2.7 Monolithic Security Bounds

We could obtain bounds for the transforms $\text{FO}_{m,c}^\times, \text{FO}_{m,c}^\perp, \text{SFO}_{m,c}^\times, \text{SFO}_{m,c}^\perp$, by combining the bounds for each component separately. This would introduce some redundancy, as certain

terms would be unnecessarily repeated in both stages of the proof. In this section, we prove a monolithic security bound for $\text{SFO}_{m,c}^\chi$. Comparing this bound to the sum of the modular bounds for ST and $\text{U}_{m,c}^\chi$ will show the redundant terms, which are the same for each version of SFO. A table of monolithic bounds is shown in Figure 4.40.

Theorem 4.2.9 ($\text{PKE IND}_{n,u}\text{-CPA} \xrightarrow{\text{ROM}} \text{SFO}_{\text{F}(pk),m,c}^\chi[\text{PKE}, \text{G}, \text{H}, \text{len}_{\text{salt}}] \text{IND}_{n,u}\text{-CCA}$). *If PKE is $\delta(u)$ -correct, then $\text{SFO}_{\text{F}(pk),m,c}^\chi[\text{PKE}, \text{G}, \text{H}, \text{len}_{\text{salt}}]$ is $\delta(u)$ -correct in the random oracle model. Moreover for any $\text{IND}_{n,u}\text{-CCA}$ adversary B against KEM^χ , issuing at most q_{H} queries to H , q_{G} queries to G and q_{D} queries to $\mathcal{O}_{\text{Decaps}^\chi}$, there exists an $\text{IND}_{n,u}\text{-CPA}$ adversary A against PKE such that*

$$\begin{aligned} \text{Adv}_{\text{KEM}^\chi}^{\text{IND}_{n,u}\text{-CCA}}(\text{B}) &\leq \frac{u^2}{2^\ell} + \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}} + \max_{j \in [u]} \frac{2n_j(q_{\text{G}} + q_{\text{H}})}{|\mathcal{M}||\mathfrak{L}_{S_j}|} + \frac{q_{\text{H}}}{|\mathcal{M}|} \\ &\quad + (q_{\text{G}} + q_{\text{H}} + q_{\text{D}}) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\text{A}) \end{aligned}$$

where ℓ is the min entropy of $\text{F}(pk)$ i.e. $\ell = H_\infty(pk)$ where $(pk, sk) \leftarrow \text{KeyGen}$, n_j is the number of queries of the form $\text{Chall}(j)$ and $\mathfrak{L}_{S_j}$ is the list of salts sampled for user j . Furthermore A and B have similar run-time.

Proof. The correctness bound is trivial. To prove the security claim, consider the sequence of games starting from Figure 4.32.

G_0 is the $\text{IND}_{n,u}\text{-CCA}$ -game against $\text{KEM}^\chi := \text{SFO}_{\text{pki},m,c}^\chi[\text{PKE}, \text{G}, \text{H}, \text{len}_{\text{salt}}]$ hence

$$|\Pr[\text{G}_0 \Rightarrow 1]| = \text{Adv}_{\text{KEM}^\chi}^{\text{IND}_{n,u}\text{-CCA}}(\text{B})$$

In G_1 we introduce a flag COLL and abort if there is a collision in public key identifiers, which allows each pki to be uniquely identified as $\text{F}(pk_j)$ for some j . This change is unnoticed as long as there is no collision in public key identifiers. By the birthday bound

$$|\Pr[\text{G}_0 \Rightarrow 1] - \Pr[\text{G}_1 \Rightarrow 1]| = \frac{u^2}{2^\ell}$$

where ℓ is the min entropy of $\text{F}(pk)$. where ℓ is the min entropy of $\text{F}(pk)$ i.e. $\ell = H_\infty(pk)$ where $(pk, sk) \leftarrow \text{KeyGen}$. In many cases public keys are very large, and so even a very large public key identifier, with high entropy, amounts to a significant reduction in overhead, while having negligible impact on security level. In FrodoKEM, for example, a 256 bit hash value $\text{pkh} \leftarrow \text{SHAKE}(pk, \text{len}_{\text{pkh}})$ is used as a public key identifier.

$\mathbf{G}_0(\mathbf{B})$ <hr/> 1 : $b \xleftarrow{\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$ 4 : $s_j \xleftarrow{\$} \mathcal{M}$ 5 : $\vec{pk} = (pk_1, \dots, pk_j)$ 6 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, \mathbf{G}, \mathbf{H}}(\vec{pk})$ 7 : return $\llbracket b = b' \rrbracket$	$\mathbf{G}(j, m \parallel \text{salt})$ <hr/> 16 : if $\exists r$ s.t. $(m \parallel \text{salt}, r) \in \mathfrak{L}_{\mathbf{G}}$ 17 : return r 18 : $r \xleftarrow{\$} \mathcal{R}$ 19 : $\mathfrak{L}_{\mathbf{G}} \leftarrow \mathfrak{L}_{\mathbf{G}} \cup \{(m \parallel \text{salt}, r)\}$ 20 : return r
$\text{Chall}(j)$ <hr/> 8 : $(m, \text{salt}) \xleftarrow{\$} \mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}$ 9 : $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m\}$ 10 : $r \leftarrow \mathbf{G}(m \parallel \text{salt})$ 11 : $c = \text{Enc}(pk_j, m; r)$ 12 : $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c \parallel \text{salt}\}$ 13 : $\text{ss}_0 = \mathbf{H}(j, \mathbf{F}(pk_j), m, c \parallel \text{salt})$ 14 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$ 15 : return ss	$\mathbf{H}(j, \text{pki}, m, c \parallel \text{salt})$ <hr/> 24 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \parallel \text{salt}, \text{ss}) \in \mathfrak{L}_{\mathbf{H}_j}$ 25 : return ss 26 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 27 : $\mathfrak{L}_{\mathbf{H}_j} \leftarrow \mathfrak{L}_{\mathbf{H}_j} \cup \{(\text{pki}, m, c \parallel \text{salt}, \text{ss})\}$ 28 : return ss
	$\mathcal{O}_{\text{Decaps}^\times}(j, c \parallel \text{salt})$ <hr/> 29 : if $c \in \mathfrak{L}_{C_j}$ 30 : abort 31 : $m' \leftarrow \text{Dec}(sk'_j, c \parallel \text{salt})$ 32 : if $m' = \perp$ or $\text{Enc}(pk_j, m'; \mathbf{G}(j, m' \parallel \text{salt})) \neq c$ 33 : $\text{ss} \leftarrow \mathbf{H}(j, \text{pki}, \text{id}, s_j, c \parallel \text{salt})$ 34 : $\text{ss} \leftarrow \mathbf{H}(j, \text{pki}, m', c \parallel \text{salt})$ 35 : $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$ 36 : return ss

Figure 4.32: $\mathbf{G}_0$ for the proof of Theorem 4.2.9

G₁(B)	G(j, m salt)
1 : $b \xleftarrow{\$} \{0, 1\}$	18 : if $\exists r$ s.t. $(m \text{salt}, r) \in \mathfrak{L}_G$
2 : for $j = 1, \dots, u$	19 : return r
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	20 : $r \xleftarrow{\$} \mathcal{R}$
4 : $s_j \xleftarrow{\$} \mathcal{M}$	21 : $\mathfrak{L}_G \leftarrow \mathfrak{L}_G \cup \{(m \text{salt}, r)\}$
5 : $\vec{pk} = (pk_1, \dots, pk_j)$	22 : return r
6 : if $\exists i, j \in [u]$ s.t. $F(pk_i) = F(pk_j)$	H(j, pki, m, c salt)
7 : COLL = true; abort	26 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \text{salt}, \text{ss}) \in \mathfrak{L}_{H_j}$
8 : $b' \leftarrow \text{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\times}, G, H}(\vec{pk})$	27 : return ss
9 : return $\llbracket b = b' \rrbracket$	28 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
Chall(j)	29 : $\mathfrak{L}_{H_j} \leftarrow \mathfrak{L}_{H_j} \cup \{(\text{pki}, m, c \text{salt}, \text{ss})\}$
10 : $(m, \text{salt}) \xleftarrow{\$} \mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}$	30 : return ss
11 : $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m\}$	$\mathcal{O}_{\text{Decaps}^\times}(j, c \text{salt})$
12 : $r \leftarrow G(m \text{salt})$	29 : if $c \in \mathfrak{L}_{C_j}$
13 : $c = \text{Enc}(pk_j, m; r)$	30 : abort
14 : $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c \text{salt}\}$	31 : $m' \leftarrow \text{Dec}(sk'_j, c \text{salt})$
15 : $\text{ss}_0 = H(j, F(pk_j), m, c \text{salt})$	32 : if $m' = \perp$ or $\text{Enc}(pk_j, m'; G(j, m' \text{salt})) \neq c$
16 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	33 : $\text{ss} \leftarrow H(j, \text{pki}, \text{id}, s_j, c \text{salt})$
17 : return ss	34 : $\text{ss} \leftarrow H(j, \text{pki}, m', c \text{salt}) \mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c \text{salt}, \text{ss}\}$
	35 : return ss

Figure 4.33: **G₁** for the proof of Theorem 4.2.9

In **G₂** we change the experiment so that message/salt pairs are sampled without replacement for each user, since otherwise the same message/salt pair would lead to identical KEM keys, and the adversary could distinguish between the real world and the random world. Since the total number of challenge messages is n , by the birthday bound we have

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}}.$$

G₂(B)	G(j, m salt)
1 : $b \xleftarrow{\$} \{0, 1\}$	18 : if $\exists r$ s.t. $(m \text{salt}, r) \in \mathcal{L}_G$
2 : for $j = 1, \dots, u$	19 : return r
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	20 : $r \xleftarrow{\$} \mathcal{R}$
4 : $s_j \xleftarrow{\$} \mathcal{M}$	21 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \text{salt}, r)\}$
5 : $\vec{pk} = (pk_1, \dots, pk_j)$	22 : return r
6 : if $\exists i, j \in [u]$ s.t. $F(pk_i) = F(pk_j)$	H(j, pki, m, c salt)
7 : $\text{COLL} = \text{true}$; abort	26 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$
8 : $b' \leftarrow \text{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}^\perp}, G, H}(\vec{pk})$	27 : return ss
9 : return $\llbracket b = b' \rrbracket$	28 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
Chall(j)	29 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(\text{pki}, m, c \text{salt}, \text{ss})\}$
10 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}} \setminus (\mathcal{L}_{M_j} \times \mathcal{L}_{S_j}))$	30 : return ss
11 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	$\mathcal{O}_{\text{Decaps}^\perp}(j, c \text{salt})$
12 : $r \leftarrow G(m \text{salt})$	29 : if $c \in \mathcal{L}_{C_j}$
13 : $c = \text{Enc}(pk_j, m; r)$	30 : abort
14 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \text{salt}\}$	31 : $m' \leftarrow \text{Dec}(sk'_j, c \text{salt})$
15 : $\text{ss}_0 \leftarrow H(j, F(pk_j), m, c \text{salt})$	32 : if $m' = \perp$ or $\text{Enc}(pk_j, m'; G(j, m' \text{salt})) \neq c$
16 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	33 : $\text{ss} \leftarrow H(j, \text{pki}, \text{id}, s_j, c \text{salt})$
17 : return ss	34 : $\text{ss} \leftarrow H(j, \text{pki}, m', c \text{salt})$
	35 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \text{salt}, \text{ss}\}$
	36 : return ss

Figure 4.34: **G₂** for the proof of Theorem 4.2.9.

