HYBRID KEY ENCAPSULATION MECHANISMS

A Secure Key Encapsulation Mechanism in Quantum Hybrid Settings

By Brian Goncalves, HBSc.

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree Master of Science

McMaster University MASTER OF SCIENCE (2018) Hamilton, Ontario (Mathematics)

TITLE:A Secure Key Encapsulation Mechanism in Quantum Hybrid Settings

AUTHOR: Brian Goncalves, HBSc. (McMaster University)

SUPERVISOR: Dr. Douglas Steblia

NUMBER OF PAGES: **xi, 43**

Lay Abstract

Quantum computers present a threat to current cryptography, as they would be able to break many widely used public-key encryption schemes. In order maintain the security of communication infrastructure it is important that quantum-resistant algorithms become more common in use. However, adoption of quantum-resistant algorithms has been relatively slow, in part due to not wanting to risk abandoning schemes that are secure currently. In this thesis we focus on a specific type of scheme called a *key encapsulation mechanism* (KEM), used to fix a session key for communicating. We construct a secure way to combine currently secure KEMs and quantum-resistant KEMs that are secure now and against quantum computer. Our construction is simple enough that it can be implemented efficiently to provide quantum-resistant security, thus encouraging adoption of quantum-resistant algorithms.

Abstract

Quantum computers pose a long-term threat to many currently used cryptographic schemes as they are able to efficiently solve the computational problems those schemes are based on. This threat has lead to research into quantum-resistant cryptographic schemes to eventually replace those currently used, as well as research into how to ease the transition from classical schemes to quantum-resistant ones. One approach to address these issues is to use a combiner that creates *hybrid* schemes, that is schemes which are classically and quantum-resistant, to protect against quantum attacks and maintain current security guarantees. Such combiners are used as a way to provide trust from different schemes and their differing computational difficulty assumptions rather than a single scheme. which may later become vulnerable. An important type of scheme that must be secure against both classical and quantum attacks are *key encapsulation mechanisms* (KEMs), as they are commonly used for constructing public-key encryption and key exchange protocols. We first define new security notions for KEMs modeling attackers of various levels of quantum power ranging from fully classical to fully quantum. We then construct a combiner that creates hybrid schemes for key encapsulation mechanisms which is secure against adversaries with varying levels of quantum power over time and can be implemented efficiently. Our construction provides an efficient method to combine KEMs using an additional scheme. This construction is also general enough that it can be implemented in settings such as key exchange protocols, like those used in the Transport Layer Security (TLS) protocol for web browsers, without affecting existing structure meaningfully.

Acknowledgements

First I want to thank my advisor Dr. Douglas Steblia for his guidance, advice, and the opportunities he has given me during my graduate career. I am forever grateful for all these things but most importantly, I am grateful for introducing me to the field of post-quantum cryptography which I enjoy more than any other. I consider the decision to study under him the best of my academic career.

I would also like to thank Dr. Marc Fischlin, Nina Bindel, and Jacqueline Brendel for the opportunity to work with them on this thesis and co-author a paper together. Their valuable input and feedback to my research was crucial to this thesis.

I also want to thank my family each of whom supported me throughout my life and academic career, and encouraged me to achieve all I could.

Lastly, I want to thank Jenna. Her unwavering support, encouragement, and faith in me throughout these past years has been a vital part of my life, and achieving this accomplishment which could not have been done without her. I forever grateful to and for her. Thank you.

Contents

List of Abbreviations and Symbols

PKE	Public Key Encryption
KEM	Key Encapsulation Mechanism
KeyGen	Key Generation Algorithm
Enc	Encryption Algorithm
Encap	Encapsulation Algorithm
Dec	Decryption Algorithm
Decap	Decapsulation Algorithm
MAC	Message Authentication Code
Verify	Verification Algorithm
IND	Indistinguishability
OW	One-Waynesss
CPA	Chosen Plaintext Attack
CCA	Chosen Ciphertext Attack
OTS	One Time Strongly Unforgeable
MVA	Multiple Verification Attack
AKE	Authenticated Key Exchange
$\{0,1\}^n$	The set of binary strings of length n
$\{0,1\}^*$	The set of binary strings of arbitrary length
$\|$	Concatenation symbol
$\|B\|$	Cardinality of the set B
$\mathcal{A}(\cdot; r)$	Algorithm $\mathcal{A}$ running on randomness r
$\leftarrow \mathcal{A}$	Output of $\mathcal{A}$
$\leftarrow_{\$}$	Output selected probabilistically
$\mathcal{A}.\mathsf{Subroutine}$	A Subroutine of $\mathcal{A}$
Π	A Public Key Encryption Scheme
$\mathcal{K}$	A Key Encapsulation Mechanism