$\mathbf{G}_3(\mathbf{B})$ <hr/> 1 : $b \xleftarrow{\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$ 4 : $s_j \xleftarrow{\$} \mathcal{M}$ 5 : $\vec{pk} = (pk_1, \dots, pk_j)$ 6 : if $\exists i, j \in [u]$ s.t. $F(pk_i) = F(pk_j)$ 7 : $\text{COLL} = \text{true}$; abort 8 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^\perp, \mathbf{G}, \mathbf{H}}(\vec{pk})$ 9 : return $\llbracket b = b' \rrbracket$ $\text{Chall}(j)$ <hr/> 10 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}) \setminus (\mathfrak{L}_{M_j} \times \mathfrak{L}_{S_j})$ 11 : $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m\}$ 12 : $r \leftarrow \mathbf{G}(m \parallel \text{salt})$ 13 : $c = \text{Enc}(pk_j, m; r)$ 14 : $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c \parallel \text{salt}\}$ 15 : $\text{ss}_0 = \mathbf{H}(j, F(pk_j), m, c \parallel \text{salt})$ 16 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$ 17 : return ss	$\mathbf{G}(j, m \parallel \text{salt})$ <hr/> 18 : if $\exists r$ s.t. $(m \parallel \text{salt}, r) \in \mathfrak{L}_{\mathbf{G}}$ 19 : return r 20 : $r \xleftarrow{\$} \mathcal{R}$ 21 : $\mathfrak{L}_{\mathbf{G}} \leftarrow \mathfrak{L}_{\mathbf{G}} \cup \{(m \parallel \text{salt}, r)\}$ 22 : return r $\mathbf{H}(j, \text{pki}, m, c \parallel \text{salt})$ <hr/> 26 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \parallel \text{salt}, \text{ss}) \in \mathfrak{L}_{\mathbf{H}_j}$ 27 : return ss 28 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 29 : if $m = s_j$ 30 : $\text{QUERY} = \text{true}$ 31 : abort 32 : $\mathfrak{L}_{\mathbf{H}_j} \leftarrow \mathfrak{L}_{\mathbf{H}_j} \cup \{(\text{pki}, m, c \parallel \text{salt}, \text{ss})\}$ 33 : return ss $\mathcal{O}_{\text{Decaps}}^\perp(j, c \parallel \text{salt})$ <hr/> 31 : if $c \in \mathfrak{L}_{C_j}$ 32 : abort 33 : $m' \leftarrow \text{Dec}(sk'_j, c \parallel \text{salt})$ 34 : if $m' = \perp$ or $\text{Enc}(pk_j, m'; \mathbf{G}(j, m' \parallel \text{salt})) \neq c$ 35 : $\text{ss} \leftarrow \mathbf{H}'(j, \text{pki}, \text{id}, m', c \parallel \text{salt})$ 36 : if $m' = s_j$ 37 : $\text{ss} \leftarrow \mathbf{H}'(j, \text{pki}, \text{id}, m', c \parallel \text{salt})$ 38 : $\text{ss} \leftarrow \mathbf{H}(j, \text{pki}, m', c \parallel \text{salt})$ 39 : $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$ 40 : return ss
---	--

Figure 4.35: $\mathbf{G}_3$ for the proof of Theorem 4.2.9.

In $\mathbf{G}_3$ we abort whenever $\mathbf{H}$ is queried on s_j , and we make the outputs of $\mathbf{H}$ perfectly random when an input is rejected by making use of a separate internal random oracle $\mathbf{H}'$. $\mathbf{G}_3$ and $\mathbf{G}_2$ are identical unless the adversary queries $\mathbf{H}$ on s_j . Thus

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \frac{q_{\mathbf{H}}}{|\mathcal{M}|}.$$

In $\mathbf{G}_4$, we remove dependence on the secret key and claim that the oracles $\mathbf{H}$ and $\mathcal{O}_{\text{Decaps}^\times}$ are patched so that, assuming there is no correctness error, $\mathbf{G}_3$ and $\mathbf{G}_4$ are identical. Hence, if CORR is the event that $\mathbf{B}$ queries $\mathbf{G}$, with a j and $m \parallel \text{salt}$ such that $m \neq \text{Dec}(sk_j, \text{Enc}(pk_j, m; \mathbf{G}(m \parallel \text{salt})))$, by $\delta(u)$ -correctness, we have that

$$\Pr[\text{CORR}] \leq (q_{\mathbf{G}} + q_{\mathbf{H}} + q_D) \cdot \delta(u)$$

Now we show that $\mathbf{G}_4$ and $\mathbf{G}_3$ proceed identically, conditioned on $\neg \text{CORR}$.

Consider a query $\mathcal{O}_{\text{Decaps}^\times}(j, c \parallel \text{salt})$ where $c = \text{Enc}(pk_j, m; \mathbf{G}(j, m \parallel \text{salt}))$. Let $m' = \text{Dec}(\text{pki}, sk_j, c \parallel \text{salt})$ and $c' = \text{Enc}(pk_j, m'; \mathbf{G}(j, m' \parallel \text{salt}))$.

- **Case 1:** $m' = \perp$, then in both $\mathbf{G}_4$ and $\mathbf{G}_3$ $\mathcal{O}_{\text{Decaps}^\times}$ outputs a uniformly random ss, and $\mathbf{H}$ cannot be queried on m .
- **Case 2:** $m \neq \perp$ and $c \neq c'$. In this case, $\mathcal{O}_{\text{Decaps}^\times}$ will output a uniformly random secret in either game. However, in $\mathbf{G}_4$ a query $\mathbf{H}(j, m, c \parallel \text{salt})$ would return the same value as $\mathcal{O}_{\text{Decaps}^\times}(j, c \parallel \text{salt})$, whereas in $\mathbf{G}_3$ $\mathcal{O}_{\text{Decaps}^\times}(j, c \parallel \text{salt})$ would output $\mathbf{H}'(j, \text{pki}, c \parallel \text{salt})$, while $\mathbf{H}(j, \text{pki}, m, c \parallel \text{salt})$ would output an independent random value. This can only happen if m exhibits a correctness error.
- **Case 3:** $m \neq \perp$ and $c = c'$. In $\mathbf{G}_3$ the output of $\mathcal{O}_{\text{Decaps}^\times}$ will be $\mathbf{H}(j, \text{pki}, m', c \parallel \text{salt})$. In $\mathbf{G}_4$ the output of $\mathcal{O}_{\text{Decaps}^\times}(j, c \parallel \text{salt})$ will be $\mathbf{H}(j, \text{pki}, m, c \parallel \text{salt})$ if $\mathbf{H}(j, \text{pki}, m, c \parallel \text{salt})$. Hence the output of the two games will only be different if $m \neq m'$.

By the difference lemma

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq (q_{\mathbf{G}} + q_{\mathbf{H}} + q_D) \cdot \delta(u)$$

$\mathbf{G}_4(\mathbf{B})$ <hr/> <pre> 1 : $b \xleftarrow{\\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk_j) \xleftarrow{\\$} \text{KeyGen}()$ 4 : $s_j \xleftarrow{\\$} \mathcal{M}$ 5 : $\vec{pk} = (pk_1, \dots, pk_j)$ 6 : if $\exists i, j \in [u]$ s.t. $F(pk_i) = F(pk_j)$ 7 : $\text{COLL} = \text{true}$; abort 8 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^\perp, \mathbf{G}, \mathbf{H}}(\vec{pk})$ 9 : return $\llbracket b = b' \rrbracket$ </pre> $\text{Chall}(j)$ <hr/> <pre> 10 : $(m, \text{salt}) \xleftarrow{\\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}) \setminus (\mathfrak{L}_{M_j} \times \mathfrak{L}_{S_j})$ 11 : $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m\}$ 12 : $r \leftarrow \mathbf{G}(m \parallel \text{salt})$ 13 : $c = \text{Enc}(pk_j, m; r)$ 14 : $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c \parallel \text{salt}\}$ 15 : $\text{ss}_0 = \mathbf{H}(j, F(pk_j), m, c \parallel \text{salt})$ 16 : $\text{ss}_1 \xleftarrow{\\$} \mathcal{K}$ 17 : return ss </pre>	$\mathbf{G}(j, m \parallel \text{salt})$ <hr/> <pre> 18 : if $\exists r$ s.t. $(m \parallel \text{salt}, r) \in \mathfrak{L}_{\mathbf{G}}$ 19 : return r 20 : $r \xleftarrow{\\$} \mathcal{R}$ 21 : $\mathfrak{L}_{\mathbf{G}} \leftarrow \mathfrak{L}_{\mathbf{G}} \cup \{(m \parallel \text{salt}, r)\}$ 22 : return r </pre> $\mathbf{H}(j, \text{pki}, m, c \parallel \text{salt})$ <hr/> <pre> 23 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \parallel \text{salt}, \text{ss}) \in \mathfrak{L}_{\mathbf{H}_j}$ 24 : return ss 25 : if $m = s_j$ 26 : $\text{QUERY} = \text{true}$ 27 : abort 28 : if $\text{Enc}_1(pk_j, m) = c$ 29 : if $\exists \text{ss}'$ s.t. $(\text{pki}, c \parallel \text{salt}, \text{ss}') \in \mathfrak{L}_{D_j}$ 30 : $\text{ss} := \text{ss}'$ 31 : else 32 : $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{\text{pki}, c \parallel \text{salt}, \text{ss}\}$ 33 : $\mathfrak{L}_{\mathbf{H}_j} \leftarrow \mathfrak{L}_{\mathbf{H}_j} \cup \{(\text{pki}, m, c \parallel \text{salt}, \text{ss})\}$ 34 : return ss </pre> $\mathcal{O}_{\text{Decaps}}^\perp(j, c \parallel \text{salt})$ <hr/> <pre> 35 : if $c \in \mathfrak{L}_{C_j}$ 36 : abort 37 : if $\exists \text{ss}$ s.t. $(\text{pki}, c \parallel \text{salt}, \text{ss}) \in \mathfrak{L}_{D_j}$ 38 : return ss 39 : $\text{ss} \xleftarrow{\\$} \mathcal{K}$ 40 : $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$ 41 : return ss </pre>
--	--

Figure 4.36: $\mathbf{G}_4$ for the proof of Theorem 4.2.9.