$\mathcal{M}$	A Message Authentication Code
$M_{\mathcal{A}}$	The message space of algorithm $\mathcal{A}$
$KS_{\mathcal{A}}$	The key space pace of algorithm $\mathcal{A}$
$Perm(S)$	The set of permutations on a set S
$GF(2^n)$	Galois Field of order 2^n

List of Figures

Chapter 1

Introduction

Cryptography is a vital part of the modern world. It is used to provide trust in the digital communication between parties involved by ensuring either the confidentiality or integrity of the communications of those involved. Thus, it is important to ensure the assumptions and uses of cryptographic schemes are sound and secure. Currently, many public-key schemes are reliant upon the computational difficulty of prime factorization [22], or discrete logarithms [12]. However, in the long term these assumptions may not be secure foundations due to Shor's algorithm [23] and quantum computing providing efficient solutions to those problems. It is therefore important to address the long term issue of the threat that quantum computing poses to cryptography.

There has been much work done to develop quantum-resistant cryptographic schemes based on problems thought to be difficult to solve even with access to a quantum computer. However, while such schemes exist, such as Regev's public-key encryption scheme based on lattice problems [21], adoption of these schemes for new protocols and applications have been slow. This slow transition to quantum resistant schemes partially stems from the uncertainty of the computational hardness assumptions of these schemes owing to their relative novelty as bases for cryptographic primitives. As such eventual work may show the computational difficulty of the problems those schemes are based on to be an insecure basis for primitives, in that they may be efficiently solved by quantum computing as well, or even a classical computer. Thus with Shor's algorithm [23] and quantum computing will render many current cryptographic assumptions insecure in the future, and the slow adoption and uncertainty of the definite security of these quantum resistant schemes; there is a need in the intervening time for algorithms which addresses these issues of slow adoption, quantum resistance, and assurance in the computational difficulty assumptions.

Robust combiners offer a tool to solve the above challenge. A robust combiner takes two or more algorithms of the same kind and combine them such that the result is secure to some specification as long some amount of the input algorithms meet that security specification. Thus, using combiners it is possible to construct new secure algorithms from existing ones. Formalized by Harnik *et al.* [15] there has been previous work to use combiners to construct secure algorithms for various primitives. Both Zhang *et al.* [26] and Herzberg [16] proved results on robust combiner for public-key encryption. Bindel *et al.* [7] used combiners to constructed a so called "hybrid" secure, that is secure against both classical and quantum computers, robust combiner for digital signatures, which are used to provide authentication of messages. Giacon *et al.* [13] used combiners for *key encapsulation mechanisms* (KEMs), which are used to establish a shared ephemeral key to

provide confidentiality of messages, the primitive this thesis is focused on. However, the results of Giacon *et al.* [13] are focused on robust combiners in the classical setting for KEMs instead of the so called "hybrid" setting, that is considering both classical and quantum attacks, which this thesis will.

The main result of this thesis is to construct a KEM combiner which is both classically and quantum resistant. Such a KEM combiner addresses the issues of slow transition to post-quantum resistant algorithms and the uncertain computational difficulty of those assumptions; furthermore, it can be applied to work done jointly with Fischlin *et al.* [6] on hybrid KEMs and hybrid authenticated key exchange (AKE) protocols. Additionally, we define a new security model for KEMs that represents the transition from current day classical computers to full quantum computers.

The structure of the thesis is as follows. In Chapter 2, we provide an overview of the necessary fundamental definitions of cryptographic algorithms and functions, a brief description of quantum computing, an overview of standard security proof method, and security definitions for the necessary schemes. In Chapter 3, we outline the new security model and notions for KEMs to model the transition from fully classical to fully quantum computers and formally define robust combiners. In Chapter 4, we prove that those new security notions for KEMs form a nontrivial hierarchy. In Chapter 5, we construct a hybridly secure robust combiner, that is a combiner which is secure against both classical and quantum attacks, for KEMs and prove the security of this combiner within the hierarchy. In Chapter 6 we provide a summary of this thesis, and discuss applications of the result as done in [6].

Chapter 2

Preliminaries

This chapter provides the necessary background for the remainder of this thesis. Section 2.1 introduces the relevant definitions of the needed cryptographic schemes, as well as the matching notion of correctness, two types of relevant functions. In Section 2.2 we provide an introduction to quantum computing. In Section 2.3 we define the relevant security notions of the defined cryptographic schemes and discuss how those security notions are modeled and proven.

2.1 Basic Cryptographic Notation & Definition

In this section we define the basic notation used as well as several fundamental definitions to cryptography. We begin with a description of some notation.

We let $\mathcal{A}(x)$ denote an algorithm $\mathcal{A}(\cdot)$ that runs on input x. Certain algorithms require some randomness and will generate that randomness themselves, however if we wish to specify the randomness be used, say r, we write that as

$$\mathcal{A}(\cdot; r).$$

If $\mathcal{A}$ produces an output after running we represent this as

$$y \leftarrow \mathcal{A}(x).$$

We note that in general $\mathcal{A}$ may produce an output on empty input in. When an algorithm $\mathcal{A}(\cdot)$ has access to some black box oracle which can take input from $\mathcal{A}(\cdot)$ it is written as

$$\mathcal{A}(\cdot)^{\mathcal{B}(\cdot)}.$$

We refer to subroutines within $\mathcal{A}(\cdot)$ as $\mathcal{A}.\mathsf{Subroutine}$.

If an output, y, is to be selected from a set, or space, S or as an output for an algorithm $\mathcal{A}(\cdot)$

probabilistically we will use the notation

$$y \leftarrow\!\!\$\ S$$

for sets, and

$$y \leftarrow\!\!\$\ \mathcal{A}(\cdot)$$

for algorithms. For the purposes of this thesis when we select an element from a set or space we will be doing so uniformly at random.

To denote the exclusive or addition, or XOR, of two bit strings b_1 and b_2 we express it as

$$b_1 \oplus b_2.$$

We denote the splitting of a string s into n sub-strings as

$$s_1 \| \ldots \| s_n \leftarrow s.$$

We begin the definitions with how two parties may interact starting with communicating messages via a *public-key encryption*. For simplicity we implicitly assume all algorithms are polynomial time computable.

Definition 1 (**Public-Key Encryption Scheme** [1])**.** *We say a triple of algorithms* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *form a public-key encryption (PKE) scheme, if:*

- KeyGen*: The key generation algorithm is a probabilistic algorithm which on input* 1^n $(n \in \mathbb{N})$ *outputs a pair,* (pk, sk)*, of public and secret keys.*
- Enc*: The encryption algorithm is a probabilistic algorithm that takes two inputs, a public-key* pk *and a plaintext,* m*, from a designated message space* M_Π*, and outputs a ciphertext* c*.*
- Dec*: The decryption algorithm is a deterministic algorithm that takes as input a secret key* sk *and ciphertext* c *and returns the plaintext* m*.*

Definition 2 (**Correctness of PKEs**)**.** *We say that a public-key encryption scheme,* Π*, is* ϵ*-correct if:*

$$\Pr[\mathsf{Dec}(sk, c) \neq m | (pk, sk) \leftarrow \mathsf{KeyGen}, c \leftarrow \mathsf{Enc}(pk, m)] \leq \epsilon.$$

We say that a PKE, Π*, is correct if for all messages* m *in the message space* M*,* (pk, sk) *output*

by KeyGen*, and ciphertexts* $c = \mathsf{Encaps}(pk, m)$ *that*

$$\Pr[\mathsf{Dec}(sk, c) \neq m | (pk, sk) \leftarrow \mathsf{KeyGen}, c \leftarrow \mathsf{Enc}(pk, m)] = 0.$$

Next, if instead of a message being communicated between the two parties, the parties wish to share an ephemeral key for a future interaction the parties will interact via a *key encapsulation mechanism.*