In $\mathbf{G}_5$ we raise a flag **CHAL** and abort if the adversary is able to correctly produce a KEM key that would match a challenge ciphertext. That is, we abort if the adversary queries $\mathbf{H}(\text{pki}, j, m, c \parallel \text{salt})$ or $\mathbf{G}(j, m \parallel \text{salt})$ where $c \in \mathcal{L}_{C_j}$ and $\text{Enc}(pk_j, m; \mathbf{G}(m; \text{salt})) = c$. We also change **Chall** so that it samples r at random, rather than fixing $r \leftarrow \mathbf{G}(m)$. This allows a CPA adversary to simulate **CHAL** in $\mathbf{G}_5$, using their own challenge oracle for PKE. Provided **CHAL** is not raised, this change goes unnoticed since $\mathbf{B}$ is not given access to $\mathbf{G}(m)$. We have

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \Pr[\text{CHAL}]$$

Furthermore, in $\mathbf{G}_5$ the adversary is provided with random shared secrets independent of the bit b . Then we have

$$|\Pr[\mathbf{G}_5 \Rightarrow 1]| = \frac{1}{2}$$

It remains to bound **CHAL**. To this end, we construct an $\text{IND}_{n,u}\text{-CPA}$ adversary $\mathbf{A}$ against PKE, that perfectly simulates $\mathbf{G}_5$ for $\mathbf{B}$, and wins whenever **CHAL** is raised.

We claim that $\mathbf{A}$ simulates $\mathbf{G}_5$ for $\mathbf{B}$. To simulate **CHAL** for b , $\mathbf{A}$ samples $\text{salt} \leftarrow \{0, 1\}^{\text{len}_{\text{salt}}}$, and appends salt to $\mathcal{L}_{S_j}$. Then $\mathbf{A}$ samples $m_0 \leftarrow \mathcal{MS}_{\text{new}_{j,0}}$ and appends $m_0 \parallel \text{salt}$ to $\mathcal{L}_{M_{j,0}}$, and samples $m_1 \leftarrow \mathcal{MS}_{\text{new}_{j,1}}$ and appends $m_1 \parallel \text{salt}$ to $\mathcal{L}_{M_{j,1}}$. Then $\mathbf{A}$ queries its own **Chall** on input (j, m_0, m_1) to obtain $c = \text{Enc}(pk, m_b)$ and appends $c \parallel \text{salt}$ to $\mathcal{L}_{C_j}$. $\mathbf{A}$ samples a random $\text{ss} \xleftarrow{\$} \mathcal{K}$, and returns $(c \parallel \text{salt}, K)$ to $\mathbf{B}$, which matches the output of **Chall** in $\mathbf{G}_5$. Observe that $\mathcal{O}_{\text{Decaps}^\times}$ has been modified to no longer make use of sk_j , and $\mathbf{H}$ is patched so that its output always matches that of $\mathcal{O}_{\text{Decaps}^\times}$. Hence the simulation is perfect.

Let **BADG** be the event that $\mathbf{B}$ queries $\mathbf{G}$ on any $(j, m \parallel \text{salt})$ or $\mathbf{H}(\mathbf{F}(pk_j), m, c \parallel \text{salt})$ with $m \parallel \text{salt} \in \mathcal{L}_{M_{j,1-b}}$.

$$\Pr[\text{BADG}] = \max_{j \in [u]} \frac{n_j \cdot (q_{\mathbf{H}} + q_{\mathbf{G}})}{|\mathcal{M}| |\mathcal{L}_{S_j}|}$$

This imposes a trade-off on $\mathbf{B}$. As n_j increases, $|\mathcal{L}_{S_j}|$ increases proportionately. The relative size of $\mathcal{L}_{S_j}$ decreases with the number of collisions in salt values, and is therefore minimized when $n_j = n$.

G₅(B)	H(j, pki, m, c salt)
1 : $b \xleftarrow{\$} \{0, 1\}$	25 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$
2 : for $j = 1, \dots, u$	26 : return ss
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	27 : if $m = s_j$
4 : $s_j \xleftarrow{\$} \mathcal{M}$	28 : $\text{QUERY} = \text{true}$
5 : $\vec{pk} = (pk_1, \dots, pk_j)$	29 : abort
6 : if $\exists i, j \in [u]$ s.t. $F(pk_i) = F(pk_j)$	30 : if $m \in \mathcal{L}_{M_j}$
7 : $\text{COLL} = \text{true};$ abort	31 : $\text{CHAL} = \text{true};$ abort
8 : $b' \leftarrow \text{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^\perp, \text{G}, \text{H}}(\vec{pk})$	32 : if $\text{Enc}_1(pk_j, m) = c$
9 : return $\llbracket b = b' \rrbracket$	33 : if $\exists \text{ss}'$ s.t. $(\text{pki}, c \text{salt}, \text{ss}') \in \mathcal{L}_{D_j}$
Chall(j)	34 : $\text{ss} := \text{ss}'$
10 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}) \setminus (\mathcal{L}_{M_j} \times \mathcal{L}_{S_j})$	35 : else
11 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	36 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{\text{pki}, c \text{salt}, \text{ss}\}$
12 : $r \xleftarrow{\$} \mathcal{R}$	37 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(\text{pki}, m, c \text{salt}, \text{ss})\}$
13 : $c = \text{Enc}(pk_j, m; r)$	38 : return ss
14 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \text{salt}\}$	
15 : $\text{ss}_0 \xleftarrow{\$} \mathcal{K}$	$\mathcal{O}_{\text{Decaps}}^\perp(j, c \text{salt})$
16 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	30 : if $c \in \mathcal{L}_{C_j}$
17 : return ss	31 : abort
G(j, m salt)	32 : if $\exists \text{ss}$ s.t. $(\text{pki}, c \text{salt}, \text{ss}) \in \mathcal{L}_{D_j}$
18 : if $\exists r$ s.t. $(m \text{salt}, r) \in \mathcal{L}_G$	33 : return ss
19 : return r	34 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
20 : $r \xleftarrow{\$} \mathcal{R}$	35 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \text{salt}, \text{ss}\}$
21 : if $m \in \mathcal{L}_{M_j}$	36 : return ss
22 : $\text{CHAL} = \text{true};$ abort	
23 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \text{salt}, r)\}$	
24 : return r	

Figure 4.37: **G₅** for the proof of Theorem 4.2.9.

$A(\vec{pk})$
<pre> 1 : $\tilde{b} \leftarrow \mathcal{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^\perp, \mathcal{H}, \mathcal{G}}(\vec{pk})$ 2 : if $\exists (m_0 \parallel \text{salt}) \in \mathcal{L}_{M_{j,0}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_{G_j} \text{ or } (F(pk_j), m, c \parallel \text{salt}) \in \mathcal{L}_{H_j})$ 3 : if $\exists (m_1 \parallel \text{salt}) \in \mathcal{L}_{M_{j,1}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_{G_j} \text{ or } (F(pk_j), m, c \parallel \text{salt}) \in \mathcal{L}_{H_j})$ 4 : $b' \xleftarrow{\\$} \{0, 1\}$ 5 : else $b' := 0$ 6 : if $\exists (m_1 \parallel \text{salt}) \in \mathcal{L}_{M_{j,1}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_{G_j} \text{ or } (F(pk_j), m, c \parallel \text{salt}) \in \mathcal{L}_{H_j})$ 7 : if $\exists (m_0 \parallel \text{salt}) \in \mathcal{L}_{M_{j,0}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_{G_j} \text{ or } (F(pk_j), m, c \parallel \text{salt}) \in \mathcal{L}_{H_j})$ 8 : $b' \xleftarrow{\\$} \{0, 1\}$ 9 : else $b' := 1$ 10 : else $b' \xleftarrow{\\$} \{0, 1\}$ 11 : return $\llbracket b = b' \rrbracket$ </pre>

Figure 4.38: Adversaries B_1 and B_2 for the proof of 4.2.9

Consider the advantage of A against the $\text{IND}_{n,u}\text{-CPA}$ security of PKE. D looks for random oracle queries containing $m_0 \in \mathcal{L}_{M_{j,0}}$ or $m_1 \in \mathcal{L}_{M_{j,1}}$. If one is found, this indicates that either the flag CHAL or BADG was raised. If CHAL is raised, and BADG is not, then A detects a random oracle query containing $m \in \mathcal{L}_{M_{j,b}}$. If neither CHAL or BADG is raised, A returns a random guess, as is the case when both CHAL and BADG are raised. Then provided BADG is not raised, A wins the game with probability 1 if CHAL is raised, and with probability $\frac{1}{2}$ otherwise. Hence

$$\begin{aligned}
\text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(A) + \Pr[\text{BADG}] &\geq \left| \Pr[b = b' | \neg \text{BADG}] - \frac{1}{2} \right| \\
&= \left| \Pr[\text{CHAL}] + \frac{1}{2} \Pr[\neg \text{CHAL}] - \frac{1}{2} \right| \\
&= \frac{1}{2} \Pr[\text{CHAL}].
\end{aligned}$$

And so

$$\Pr[\text{CHAL}] \leq \max_{j \in [u]} \frac{(2n_j \cdot q_H + q_G)}{|\mathcal{M}| |\mathcal{L}_{S_j}|} + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(A).$$

We claim that this term does not impact tightness. Consider

$$\max_{j \in [u]} \frac{n_j}{|\mathcal{L}_{S_j}|}.$$

The expected number of distinct salt values for n_j queries to **Chall**, can be obtained by the formula for the expected number of distinct results, when drawing n_j samples from a set with replacement.

$$\mathbb{E}(|\mathcal{L}_{S_j}|) = 2^{\text{len}_{\text{salt}}} \left(1 - \left(1 - \frac{1}{2^{\text{len}_{\text{salt}}}} \right)^{n_j} \right)$$

Fix $\text{len}_{\text{salt}} = \lfloor \log_2(n_j) \rfloor$. The above formula yields

$$\begin{aligned} \max_{j \in [u]} \frac{n_j}{\mathbb{E}(|\mathcal{L}_{S_j}|)} &\approx \frac{1}{\left(1 - \left(1 - \frac{1}{2^{\text{len}_{\text{salt}}}} \right)^{n_j} \right)} \\ \max_{j \in [u]} \frac{n_j}{\mathbb{E}(|\mathcal{L}_{S_j}|)} &\approx 1.58 \end{aligned}$$

Adding all inequalities gives

$$\begin{aligned} \text{Adv}_{\text{KEM}^\perp}^{\text{IND}_{n,u}\text{-CCA}}(\text{B}) &\leq \frac{u^2}{2^\ell} + \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}} + \max_{j \in [u]} \frac{2n_j(q_G + q_H)}{|\mathcal{M}||\mathcal{L}_{S_j}|} + \frac{q_H}{|\mathcal{M}|} \\ &\quad + (q_G + q_H + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\text{A}) \end{aligned}$$

as desired. $\square$

Theorem 4.2.10 (PKE $\text{IND}_{n,u}\text{-CPA} \xrightarrow{\text{ROM}} \text{SFO}_{(\text{pki}, m, c)}^\perp[\text{PKE}, G, H, \text{len}_{\text{salt}}] \text{IND}_{n,u}\text{-CCA}$). *If PKE is $\delta(u)$ -correct and γ -spread, then $\text{KEM}^\perp := \text{SFO}_{(\text{pki}, m, c)}^\perp[\text{PKE}, G, H, \text{len}_{\text{salt}}]$ is $\delta(u)$ -correct in the random oracle model. Moreover for any $\text{IND}_{n,u}\text{-CCA}$ adversary B against $\text{KEM}^\perp$, issuing at most q_H queries to H , q_G queries to G and q_D queries to $\mathcal{O}_{\text{Decaps}^\perp}$, there exists an $\text{IND}_{n,u}\text{-CPA}$ adversary A against PKE such that*

$$\begin{aligned} \text{Adv}_{\text{KEM}^\perp}^{\text{IND}_{n,u}\text{-CCA}}(\text{B}) &\leq \frac{u^2}{2^\ell} + \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}} + \frac{2n(q_G + q_H)}{|\mathcal{L}_{S_j}||\mathcal{M}|} + q_D \cdot 2^{-\gamma} \\ &\quad + (q_G + q_H + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\text{A}) \end{aligned}$$

where ℓ is the min entropy of $F(pk)$ i.e. $\ell = H_\infty(pk)$ where $(pk, sk) \leftarrow \text{KeyGen}$, n_j is the number of queries of the form $\text{Chall}(j)$ and $\mathfrak{L}_{S_j}$ is the list of salts sampled for user j . Furthermore **A** and **B** have similar run-time.