Definition 3 (**Key Encapsulation Mechanism** [10])**.** *We say triple of algorithms* $\mathcal{K} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *form a key encapsulation mechanism (KEM), if:*

- KeyGen*: The key generation algorithm is a probabilistic algorithm which on input* 1^n $(n \in \mathbb{N})$ *outputs a pair,* (pk, sk)*, of public and secret keys.*
- Encaps*: The encapsulation algorithm is a probabilistic algorithm that takes one input, a public-key* pk *and produces a pair of related outputs, a ciphertext* c *and a ephemeral key* k *from some designated key space,* $KS_{\mathcal{K}}$*.*
- Decaps*: The decapsulation algorithm is a deterministic algorithm that takes as input a secret key* sk *and ciphertext* c *and returns the related ephemeral key* k*.*

Definition 4 (**Correctness of KEMs**)**.** *We say that a KEM* $\mathcal{K}$ *is* ϵ*-correct if:*

$$\Pr[\mathsf{Decaps}(sk, c) \neq k | (pk, sk) \leftarrow \mathsf{KeyGen}, (c, k) \leftarrow \mathsf{Encaps}(pk)] \leq \epsilon.$$

We say a KEM, $\mathcal{K}$*, is correct if for all* (pk, sk) *output by* KeyGen*, and ciphertext key pairs* (c, k) *outputted by* $\mathsf{Encaps}(pk)$ *we have that*

$$\Pr[\mathsf{Decaps}(sk, c) \neq k | (pk, sk) \leftarrow \mathsf{KeyGen}, (c, k) \leftarrow \mathsf{Encaps}(pk)] = 0.$$

Note that KEM can be constructed from a PKE straightforwardly by selecting a message uniformly at random then encrypting that message.

Finally, if one party wishes to validate that the message they received was from the party claiming to send the message then the parties may interact using a *message authentication code.*

Definition 5 (**Message Authentication Code** [18])**.** *We say that a triple of algorithms* $\mathcal{M} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ *form a message authentication code (MAC), if:*

- KeyGen*: The key generation algorithm is a probabilistic algorithm outputs a key* k*.*

- MAC*: A possibly probabilistic algorithm that takes as input a key k and a message $m \in \{0,1\}^*$ and outputs a tag τ.*
- Verify*: A (typically) deterministic algorithm that takes as input a key k, a message m, and a tag τ and returns 1 if $\mathsf{MAC}(k,m) = \tau$ and 0 otherwise.*

Definition 6 (**Correctness of MACs**)**.** *We say that a $\mathcal{M}$ is ϵ-correct if:*

$$\Pr[\mathsf{Verify}(k,m,\tau) = 0 | k \leftarrow \mathsf{KeyGen}, \tau \leftarrow \mathsf{MAC}(k,m)] \leq \epsilon.$$

We say a MAC, $\mathcal{M}$, is correct if

$$\Pr[\mathsf{Verify}(k,m,\tau) = 0 | k \leftarrow \mathsf{KeyGen}, \tau \leftarrow \mathsf{MAC}(k,m)] = 0.$$

We next define an important type of function used in proving security of schemes.

Definition 7 (**Negligible Function** [10])**.** *A function F mapping non-negative integers to non-negative reals is called negligible if for all positive numbers c, there exists an integer $\lambda_0(c) \geq 0$ such that for all $\lambda > \lambda_0(c)$ we have $F(\lambda) < \frac{1}{\lambda^c}$.*

When proving security, negligible functions act as an upper bound on the probability of a security failure.

A collection of functions in which distinct inputs are unlikely to collide are called *Universal*. Formally, this means:

Definition 8 (**Universal Hash Function** [9])**.** *Let H be a finite collection of functions, each of which maps from a set U to B, with B finite. H is called universal if: for all $x, y \in U, x \neq y$ we have that $\Pr[h(x) = h(y) | h \in H] \leq \frac{1}{|B|}$.*

2.2 Quantum Computing

We now provide a brief introduction to quantum computation knowledge used for this thesis. Nielsen and Chuang [19] provide a standard text with a more complete explanation of the subject.

Let $\mathcal{H}$ be a finite-dimensional complex Hilbert space with an inner product. Vectors in $\mathcal{H}$ are denoted with "braket" notation, with $|x\rangle$ being vector in $\mathcal{H}$, and $\langle x|$ denoting the complex conjugate transpose of $|x\rangle$. The inner product on $|x\rangle, |y\rangle$ is then given by $\langle x|y\rangle$. A quantum state is defined as a vector in $\mathcal{H}$ with norm 1. Let $\{|x\rangle\}_x$ be a basis for $\mathcal{H}$, then any quantum state $|y\rangle$ in $\mathcal{H}$ can be represented in superposition as,

$$|y\rangle = \sum \phi_x |x\rangle,$$

where ϕ_x are complex numbers such that $|y\rangle$ has norm 1.

For two quantum systems $\mathcal{H}_1$ and $\mathcal{H}_2$ the joint quantum system is given by the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$; for two states $|x\rangle$ in $\mathcal{H}_1$ and $|y\rangle$ in $\mathcal{H}_2$, then the joint quantum state is represent as $|x\rangle |y\rangle$, or $|x, y\rangle$. Quantum operations on $\mathcal{H}$ are represented by unitary transformations $\mathbf{U}$. Consequently, these quantum operations are, in fact, reversible prior to measurement, as during quantum computation they are unitary matrices. This is notable as it imposes some constraints with the quantization classical operations such as decryption or decapsulation. In particular, let A be a classical algorithm with input x in $\{0,1\}^a$ and output y in $\{0,1\}^b$ and

$$\{0,1\}^a \times \{0,1\}^b \to \{0,1\}^a \times \{0,1\}^b : (x,t) \mapsto (x, t \oplus A(x))$$

be a classical reversible mapping. Then the corresponding unitary transformation $\mathbf{A}$ acting linearly on quantum states is given by

$$\mathbf{A} : \sum_{x,t} \psi_{x,t} |x,t\rangle \mapsto \sum_{x,t} \psi_{x,t} |x, t \oplus A(x)\rangle .$$

For full generality an additional workspace register may be included with the input and output registers. Thus, the general quantization of the classical algorithm is

$$\mathbf{A} : \sum_{x,t,z} \psi_{x,t,z} |x,t,z\rangle \mapsto \sum_{x,t,z} \psi_{x,t,z} |x, t \oplus A(x), z\rangle .$$

2.3 Proofs & Experiments

In this section, we give a description of how security is formally defined and modeled. We also define the needed security notions for PKEs, KEMs, and MACs for this thesis.

Security of cryptographic schemes is modeled with pairs of goals and attacks [1]. These goals define the desired security properties that is trying to be achieved by using such schemes. Attacks refers to the computational power and access that an outside malicious party, or adversary, has available to them. The notation used for the goal and attack model is typically written as *goal-attack*. In the notation it is implicitly assumed that the adversary has classical computational powers, and when the adversary has quantum computational power it is expressed in *goal-attack* notation as *goal*-q*attack*. The distinction between classical and quantum security is an important one, as even a simple quantum adversary is able to break many classically secure algorithms. For example, schemes relying on the computational difficulty of prime factorization, such as those based on RSA encryption [22], or discrete logarithms, such as those based on the Diffie-Hellman key exchange [12], are secure against classical adversaries under various *goal-attack* pairs. However, both prime factorization and discrete logarithms are solvable in polynomial time by simple quantum adversary that runs Shor's algorithm [23]. Thus, it a distinction that must be made when discussing a schemes' security goals and potential attacks.

Security proofs are done in terms of *experiments* [1] and *games* [24] written in pseudocode and *advantages*. Experiments are idealized scenarios that are intended to formally describe security of a cryptographic scheme with respect to a specific goal and a specific attack. Written in pseudocode the experiment outlines the initialization of a scheme, for example a PKE generating the public and secret key pair, as well as an adversary and the powers granted to them, as well as a security break condition that if the adversary meets, implies that the scheme cannot achieve its intended goal with respect to the attack. Such break conditions are typically weak/limited versions of the goal the scheme is intended to provide, with the idea being that if an adversary given certain computational powers cannot even break such a weak version of the security goal then the scheme is secure to use for the full goal.