Proof. The correctness bound is trivial. To prove the security claim, consider the sequence of games starting from Figure 4.39.

$\mathbf{G}_0$ is the $\text{IND}_{n,u}\text{-CCA}$ -game against $\text{KEM}^\perp := \text{SFO}_{\text{pki},m,c}^\perp[\text{PKE}, \mathbf{G}, \mathbf{H}, \text{len}_{\text{salt}}]$ hence

$$|\Pr[\mathbf{G}_0 \Rightarrow 1]| = \text{Adv}_{\text{KEM}^\perp}^{\text{IND}_{n,u}\text{-CCA}}(\mathbf{B})$$

In $\mathbf{G}_1$ we introduce a flag **COLL** and abort if there is a collision in public key identifiers, which allows each **pki** to be uniquely identified as $F(pk_j)$ for some j . This change is unnoticed as long as there is no collision in public key identifiers. By the birthday bound

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| = \frac{u^2}{2^\ell}$$

where ℓ is the min entropy of $F(pk)$. where ℓ is the min entropy of $F(pk)$ i.e. $\ell = H_\infty(pk)$ where $(pk, sk) \leftarrow \text{KeyGen}$. In many cases public keys are very large, and so even a very large public key identifier, with high entropy, amounts to a significant reduction in overhead, while having negligible impact on security level.

In $\mathbf{G}_2$ we change the experiment so that message/salt pairs are sampled without replacement for each user, since otherwise the same message/salt pair would lead to identical KEM keys, and the adversary could distinguish between the real world and the random world. Since the total number of challenge messages is n , by the birthday bound we have

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}}.$$

In $\mathbf{G}_3$ we patch the random oracles so that **Decaps** and **H** always return the same thing. Even in the event of an correctness error, the view of the adversary is identical, since $\mathcal{O}_{\text{Decaps}}^\perp$ implicitly makes a **CVO** query on lines 33-35, and will return $\perp$ on a correctness error, as in $\mathbf{G}_2$

$\mathbf{G}_0(\mathbf{B})$ <hr/> 1 : $b \xleftarrow{\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$ 4 : $\vec{pk} = (pk_1, \dots, pk_j)$ 5 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^\perp, \mathbf{G}, \mathbf{H}}(\vec{pk})$ 6 : return $\llbracket b = b' \rrbracket$	$\mathbf{G}(j, m \parallel \text{salt})$ <hr/> 15 : if $\exists r$ s.t. $(m \parallel \text{salt}, r) \in \mathfrak{L}_{\mathbf{G}}$ 16 : return r 17 : $r \xleftarrow{\$} \mathcal{R}$ 18 : $\mathfrak{L}_{\mathbf{G}} \leftarrow \mathfrak{L}_{\mathbf{G}} \cup \{(m \parallel \text{salt}, r)\}$ 19 : return r
$\text{Chall}(j)$ <hr/> 7 : $(m, \text{salt}) \xleftarrow{\$} \mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}$ 8 : $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m\}$ 9 : $r \leftarrow \mathbf{G}(m \parallel \text{salt})$ 10 : $c = \text{Enc}(pk_j, m; r)$ 11 : $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c \parallel \text{salt}\}$ 12 : $\text{ss}_0 = \mathbf{H}(j, \mathbf{F}(pk_j), m, c \parallel \text{salt})$ 13 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$ 14 : return ss	$\mathbf{H}(j, \text{pki}, m, c \parallel \text{salt})$ <hr/> 20 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \parallel \text{salt}, \text{ss}) \in \mathfrak{L}_{\mathbf{H}_j}$ 21 : return ss 22 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 23 : $\mathfrak{L}_{\mathbf{H}_j} \leftarrow \mathfrak{L}_{\mathbf{H}_j} \cup \{(\text{pki}, m, c \parallel \text{salt}, \text{ss})\}$ 24 : return ss
	$\mathcal{O}_{\text{Decaps}}^\perp(j, c \parallel \text{salt})$ <hr/> 25 : if $c \in \mathfrak{L}_{C_j}$ 26 : abort 27 : $m' \leftarrow \text{Dec}(sk'_j, c \parallel \text{salt})$ 28 : if $m' = \perp$ or $\text{Enc}(pk_j, m'; \mathbf{G}(j, m' \parallel \text{salt})) \neq c$ 29 : return $\perp$ 30 : $\text{ss} \leftarrow \mathbf{H}(j, \text{pki}, m', c \parallel \text{salt})$ 31 : $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$ 32 : return ss

Figure 4.39: $\mathbf{G}_0$ for the proof of Theorem 4.2.10

$$\Pr[\mathbf{G}_3 \Rightarrow 1] = \Pr[\mathbf{G}_2 \Rightarrow 1]$$

G₁(B)	G(j, m salt)
1 : $b \xleftarrow{\$} \{0, 1\}$	17 : if $\exists r$ s.t. $(m \text{salt}, r) \in \mathcal{L}_G$
2 : for $j = 1, \dots, u$	18 : return r
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	19 : $r \xleftarrow{\$} \mathcal{R}$
4 : $\vec{pk} = (pk_1, \dots, pk_j)$	20 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \text{salt}, r)\}$
5 : if $\exists i, j \in [u]$ s.t. $F(pk_i) = F(pk_j)$	21 : return r
6 : COLL = true; abort	
7 : $b' \leftarrow B^{\text{Chall}, \mathcal{O}_{\text{Decaps}}, \mathcal{L}, G, H}(\vec{pk})$	H(j, pki, m, c salt)
8 : return $\llbracket b = b' \rrbracket$	22 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$
	23 : return ss
Chall(j)	24 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
9 : $(m, \text{salt}) \xleftarrow{\$} \mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}$	25 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(\text{pki}, m, c \text{salt}, \text{ss})\}$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	26 : return ss
11 : $r \leftarrow G(m \text{salt})$	
12 : $c = \text{Enc}(pk_j, m; r)$	$\mathcal{O}_{\text{Decaps}}^\perp(j, c \text{salt})$
13 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \text{salt}\}$	27 : if $c \in \mathcal{L}_{C_j}$
14 : $\text{ss}_0 = H(j, F(pk_j), m, c \text{salt})$	28 : abort
15 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	29 : $m' \leftarrow \text{Dec}(sk'_j, c \text{salt})$
16 : return ss	30 : if $m' = \perp$ or $\text{Enc}(pk_j, m'; G(j, m' \text{salt})) \neq c$
	31 : return $\perp$
	32 : $\text{ss} \leftarrow H(j, \text{pki}, m', c \text{salt})$
	33 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \text{salt}, \text{ss}\}$
	34 : return ss

Figure 4.40: **G₁** for the proof of Theorem 4.2.10

$\mathbf{G}_2(\mathbf{B})$	$\mathbf{G}(j, m \parallel \text{salt})$
1 : $b \xleftarrow{\$} \{0, 1\}$	17 : if $\exists r$ s.t. $(m \parallel \text{salt}, r) \in \mathcal{L}_{\mathbf{G}}$
2 : for $j = 1, \dots, u$	18 : return r
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	19 : $r \xleftarrow{\$} \mathcal{R}$
4 : $\vec{pk} = (pk_1, \dots, pk_j)$	20 : $\mathcal{L}_{\mathbf{G}} \leftarrow \mathcal{L}_{\mathbf{G}} \cup \{(m \parallel \text{salt}, r)\}$
5 : if $\exists i, j \in [u]$ s.t. $\mathbf{F}(pk_i) = \mathbf{F}(pk_j)$	21 : return r
6 : $\text{COLL} = \text{true}$; abort	
7 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^{\perp}, \mathbf{G}, \mathbf{H}}(\vec{pk})$	$\mathbf{H}(j, \text{pki}, m, c \parallel \text{salt})$
8 : return $\llbracket b = b' \rrbracket$	22 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \parallel \text{salt}, \text{ss}) \in \mathcal{L}_{\mathbf{H}_j}$
	23 : return ss
$\mathbf{Chall}(j)$	24 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
9 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}) \setminus (\mathcal{L}_{M_j} \times \mathcal{L}_{S_j})$	25 : $\mathcal{L}_{\mathbf{H}_j} \leftarrow \mathcal{L}_{\mathbf{H}_j} \cup \{(\text{pki}, m, c \parallel \text{salt}, \text{ss})\}$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	26 : return ss
11 : $r \leftarrow \mathbf{G}(m \parallel \text{salt})$	
12 : $c = \text{Enc}(pk_j, m; r)$	$\mathcal{O}_{\text{Decaps}}^{\perp}(j, c \parallel \text{salt})$
13 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \parallel \text{salt}\}$	27 : if $c \in \mathcal{L}_{C_j}$
14 : $\text{ss}_0 \leftarrow \mathbf{H}(j, \mathbf{F}(pk_j), m, c \parallel \text{salt})$	28 : abort
15 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	29 : $m' \leftarrow \text{Dec}(sk_j, c \parallel \text{salt})$
16 : return ss	30 : if $m' = \perp$ or $\text{Enc}(pk_j, m'; \mathbf{G}(j, m' \parallel \text{salt})) \neq c$
	31 : return $\perp$
	32 : $\text{ss} \leftarrow \mathbf{H}(j, \text{pki}, m', c \parallel \text{salt})$
	33 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$
	34 : return ss

Figure 4.41: $\mathbf{G}_2$ for the proof of Theorem 4.2.10.

In $\mathbf{G}_4$, we modify $\mathcal{O}_{\text{Decaps}}^{\perp}$, so that instead of checking that $m' \in \mathcal{M}$, it instead checks that $\mathbf{G}(m' \parallel \text{salt})$ had been queried, and then $\text{Enc}(pk_j, m'; \mathbf{G}(m' \parallel \text{salt})) = c'$. The two games proceed identically, unless $\mathbf{B}$ was able to produce a ciphertext c such that $c = \text{Enc}(pk_j, m'; \mathbf{G}(m'))$ without querying $\mathbf{G}(m')$. In this case in $\mathbf{G}_4$ $\mathcal{O}_{\text{Decaps}}^{\perp}(j, c \parallel \text{salt})$ would return $\perp$, while in $\mathbf{G}_3$ $\mathcal{O}_{\text{Decaps}}^{\perp}(k, c \parallel \text{salt})$ would return $\mathbf{H}(j, \mathbf{F}(pk_j), m', c \parallel \text{salt})$.