To prove security, modifications are made to the pseudocode of the original experiment in order to create a scenario in which the adversary cannot achieve the security break condition and thus "win" the security experiment. Each modification denotes a new *game* [24] and must be undetectable to the adversary. This last requirement means that after each modification, or *game hop* [24] between games, that a negligible function must upper bound the the difference in probability of the adversary winning in the two games. Ultimately we wish to force the adversary into a scenario in which the most likely way to win is to simply guess an answer or attempt for the security break condition.

Finally security of a given scheme is expressed in terms of advantages. Informally, advantages represent the probability the adversary wins the security experiment for a given scheme and *goal-attack* pair. Formally the advantage of a scheme, Π, with respect to the goal-attack pair *goal-attack*, and the adversary $\mathcal{A}$ is defined to be

$$\mathsf{Adv}_{\Pi}^{\textit{goal-attack}}(\mathcal{A}) = \Pr\left[\mathrm{Expt}_{\Pi}^{\textit{goal-attack}}(\mathcal{A})\right],$$

where $\mathrm{Expt}_{\Pi}^{\textit{goal-attack}}(\mathcal{A})$ denotes the experiment of the scheme with respect to the specified goal and attack.

We will now define the security experiments for PKEs, KEMs, and MACs. For PKEs and KEMs the primary security goal is that of indistinguishability, denoted as IND [14] [1]. The desired property that indistinguishability security provides for PKEs is that an adversary cannot distinguish what message a ciphertext is encrypting. Whereas with KEMs IND [10] security is that the adversary cannot distinguish the key produced from the encapsulation algorithm and a binary string chosen uniformly at random from the same space as the key.

In terms of attacks the two most common attacks that PKEs and KEMs are designed to be resistant against are chosen plaintext attack and adaptive chosen ciphertext attack. Chosen plaintext attack, or CPA, is an attack in which the adversary only has access to the public key of the PKE [1], or KEM [10], and is able to encrypt any message they choose as many times as they choose, or run the encapsulation algorithm as many times as they choose in the case of KEMs). The adaptive chosen ciphertext attack, or CCA [20] is similar to CPA: the adversary can encrypt any message they choose as many times as they choose for PKEs [1], or encapsulate as many times as they choose in the case of KEMs [10]. The difference is that the adversary is also given access to a black box oracle that is programmed with the secret and will decrypt, or decapsulate in the case of KEMs, ciphertexts the adversary queries to it.

We provide a formal definition of the IND-CPA and IND-CCA security experiments for KEMs as the main result of this thesis is on KEMs.

Definition 9 (IND-CPA **and** IND-CCA **Security for KEMs** [10])**.** *Both the* IND-CPA *and* IND-CCA *experiments for KEMs* $\mathcal{K} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *against the adversary* $\mathcal{A}$ *consists of three phases:*

1. ***Setup****: The experiment, or challenger, runs* KeyGen *to generate a public-secret key pair. A ciphertext-key pair is encapsulated on the public key as well as another key selected uniformly at random from the key space for* $\mathcal{K}$*. Lastly, a bit* b *is chosen uniformly at random from* $\{0, 1\}$*.*

2. ***Challenge****: The adversary receives the public key, and ciphertext generated in* ***setup****. They also receive a key depending on the bit chosen. If the bit* b *was* 0 *then the adversary receives the key encapsulated by the ciphertext, otherwise they receive the key selected at random. In the* IND-CCA *experiment the adversary is also given access to a decapsulation oracle, whereas in the* IND-CPA *the adversary does not receive access to such an oracle. The stage ends when the adversary outputs a guess bit* b'*.*

3. ***Check****: Finally,* b' *is checked against* b*. If equal then the adversary has won the security experiment.*

$\underline{\mathrm{Expt}^{\mathsf{IND\text{-}CPA}}_{\mathcal{K}}(\mathcal{A})}$:
1. $(pk, sk) \leftarrow\!\!\$\ \mathcal{K}.\mathsf{KeyGen}(1^n)$
2. $(c^*, k_0) \leftarrow \mathcal{K}.\mathsf{Encaps}(pk)$
3. $k_1 \leftarrow\!\!\$\ KS_{\mathcal{K}}$
4. $b \leftarrow\!\!\$\ \{0, 1\}$
5. $b' \leftarrow \mathcal{A}(pk, c^*, k_b)$
6. return $(b = b')$