G₃(B)	H(j, pki, m, c salt)
1 : $b \xleftarrow{\$} \{0, 1\}$	22 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$
2 : for $j = 1, \dots, u$	23 : return ss
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	24 : if $\text{Enc}_1(pk_j, m) = c$
4 : $\vec{pk} = (pk_1, \dots, pk_j)$	25 : if $\exists \text{ss}'$ s.t. $(\text{pki}, c \text{salt}, \text{ss}') \in \mathcal{L}_{D_j}$
5 : if $\exists i, j \in [u]$ s.t. $F(pk_i) = F(pk_j)$	26 : $\text{ss} := \text{ss}'$
6 : COLL = true; abort	27 : else
7 : $b' \leftarrow \text{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^{\mathcal{L}}, \text{G}, \text{H}}(\vec{pk})$	28 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{\text{pki}, c \text{salt}, \text{ss}\}$
8 : return $\llbracket b = b' \rrbracket$	29 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(\text{pki}, m, c \text{salt}, \text{ss})\}$
Chall(j)	30 : return ss
9 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}) \setminus (\mathcal{L}_{M_j} \times \mathcal{L}_{S_j})$	$\mathcal{O}_{\text{Decaps}}^{\perp}(j, c \text{salt})$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	31 : if $c \in \mathcal{L}_{C_j}$
11 : $r \leftarrow \text{G}(m \text{salt})$	32 : abort
12 : $c = \text{Enc}(pk_j, m; r)$	33 : $m' \leftarrow \text{Dec}_1(sk'_j, c \text{salt})$
13 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \text{salt}\}$	34 : if $m' \notin \mathcal{M}$
14 : $\text{ss}_0 \leftarrow \text{H}(j, F(pk_j), m, c \text{salt})$	35 : return $\perp$
15 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	36 : if $\exists \text{ss}$ s.t. $(\text{pki}, c \text{salt}, \text{ss}) \in \mathcal{L}_{D_j}$
16 : return ss	37 : return ss
G(j, m salt)	38 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
17 : if $\exists r$ s.t. $(m \text{salt}, r) \in \mathcal{L}_G$	39 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \text{salt}, \text{ss}\}$
18 : return r	40 : return ss
19 : $r \xleftarrow{\$} \mathcal{R}$	
20 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \text{salt}, r)\}$	
21 : return r	

Figure 4.42: **G₃** for the proof of Theorem 4.2.10.

$G_4(B)$	$H(j, \text{pki}, m, c \parallel \text{salt})$
1 : $b \xleftarrow{\$} \{0, 1\}$	22 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \parallel \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$
2 : for $j = 1, \dots, u$	23 : return ss
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	24 : if $\text{Enc}_1(pk_j, m) = c$
4 : $\vec{pk} = (pk_1, \dots, pk_j)$	25 : if $\exists \text{ss}'$ s.t. $(\text{pki}, c \parallel \text{salt}, \text{ss}') \in \mathcal{L}_{D_j}$
5 : if $\exists i, j \in [u]$ s.t. $F(pk_i) = F(pk_j)$	26 : $\text{ss} := \text{ss}'$
6 : $\text{COLL} = \text{true}$; abort	27 : else
7 : $b' \leftarrow B^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^{\mathcal{L}}, G, H}(\vec{pk})$	28 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{\text{pki}, c \parallel \text{salt}, \text{ss}\}$
8 : return $\llbracket b = b' \rrbracket$	29 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(\text{pki}, m, c \parallel \text{salt}, \text{ss})\}$
$\text{Chall}(j)$	30 : return ss
9 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}) \setminus (\mathcal{L}_{M_j} \times \mathcal{L}_{S_j})$	$\mathcal{O}_{\text{Decaps}}^\perp(j, c \parallel \text{salt})$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	31 : if $c \in \mathcal{L}_{C_j}$
11 : $r \leftarrow G(m \parallel \text{salt})$	32 : abort
12 : $c = \text{Enc}(pk_j, m; r)$	33 : $m' \leftarrow \text{Dec}_1(sk'_j, c \parallel \text{salt})$
13 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \parallel \text{salt}\}$	34 : if $\nexists r$, s.t. $(m', r) \in \mathcal{L}_{G_j}$ s.t. $\text{Enc}(pk, m; r \parallel \text{salt}) = c$
14 : $\text{ss}_0 \leftarrow H(j, F(pk_j), m, c \parallel \text{salt})$	35 : return $\perp$
15 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	36 : if $\exists \text{ss}$ s.t. $(\text{pki}, c \parallel \text{salt}, \text{ss}) \in \mathcal{L}_{D_j}$
16 : return ss	37 : return ss
$G(j, m \parallel \text{salt})$	38 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
17 : if $\exists r$ s.t. $(m \parallel \text{salt}, r) \in \mathcal{L}_G$	39 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$
18 : return r	40 : return ss
19 : $r \xleftarrow{\$} \mathcal{R}$	
20 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \parallel \text{salt}, r)\}$	
21 : return r	

Figure 4.43: G_4 for the proof of Theorem 4.2.10.

This implies that B found $r \neq G(m' \parallel \text{salt})$, such that $\text{Enc}(pk, m; G(m' \parallel \text{salt})) = \text{Enc}(pk, m; r)$. For a single decapsulation query, this occurs with probability $2^{-\gamma}$, by the assumption that

PKE is γ -spread. For q_D decapsulation queries, we have that

$$\Pr[\mathbf{G4} \Rightarrow 1] - \Pr[\mathbf{G3} \Rightarrow 1] \leq q_D \cdot 2^{-\gamma}$$

in $\mathbf{G}_5$ we remove all dependence on the secret key, by no longer checking decryption c during decapsulation. Instead, we check for queries to $\mathbf{G}(j, -)$ which encrypt to c . This change goes unnoticed, unless the adversary was able to query a message and a ciphertext which exhibit a correctness error. Since there are $q_G + q_D + q_H$ implicit and explicit queries to $\mathbf{G}$ we have

$$\Pr[\mathbf{G5} \Rightarrow 1] - \Pr[\mathbf{G4} \Rightarrow 1] \leq (q_D + q_H + q_G) \cdot \delta(u)$$

In $\mathbf{G}_6$ we raise a flag CHAL and abort if the adversary is able to correctly produce a KEM key that would match a challenge ciphertext. That is, we abort if the adversary queries $\mathbf{H}(\text{pki}, j, m, c \parallel \text{salt})$ or $\mathbf{G}(j, m \parallel \text{salt})$ where $c \in \mathcal{L}_{C_j}$ and $\text{Enc}(pk_j, m; \mathbf{G}(m; \text{salt})) = c$. We also change **Chall** so that it samples r at random, rather than fixing $r \leftarrow \mathbf{G}(m)$. This allows a CPA adversary to simulate CHAL in $\mathbf{G}_5$, using their own challenge oracle for PKE. Provided CHAL is not raised, this change goes unnoticed since $\mathbf{B}$ is not given access to $\mathbf{G}(m)$. We have

$$|\Pr[\mathbf{G5} \Rightarrow 1] - \Pr[\mathbf{G4} \Rightarrow 1]| \leq \Pr[\text{CHAL}]$$

Furthermore, in $\mathbf{G}_5$ the adversary is provided with random shared secrete independent of the bit b . Then we have

$$|\Pr[\mathbf{G5} \Rightarrow 1]| = \frac{1}{2}$$

It remains to bound CHAL. To this end, we construct an $\text{IND}_{n,u}\text{-CPA}$ adversary $\mathbf{A}$ against PKE, that perfectly simulates $\mathbf{G}_5$ for $\mathbf{B}$, and wins whenever CHAL is raised.

We introduce a shorthand notation here. Let $\mathcal{MS}_{\text{new}_{j,b}} = \mathcal{M} \times 2^{\text{len}_{\text{salt}}} \setminus (\mathcal{L}_{M_{j,b}} \times \mathcal{L}_{S_j})$. We claim that $\mathbf{A}$ simulates $\mathbf{G}_5$ for $\mathbf{B}$. To simulate CHAL for b , $\mathbf{A}$ samples $\text{salt} \leftarrow \{0, 1\}^{\text{len}_{\text{salt}}}$, and appends salt to $\mathcal{L}_{S_j}$. Then $\mathbf{A}$ samples $m_0 \leftarrow \mathcal{MS}_{\text{new}_{j,0}}$ and appends $m_0 \parallel \text{salt}$ to $\mathcal{L}_{M_{j,0}}$, and samples $m_1 \leftarrow \mathcal{MS}_{\text{new}_{j,1}}$ and appends $m_1 \parallel \text{salt}$ to $\mathcal{L}_{M_{j,1}}$. Then $\mathbf{A}$ queries its own **Chall** on input (j, m_0, m_1) to obtain $c = \text{Enc}(pk, m_b)$ and appends $c \parallel \text{salt}$ to $\mathcal{L}_{C_j}$. $\mathbf{A}$ samples a random $\text{ss} \xleftarrow{\$} \mathcal{K}$, and returns $(c \parallel \text{salt}, K)$ to $\mathbf{B}$, which matches the output of **Chall** in $\mathbf{G}_5$. Observe that $\mathcal{O}_{\text{Decaps}}^\times$ has been modified to no longer make use of sk_j , and $\mathbf{H}$ is patched so that its output always matches that of $\mathcal{O}_{\text{Decaps}}^\times$. Hence the simulation is perfect.

G₅(B)	H(j, pki, m, c salt)
1 : $b \xleftarrow{\$} \{0, 1\}$	22 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \text{salt}, \text{ss}) \in \mathcal{L}_{H_j}$
2 : for $j = 1, \dots, u$	23 : return ss
3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$	24 : if $\text{Enc}_1(pk_j, m) = c$
4 : $\vec{pk} = (pk_1, \dots, pk_j)$	25 : if $\exists \text{ss}'$ s.t. $(\text{pki}, c \text{salt}, \text{ss}') \in \mathcal{L}_{D_j}$
5 : if $\exists i, j \in [u]$ s.t. $F(pk_i) = F(pk_j)$	26 : $\text{ss} \leftarrow \text{ss}'$
6 : $\text{COLL} = \text{true};$ abort	27 : else
7 : $b' \leftarrow \text{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^{\mathcal{L}}, \text{G}, \text{H}}(\vec{pk})$	28 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{\text{pki}, c \text{salt}, \text{ss}\}$
8 : return $\llbracket b = b' \rrbracket$	29 : $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{(\text{pki}, m, c \text{salt}, \text{ss})\}$
Chall(j)	30 : return ss
9 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}) \setminus (\mathcal{L}_{M_j} \times \mathcal{L}_{S_j})$	$\mathcal{O}_{\text{Decaps}}^{\perp}(j, c \text{salt})$
10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$	31 : if $c \in \mathcal{L}_{C_j}$
11 : $r \leftarrow \text{G}(m \text{salt})$	32 : abort
12 : $c = \text{Enc}(pk_j, m; r)$	33 : if $\nexists r$, s.t. $(m, r) \in \mathcal{L}_{G_j}$
13 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \text{salt}\}$	s.t. $\text{Enc}(pk_j, m; r \text{salt}) = c$
14 : $\text{ss}_0 \leftarrow \text{H}(j, F(pk_j), m, c \text{salt})$	34 : return $\perp$
15 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$	35 : if $\exists \text{ss}$ s.t. $(\text{pki}, c \text{salt}, \text{ss}) \in \mathcal{L}_{D_j}$
16 : return ss	36 : return ss
G(j, m salt)	37 : $\text{ss} \xleftarrow{\$} \mathcal{K}$
17 : if $\exists r$ s.t. $(m \text{salt}, r) \in \mathcal{L}_G$	38 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \text{salt}, \text{ss}\}$
18 : return r	39 : return ss
19 : $r \xleftarrow{\$} \mathcal{R}$	
20 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \text{salt}, r)\}$	
21 : return r	

Figure 4.44: **G₅** for the proof of Theorem 4.2.10.

$\mathbf{G}_6(\mathbf{B})$ <hr/> 1 : $b \xleftarrow{\$} \{0, 1\}$ 2 : for $j = 1, \dots, u$ 3 : $(pk_j, sk_j) \xleftarrow{\$} \text{KeyGen}()$ 4 : $\vec{pk} = (pk_1, \dots, pk_j)$ 5 : if $\exists i, j \in [u]$ s.t. $F(pk_i) = F(pk_j)$ 6 : $\text{COLL} = \text{true}$; abort 7 : $b' \leftarrow \mathbf{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^{\mathcal{L}}, \mathbf{G}, \mathbf{H}}(\vec{pk})$ 8 : return $\llbracket b = b' \rrbracket$	$\mathbf{H}(j, \text{pki}, m, c \parallel \text{salt})$ <hr/> 25 : if $\exists \text{ss}$ s.t. $(\text{pki}, m, c \parallel \text{salt}, \text{ss}) \in \mathcal{L}_{\text{H}_j}$ 26 : return ss 27 : if $m \in \mathcal{L}_{M_j}$ 28 : $\text{CHAL} = \text{true}$ 29 : abort 30 : if $\text{Enc}_1(pk_j, m) = c$ 31 : if $\exists \text{ss}'$ s.t. $(\text{pki}, c \parallel \text{salt}, \text{ss}') \in \mathcal{L}_{D_j}$ 32 : $\text{ss} := \text{ss}'$ 33 : else 34 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{\text{pki}, c \parallel \text{salt}, \text{ss}\}$ 35 : $\mathcal{L}_{\text{H}_j} \leftarrow \mathcal{L}_{\text{H}_j} \cup \{(\text{pki}, m, c \parallel \text{salt}, \text{ss})\}$ 36 : return ss
$\text{Chall}(j)$ <hr/> 9 : $(m, \text{salt}) \xleftarrow{\$} (\mathcal{M} \times \{0, 1\}^{\text{len}_{\text{salt}}}) \setminus (\mathcal{L}_{M_j} \times \mathcal{L}_{S_j})$ 10 : $\mathcal{L}_{M_j} \leftarrow \mathcal{L}_{M_j} \cup \{m\}$ 11 : $r \leftarrow \mathbb{R}$ 12 : $c = \text{Enc}(pk_j, m; r)$ 13 : $\mathcal{L}_{C_j} \leftarrow \mathcal{L}_{C_j} \cup \{c \parallel \text{salt}\}$ 14 : $\text{ss}_0 \xleftarrow{\$} \mathcal{K}$ 15 : $\text{ss}_1 \xleftarrow{\$} \mathcal{K}$ 16 : return ss	$\mathcal{O}_{\text{Decaps}}^{\perp}(j, c \parallel \text{salt})$ <hr/> 37 : if $c \in \mathcal{L}_{C_j}$ 38 : abort 39 : if $\nexists r$, s.t. $(m, r) \in \mathcal{L}_{G_j}$ 40 : s.t. $\text{Enc}(pk, m; r \parallel \text{salt}) = c$ 41 : return $\perp$ 42 : if $\exists \text{ss}$ s.t. $(\text{pki}, c \parallel \text{salt}, \text{ss}) \in \mathcal{L}_{D_j}$ 43 : return ss 44 : $\text{ss} \xleftarrow{\$} \mathcal{K}$ 45 : $\mathcal{L}_{D_j} \leftarrow \mathcal{L}_{D_j} \cup \{c \parallel \text{salt}, \text{ss}\}$ 46 : return ss
$\mathbf{G}(j, m \parallel \text{salt})$ <hr/> 17 : if $\exists r$ s.t. $(m \parallel \text{salt}, r) \in \mathcal{L}_G$ 18 : return r 19 : $r \xleftarrow{\$} \mathcal{R}$ 20 : if $m \in \mathcal{L}_{M_j}$ 21 : $\text{CHAL} = \text{true}$ 22 : abort 23 : $\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m \parallel \text{salt}, r)\}$ 24 : return r	

Figure 4.45: $\mathbf{G}_6$ for the proof of Theorem 4.2.10.

$A(\vec{pk})$	
1 :	$\tilde{b} \leftarrow \mathcal{B}^{\text{Chall}, \mathcal{O}_{\text{Decaps}}^\perp, \text{H}, \text{G}}(\vec{pk})$
	if $\exists (m_0 \parallel \text{salt}) \in \mathcal{L}_{M_{j,0}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_{G_j})$
2 :	or $(F(pk_j), m, c \parallel \text{salt}) \in \mathcal{L}_{H_j}$
3 :	if $\exists (m_1 \parallel \text{salt}) \in \mathcal{L}_{M_{j,1}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_{G_j} \text{ or } (F(pk_j), m, c \parallel \text{salt}) \in \mathcal{L}_{H_j})$
4 :	$b' \xleftarrow{\$} \{0, 1\}$
5 :	else $b' := 0$
6 :	if $\exists (m_1 \parallel \text{salt}) \in \mathcal{L}_{M_{j,1}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_{G_j} \text{ or } (F(pk_j), m, c \parallel \text{salt}) \in \mathcal{L}_{H_j})$
7 :	if $\exists (m_0 \parallel \text{salt}) \in \mathcal{L}_{M_{j,0}}$ and $((m \parallel \text{salt}, r) \in \mathcal{L}_{G_j} \text{ or } (F(pk_j), m, c \parallel \text{salt}) \in \mathcal{L}_{H_j})$
8 :	$b' \xleftarrow{\$} \{0, 1\}$
9 :	else $b' := 1$
10 :	else $b' \xleftarrow{\$} \{0, 1\}$
11 :	return $\llbracket b = b' \rrbracket$

Figure 4.46: Adversaries B_1 and B_2 for the proof of Theorem 4.2.10.

Let BADG be the event that B queries G on any $(j, m \parallel \text{salt})$ or $H(F(pk_j), m, c \parallel \text{salt})$ with $m \parallel \text{salt} \in \mathcal{L}_{M_{j,1-b}}$. Then

$$\Pr[\text{BADG}] = \max_{j \in [u]} \frac{n_j \cdot (q_H + q_G)}{|\mathcal{M}| |\mathcal{L}_{S_j}|}.$$

This imposes a trade-off on B . As n_j increases, $|\mathcal{L}_{S_j}|$ increases proportionately. The relative size of $\mathcal{L}_{S_j}$ decreases with the number of collisions in salt values, and is therefore minimized when $n_j = n$.

Consider the advantage of A against the $\text{IND}_{n,u}\text{-CPA}$ security of PKE. D looks for random oracle queries containing $m_0 \in \mathcal{L}_{M_{j,0}}$ or $m_1 \in \mathcal{L}_{M_{j,1}}$. If one is found, this indicates that either the flag CHAL or BADG was raised. If CHAL is raised, and BADG is not, then A detects a random oracle query containing $m \in \mathcal{L}_{M_{j,b}}$. If neither CHAL or BADG is raised, A returns a random guess, as is the case when both CHAL and BADG are raised. Then provided BADG is not raised, A wins the game with probability 1 if CHAL is raised, and with probability $\frac{1}{2}$ otherwise. Hence

$$\begin{aligned}
\text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A}) + \Pr[\text{BADG}] &\geq \left| \Pr[b = b' | \neg \text{BADG}] - \frac{1}{2} \right| \\
&= \left| \Pr[\text{CHAL}] + \frac{1}{2} \Pr[\neg \text{CHAL}] - \frac{1}{2} \right| \\
&= \frac{1}{2} \Pr[\text{CHAL}]
\end{aligned}$$

And so

$$\Pr[\text{CHAL}] \leq \max_{j \in [u]} \frac{(2n_j \cdot q_{\mathbf{H}} + q_{\mathbf{G}})}{|\mathcal{M}| |\mathfrak{L}_{S_j}|} + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$$

Adding all inequalities gives

$$\begin{aligned}
\text{Adv}_{\text{KEM}^\perp}^{\text{IND}_{n,u}\text{-CCA}}(\mathbf{B}) &\leq \frac{u^2}{2^\ell} + \frac{n^2}{|\mathcal{M}| 2^{\text{len}_{\text{salt}}}} + \frac{2n(q_{\mathbf{G}} + q_{\mathbf{H}})}{|\mathfrak{L}_{S_j}| |\mathcal{M}|} + q_D \cdot 2^{-\gamma} \\
&\quad + (q_{\mathbf{G}} + q_{\mathbf{H}} + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})
\end{aligned}$$

as desired. $\square$

4.2.8 Tightness for FrodoKEM Parameters

Now we briefly consider each intermediate term in the above security bounds, using the parameters **FrodoKEM640**, and assuming $n = 2^{64}$, which we consider to be a reasonable threshold.

- The term $\frac{n^2}{|\mathcal{M}| 2^{\text{len}_{\text{salt}}}} = \frac{1}{2^{64}}$, we have $T = 2^{64}$ and $\epsilon = \frac{1}{2^{64}}$, satisfying 128 bits of security.
- The term $\max_{j \in [u]} \frac{2n_j(q_{\mathbf{G}} + q_{\mathbf{H}})}{|\mathcal{M}| |\mathfrak{L}_{S_j}|}$ satisfies tightness requirements since $\max_j \frac{n_j}{|\mathfrak{L}_{S_j}| < \frac{1}{2^{63}}}$.
- We cannot give an exact figure for $\delta(u)$ in order to bound the term $(q_{\mathbf{G}} + q_{\mathbf{H}} + q_D) \cdot \delta(u) + 2 \cdot$ for **Frodo640**, as this would require experimental data. In general, we expect the multi-user correctness term $\delta(u)$ to be close to the single user correctness term δ , which for **Frodo** was experimentally tested to be $2^{-138.7}$.

The monolithic bound is marginally tighter than the bound you would obtain from combining those obtained from ST and $\mathbf{U}_{\mathbf{F}(pk)}^{m,c}$ separately. In particular, the coefficient of $\text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}$ is 2, rather than 3, the entropy bound is $\frac{u^2}{2^\ell}$, rather than $\frac{u^2}{2^{\ell-1}}$. Monolithic bounds for the $\text{IND}_{n,u}\text{-CCA}$ security for each FO that we considered are shown in Figure 4.40. All the following bounds can be obtained from theorem 4.2.9 and 4.2.10. For example, the bound for $\text{FO}_{pk,m,c}^\perp$ can be obtained from theorem 4.2.9, by letting F be the identity function, and treating the empty string as the only salt value, and thus having $|\mathcal{L}_S| = 1$. For example, the bound for $\text{FO}_{pk,m,c}^\perp$ can be obtained from theorem 4.2.10, by letting F be the identity function, $|\mathcal{L}_S| = 1$, by treating the empty string as the only used salt.

Without public key/prefix hashing, we are unaware of a way to improve on the bound implied by [BBM00], which would be the single-target, multi-target bounds multiplied by a factor of u .

Remark. The above proofs use two hash functions G for generating randomness, and H for generating shared secrets. In practice, a single hash function such as SHAKE will be used. Some consideration needs to be given to *domain separation*. The idea is that we should be able to model G and H using a single hash function, so long as there is no overlap between their respective domains. Failure to separate domains can lead to security issues, as shown in [BDG20], which also gives techniques for domain separation. In some cases, domain separation might be free, as inputs to G and H vary sufficiently in length that there is no overlap. In other cases, as in [DHK⁺21], more careful consideration is required.

4.3 Modelling a Quantum Adversary

Even though there is no quantum algorithm for LWE at the time of writing, this does not mean that an adversary with access to a quantum computer has the same advantage as a classical adversary. Notably, in the classical setting, if an event occurs only if an adversary queries a uniformly random value, the probability of this event is given by a simple guessing bound. This is not the case in the quantum setting.

FO Variant	$\text{Adv}_{\text{KEM}}^{\text{IND}_{n,u}\text{-CCA}}$
$\text{FO}_{pk,m,c}^\chi$	$\frac{2n(q_H + q_G + n) + q_H}{ \mathcal{M} } + (q_G + q_H + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$
$\text{FO}_{pk,m,c}^\perp$	$\frac{2n(q_H + q_G + n)}{ \mathcal{M} } + q_D \cdot 2^{-\gamma} + (q_G + q_H + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$
$\text{FO}_{F(pk),m,c}^\chi$	$\frac{u^2}{2^\ell} + \frac{2n(q_H + q_G + n) + q_H}{ \mathcal{M} } + (q_G + q_H + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$
$\text{FO}_{F(pk),m,c}^\perp$	$\frac{u^2}{2^\ell} + \frac{2n(q_H + q_G + n)}{ \mathcal{M} } + q_D \cdot 2^{-\gamma} + (q_G + q_H + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$
$\text{SFO}_{pk,m,c}^\chi$	$\frac{n^2}{ \mathcal{M} 2^{\text{len}_{\text{salt}}}} + \frac{2n(q_G + q_H)}{ \mathfrak{L}_S \mathcal{M} } + \frac{q_H}{ \mathcal{M} } + (q_G + q_H + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$
$\text{SFO}_{pk,m,c}^\perp$	$\frac{n^2}{ \mathcal{M} 2^{\text{len}_{\text{salt}}}} + \frac{2n(q_G + q_H)}{ \mathfrak{L}_S \mathcal{M} } + q_D \cdot 2^{-\gamma} + (q_G + q_H + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$
$\text{SFO}_{F(pk),m,c}^\chi$	$\frac{u^2}{2^\ell} + \frac{n^2}{ \mathcal{M} 2^{\text{len}_{\text{salt}}}} + \frac{2n(q_G + q_H)}{ \mathfrak{L}_S \mathcal{M} } + \frac{q_H}{ \mathcal{M} } + (q_G + q_H + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$
$\text{SFO}_{F(pk),m,c}^\perp$	$\frac{u^2}{2^\ell} + \frac{n^2}{ \mathcal{M} 2^{\text{len}_{\text{salt}}}} + \frac{2n(q_G + q_H)}{ \mathfrak{L}_S \mathcal{M} } + q_D \cdot 2^{-\gamma} + (q_G + q_H + q_D) \cdot \delta(u) + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(\mathbf{A})$

Figure 4.47: Security bounds for various versions of the FO transform

A thorough treatment of the quantum random oracle model is outside of the scope of this project, and would constitute a chapter in itself. Aside from extending the previous bounds to the QROM, we would have no contributions to make that would warrant such a chapter.

However, using the same proof steps as in Theorem 3.2 in [DHK⁺21], and making the necessary adjustments to incorporate salts, we can obtain the following result.

Theorem 4.3.1. *Let PKE be a generic public key encryption scheme. For any quantum adversary $\mathbf{B}$ against the $\text{IND}_{n,u}\text{-CCA}$ security of $\text{KEM}^\chi := \text{SFO}[\text{PKE}, \mathbf{G}, \mathbf{H}, \text{len}_{\text{salt}}]$, there*

exists a quantum adversary A against the $\text{IND}_{n,u}$ -CPA security of PKE, with roughly the same running time, such that

$$\begin{aligned} \text{Adv}_{\text{KEM}^\chi}^{\text{IND}_{n,u}\text{-CCA}}(B) \leq & \frac{u^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}} + 4(q_G + q_H + q_D + 1) \sqrt{\frac{u \cdot n}{|\mathcal{M}||\mathcal{L}_S|}} + 16(q_G + q_H + q_D + 1)^2 \cdot \delta(u) \\ & + 4(q_H + 1) \sqrt{\frac{u}{|\mathcal{M}|}} + 2\sqrt{(q_G + q_H + q_D + 1) \text{Adv}_{\text{PKE}}^{\text{IND}_{n,u}\text{-CPA}}(A)} \end{aligned}$$

Proof. We do not provide an entire proof here. Instead we refer the reader to the proof of Theorem 3.2 in [DHK⁺21], with the following changes:

Syntactical changes:

- Wherever n is used, replace with u .
- Wherever q_C is used, replace with n .
- We make use of two separate hash functions G for generating random coins, and H for generating shared secrets, while a single hash function F is used in [DHK⁺21].
- q_F will be replaced with either q_H , q_G or $q_H + q_G$ depending on the context.

Logical changes:

- rather than s_j being a uniformly random secret of length λ we have $s_j \xleftarrow{\$} \mathcal{M}$.³
- In $\mathbf{G}_2$ we only need to sample message/salt pairs without replacement, rather than sampling messages without replacement. Hence we have

$$\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1] = \frac{n^2}{|\mathcal{M}|2^{\text{len}_{\text{salt}}}}$$

- In $\mathbf{G}_3$ we still make use of lemma 5.4 of [DHK⁺21]. However, instead of identifying H with K , and H_j with K'_j and p_j with $F(pk_j)$, we identify $H_1, \dots, H_u$ with

³Sampling s_j from $\{0,1\}^\lambda$, and using a separate internal random oracle K for generating rejected encapsulations was necessary in [DHK⁺21] for domain separation, but is not necessary here when we are using separate random oracles G and H .

$H(1, -), \dots, H(u, -)$, and identify p_j with pk_j . Furthermore we have that λ is given by Finally, q is given by q_H . And so lemma 5.4 gives that

$$\Pr[\mathbf{G3} \Rightarrow 1] - \Pr[\mathbf{G2} \Rightarrow 1] \leq 4(q_H + 1) \sqrt{\frac{u}{|M|}}$$

- The logic of $\mathbf{G}_4 - \mathbf{G}_8$ is unchanged. We only make the syntactical changes described above.
- In $\mathbf{G}_8$ to $\mathbf{G}_{10}$, the addition of salts slightly changes $\Pr[\text{FIND}_1]$ and $\Pr[[\text{FIND}]_2]$, multiplying the denominators by $|\mathcal{L}_S|$.

□

4.4 FrodoKEM

4.4.1 FrodoKEM Specifications

In this subsection, we present an up-to-date version FrodoKEM, an $\text{IND}_{n,u}\text{-CCA}$ -secure KEM, that utilizes the $\text{SFO}^\times$, to mitigate the known vulnerabilities of ephemeral FrodoKEM (eFrodoKEM). In section 3, we showed the specifications for eFrodoKEM. By using SFO instead of FO on FrodoPKE, we change only encapsulation and decapsulation. FrodoPKE and FrodoKEM.KeyGen, as well as all of the auxiliary algorithms: Frodo.Gen, Frodo.encode, Frodo.decode, Frodo.SampleMatrix, Frodo.pack and Frodo.unpack are all as presented in [Chapter 3](#) and [\[ABD⁺23a\]](#).

FrodoKEM.Encaps(seed _A b)	
1 :	$m \xleftarrow{\$} \{0, 1\}^{\text{len}_m}$
2 :	$\text{salt} \xleftarrow{\$} \{0, 1\}^{\text{len}_{\text{salt}}}$
3 :	$\text{pkh} \leftarrow \text{SHAKE}(pk, \text{len}_{\text{pkh}})$
4 :	$\text{seed}_{\text{SE}} \mathbf{k} \leftarrow \text{SHAKE}(\text{pkh}, m \text{salt}, \text{len}_{\text{seed}_{\text{SE}}} + \text{len}_{\mathbf{k}})$
5 :	$(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \dots, \mathbf{r}^{(2\overline{m}n + \overline{m}\overline{n} - 1)}) \leftarrow \text{SHAKE}(0 \times 96 \text{seed}_{\text{SE}}, (2\overline{m}n + \overline{m}\overline{n}) \cdot \text{len}_{\chi})$
6 :	$\mathbf{S}' \leftarrow \text{Frodo.SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \dots, \mathbf{r}^{\overline{m}n - 1}), \overline{m}, n, T_{\chi})$
7 :	$\mathbf{E}' \leftarrow \text{Frodo.SampleMatrix}((\mathbf{r}^{(\overline{m}n)}, \mathbf{r}^{(\overline{m}n + 1)}, \dots, \mathbf{r}^{2\overline{m}n - 1}), \overline{m}, n, T_{\chi})$
8 :	$\mathbf{A} \leftarrow \text{Frodo.Gen}(\text{seed}_{\mathbf{A}})$
9 :	$\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
10 :	$\mathbf{c}_1 \leftarrow \text{Frodo.pack}(\mathbf{B}')$
11 :	$\mathbf{E}'' \leftarrow \text{SampleMatrix}(\mathbf{r}^{(2\overline{m}n)}, \mathbf{r}^{(2\overline{m}n + 1)}, \dots, \mathbf{r}^{2\overline{m}n + \overline{m}\overline{n} - 1}), \overline{m}, \overline{n}, T_{\chi})$
12 :	$\mathbf{B} \leftarrow \text{Frodo.unpack}(\mathbf{b})$
13 :	$\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
14 :	$\mathbf{C} \leftarrow \mathbf{V} + \text{Frodo.encode}(m)$
15 :	$\mathbf{c} \leftarrow \text{Frodo.pack}(\mathbf{C})$
16 :	$\text{ss} \leftarrow \text{SHAKE}(\mathbf{c}_1, \mathbf{c}_2 \text{salt} \mathbf{k})$
17 :	return ($\mathbf{c}_1 \mathbf{c}_2 \text{salt}, \text{ss}$)

Figure 4.48: Encapsulation for FrodoKEM.

FrodoKEM has 3 variants, targeting each of the NIST security levels: FrodoKEM640, FrodoKEM976 and FrodoKEM1344. The parameters for each are given in Figure 4.50.

FrodoKEM.Decaps($c_1 \ c_2 \ \text{salt}, (s \ \text{seed}_A \ b, S^T, \text{pkh})$)	
1 :	$B' \leftarrow \text{Frodo.unpack}(c_1, \bar{m}, n)$
2 :	$C \leftarrow \text{Frodo.unpack}(c_1, \bar{m}, \bar{n})$
3 :	$M \leftarrow C - B'S$
4 :	$m' \leftarrow \text{Frodo.decode}(M)$
5 :	$\text{seed}'_{SE} \ k' \leftarrow \text{SHAKE}(\text{pkh}, m', \text{len}_{\text{seed}_{SE}} + \text{len}_k)$
6 :	$r^{(0)}, r^{(1)}, \dots, r^{(2\bar{m}n + \bar{m}\bar{n} - 1)} \leftarrow \text{SHAKE}(0x96 \ \text{seed}_{SE}, (2\bar{m}n + \bar{m}\bar{n}) \cdot \text{len}_\chi)$
7 :	$S' \leftarrow \text{Frodo.SampleMatrix}((r^{(0)}, r^{(1)}, \dots, r^{\bar{m}n - 1}), \bar{m}, n, T_\chi)$
8 :	$E' \leftarrow \text{Frodo.SampleMatrix}((r^{(\bar{m}n)}, r^{(\bar{m}n + 1)}, \dots, r^{2\bar{m}n - 1}), \bar{m}, n, T_\chi)$
9 :	$A \leftarrow \text{Frodo.Gen}(\text{seed}_A)$
10 :	$B'' \leftarrow S'A + E'$
11 :	$E'' \leftarrow \text{SampleMatrix}(r^{(2\bar{m}n)}, r^{(2\bar{m}n + 1)}, \dots, r^{2\bar{m}n + \bar{m}\bar{n} - 1}), \bar{m}, \bar{n}, T_\chi)$
12 :	$V \leftarrow S'B + E''$
13 :	$C' \leftarrow V + \text{Frodo.encode}(m')$
14 :	if $B' \ C = B'' \ C'$
15 :	$\bar{k} := k$
16 :	else
17 :	$\bar{k} := s$
18 :	$ss = \text{SHAKE}(c_1 \ c_2 \ \text{salt} \ \bar{k} \ \text{len}_{ss})$
19 :	return ss

Figure 4.49: Decapsulation for FrodoKEM.

FrodoKEM Parameters				
FrodoKEM Variant	secret key <i>sk</i>	public key <i>pk</i>	ciphertext <i>c</i>	Shared Secret <i>ss</i>
FrodoKEM640	19,888	9,616	9752	16
FrodoKEM976	31,296	15,632	15,792	24
FrodoKEM1344	43,088	21,520	21,696	32
eFrodoKEM640	19,888	9,616	9720	16
eFrodoKEM976	31,296	15,632	15,744	24
eFrodoKEM – 344	43,088	21,520	21,632	32

Figure 4.50: Size in bytes of inputs and outputs in each variant of FrodoKEM.

4.4.2 Efficiency Analysis

In this subsection, we consider the computational cost of using SFO^λ over FO^λ , as well as small efficiency improvements. We tested 3 different variants of FrodoKEM: eFrodoKEM, the unsalted version which only targets single target security, FrodoKEM, which is constructed by applying $\text{SFO}_{\text{pkh},m,c}^\lambda$ to FrodoPKE, and FrodoKEM_{*m*}, a version of FrodoKEM which removes the ciphertext from the hash input. Our experiment showed little difference between the first two. Following [BHH⁺19], applying $\text{SFO}_{\text{pkh},m}^\lambda$ rather than $\text{SFO}_{\text{pkh},m,c}^\lambda$ has no impact on security. Removing *c* from the hash input has some impact on efficiency, however. We note that the efficiency improvements suggested in [DHK⁺21] for Crystals-Kyber and Saber, are already implemented in FrodoKEM, which uses a 128 bit public key hash, rather than the entire public key. The relevant comparison in Figure 4.31 is the average time in microseconds for encapsulation and decapsulation for each security level.

The following experimental data was obtained on a 2022 Macbook pro using MacOS Sonoma 14.4.1, with an M2 processor and 8GB of RAM.

FrodoKEM Efficiency				
NIST Security Level		eFrodoKEM _{<i>pkh,m,c</i>}	FrodoKEM _{<i>pkh,m,c</i>}	FrodoKEM _{<i>pkh,m</i>}
1	KeyGen	234.281	238.149	240.814
	Encaps	279.495	279.381	266.300
	Decaps	270.542	271.395	258.119
2	KeyGen	470.520	472.838	474.927
	Encaps	507.693	510.257	484.617
	Decaps	479.453	489.805	456.875
3	KeyGen	705.170	702.292	704.511
	Encaps	779.810	778.418	742.617
	Decaps	739.910	745.915	705.027

Figure 4.51: Time in microseconds for an iteration of FrodoKEM for various versions, using SHAKE.

Chapter 5

Conclusion

In this work we introduced the salted Fujisaki-Okamoto transform, and proved that it enables a tight reduction for $\text{IND}_{n,u}\text{-CCA}$ security of the resultant KEM, to the $\text{IND}_{n,u}\text{-CPA}$ security of the underlying PKE. The general idea was that in multi-target attacks with large n , an adversary can break $\text{IND}_{n,u}\text{-CCA}$ security of a KEM, by computing a large number of ciphertexts, and searching for a collision. Probabilistic encryption from the T transform implies that a collision between a challenge ciphertext, and one computed by the adversary, means both ciphertexts will have the same underlying plaintext, breaking $\text{OW}_{n,u}\text{-PCA}$ security, and therefore $\text{IND}_{n,u}\text{-CCA}$ security as well. The incorporation of salts means that encryption is no longer deterministic, and identical ciphertexts no longer means identical messages.

We applied these results to **FrodoKEM**, showing that the most recent version, which uses the **SFO**-transform, is $\text{IND}_{n,u}\text{-CCA}$ secure, while a previous version **eFrodoKEM** which used the generic **FO**-transform, was not. We found that our incorporation of salting improved the tightness gap by a factor of 2^{63} , even when an adversary performs an attack with runtime $T = 2^{65}$. In particular, with **eFrodoKEM**, an attack with time $T = 2^{65}$ had $\epsilon = 1$, while with **FrodoKEM** the same attack has $\epsilon = \frac{1}{2^{63}}$.

5.1 Comparison to Existing Work

This work is closely related to “Faster Lattice-Based KEMs via a Generic Fujisaki-Okamoto Transform Using Prefix Hashing” by Duman et al. That work assesses the **FO**-transform in the multi-target setting. The proofs in this work follow a very similar structure to

theorem 3.1 of that work. [DHK⁺21] is itself a continuation of “A Modular Analysis of the Fujisaki-Okamoto Transformation” by Hövelmanns et al. [HHK17] shows that there is a tight reduction from IND-CCA security of a KEM: FO[PKE] tightly reduces to the IND-CPA security of PKE. We have shown this is not true for IND_{*n,u*}-CCA security, for the generic FO-transform. Our results expand on these works by analyzing the impact of salting, and providing a proof for KEMs with explicit rejection.

5.2 Future Work

The bounds we gave are all reductions to the IND_{*n,u*}-CPA security of a PKE to which the FO or SFO transform has been applied. While in the case of FrodoKEM, both IND_{*n,u*}-CPA and IND-CPA security of FrodoPKE reduce to LWE, it is still the case that

$$\text{Adv}_{\text{FrodoPKE}}^{\text{IND-CPA}} < \text{Adv}_{\text{FrodoPKE}}^{\text{IND}_{n,u}\text{-CPA}}.$$

We have not quantified the difference between $\text{Adv}_{\text{FrodoPKE}}^{\text{IND-CPA}}$ and $\text{Adv}_{\text{FrodoPKE}}^{\text{IND}_{n,u}\text{-CPA}}$, in order to bound $\text{Adv}_{\text{FrodoPKE}}^{\text{IND-CPA}}$ directly in terms of LWE. In [Ber22], Daniel Bernstein quantifies the asymptotic impact of multiple ciphertexts for lattice-based schemes. It remains to incorporate, or, if necessary, expand on, these ideas in order to quantify the difference between $\text{Adv}_{\text{FrodoPKE}}^{\text{IND-CPA}}$ and $\text{Adv}_{\text{FrodoPKE}}^{\text{IND}_{n,u}\text{-CPA}}$.

Furthermore, we would like to extend our results about the salted FO transform, and KEMs with explicit rejection, to the quantum random oracle model. We gave a sketch of the relevant ideas in Theorem 4.3.1, but did not provide a full proof.

References

- [ABD⁺23a] Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. Annex on FrodoKEM updates, 2023. <https://frodokem.org/files/FrodoKEM-annex-20230418.pdf>.
- [ABD⁺23b] Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM, learning with errors encapsulation preliminary standardization proposal. International Organization for Standardization, 2023. https://frodokem.org/files/FrodoKEM-standard_proposal-20230314.pdf.
- [BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274, Bruges, Belgium, May 14–18, 2000.
- [BCD⁺16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1006–1018, Vienna, Austria, October 24–28, 2016. ACM Press.
- [BDG20] Mihir Bellare, Hannah Davis, and Felix Günther. Separate your domains: NIST PQC KEMs, oracle cloning and read-only indistinguishability. In

Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EURO-CRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 3–32, Zagreb, Croatia, May 10–14, 2020.

- [BDK⁺18] Joppe Bos, Leo Ducas, Eike Kiltz, Lepoint T, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Danien Stehle. Crystals - Kyber: A CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367, 2018.
- [Ber22] Daniel J. Bernstein. Multi-ciphertext security degradation for lattices. Cryptology ePrint Archive, Report 2022/1580, 2022. <https://eprint.iacr.org/2022/1580>.
- [BHH⁺19] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 61–90, Nuremberg, Germany, December 1–5, 2019.
- [Den03] Alexander W. Dent. A designer’s guide to KEMs. In Paterson and Kenneth G., editors, *Cryptography and Coding*, pages 133–151, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [DHK⁺21] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, and Gregor Seiler. Faster lattice-based kems via a generic fujisaki-okamoto transform using prefix hashing. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS ’21*, page 2722–2737, New York, NY, USA, 2021. Association for Computing Machinery.
- [FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO’99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999.
- [GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

- [HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371, Baltimore, MD, USA, November 12–15, 2017.
- [LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339, San Francisco, CA, USA, February 14–18, 2011.
- [LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010.
- [LvdPdW12] Thijs Laarhoven, Joop van de Pol, and Benne de Weger. Solving hard lattice problems and the security of lattice-based cryptosystems. Cryptology ePrint Archive, Paper 2012/533, 2012. <https://eprint.iacr.org/2012/533>.
- [MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 21–39, Santa Barbara, CA, USA, August 18–22, 2013.
- [MR04] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 372–381, 2004.
- [NIS17] NIST. National institute for standards and technology. post-quantum crypto project, 2017.
- [Pei16] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016.
- [PV08] Chris Peikert and Vinod Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 536–553, Santa Barbara, CA, USA, August 17–21, 2008.

- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th annual ACM symposium on Theory of computing*, 2005.
- [RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO’91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444, Santa Barbara, CA, USA, August 11–15, 1992.
- [Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Paper 2004/332, 2004. <https://eprint.iacr.org/2004/332>.
- [SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635, Tokyo, Japan, December 6–10, 2009.