$\underline{\mathrm{Expt}^{\mathsf{IND\text{-}CCA}}_{\mathcal{K}}(\mathcal{A})}$:
1. $q_C \leftarrow 0$
2. $(sk, pk) \leftarrow\!\!\$\ \mathcal{K}.\mathsf{KeyGen}(1^n)$
3. $(c^*, k_0) \leftarrow \mathcal{K}.\mathsf{Encaps}(pk)$
4. $k_1 \leftarrow\!\!\$\ KS_{\mathcal{K}}$
5. $b \leftarrow\!\!\$\ \{0, 1\}$
6. $b' \leftarrow \mathcal{A}^{\mathcal{O}_D(sk, c^*, \cdot)}(pk, c^*, k_b)$
7. return $[b = b']$

$\underline{\mathcal{O}_D(sk, c^*, c)}$:
1. $q_C \leftarrow q_C + 1$
2. if $c = c^*$: return $\perp$
3. else: return $\mathcal{K}.\mathsf{Dec}(c, sk)$

Figure 2.3.1: IND-CPA and IND-CCA security experiments for KEMs.

It is clear that a simple adversary can win either IND experiment with probability $\frac{1}{2}$ by simply selecting the bit b' uniformly at random themselves and submitting it as their guess. It is for this reason that the advantage expression is modified to

$$\mathsf{Adv}^{goal\text{-}attack}_{\Pi}(\mathcal{A}) = \left| \Pr\left[\mathrm{Expt}^{goal\text{-}attack}_{\Pi}(\mathcal{A})\right] - \frac{1}{2} \right|,$$

to check security against more meaningful adversaries and attacks by measuring the advantage as the distance from $\frac{1}{2}$.

We also note that in both the IND-CPA and IND-CCA security experiments the adversary is attempting to distinguish which of the two messages it provided to the challenger. This is done to give the adversary a weaker, more limited version of indistinguishability, where the adversary may or may not know the message being encrypted. Furthermore, in the CCA version of the experiment the adversary is barred from querying the challenge ciphertext to the decapsulation oracle as it would allow for trivial victory for the adversary.

Another goal that both PKEs and KEMs share is that of one-wayness, denoted as OW [17]. The desired property that one-wayness security provides for both PKEs and KEMs is that an adversary cannot recover the message, or key in the case of KEMs, from the ciphertext. Similar to IND, the primary attacks against OW security are CPA and CCA.

We provide a formal definition of the the OW-CPA security experiment for PKEs as a Proposition 3 utilizes this notion of security of PKEs:

Definition 10 (OW-CPA **Security for PKEs** [1] [17])**.** *The* OW-CPA *experiment for PKE* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *against the adversary* $\mathcal{A}$ *consists of three phases:*

1. ***Setup****: The experiment, or challenger, runs* KeyGen *to generate a public-secret key pair. A message,* m^* *is selected uniformly at random from the message space,* M_Π*, and then is encrypted.*

2. ***Challenge****: The adversary receives the public key, and the ciphertext,* c^* *as in* ***Setup****. The stage ends when the adversary outputs a guess message* m*.*

3. ***Check****: Finally,* m *is checked against* m^**. If equal then the adversary has won the security experiment.*

$\underline{\text{Expt}^{\mathsf{OW\text{-}CPA}}_{\Pi}(\mathcal{A})}$:

1. $(pk, sk) \leftarrow\!\!\$\ \Pi.\mathsf{KeyGen}(1^n)$
2. $m^* \leftarrow\!\!\$\ M_\Pi$
3. $c^* \leftarrow \Pi.\mathsf{Enc}(pk, m^*)$
4. $m \leftarrow \mathcal{A}(pk, st, c^*)$
5. return $[m = m^*]$

Figure 2.3.2: OW-CPA security experiment for PKEs.

For MACs, the primary security goal that we will consider is that of *one time strongly unforgeable*, denoted OTS. This means that an adversary cannot create another valid message-tag pair after being given a valid pair. The primary attack considered is the multiple verification attack, denoted as MVA, where the adversary is able to make multiple queries to black box verification oracle on any message-tag pairs they wish, and to verify if those pairs are valid.

Formally, we describe OTS-MVA security as follows:

Definition 11 (OTS-MVA **Security for MACs**)**.** *The* OTS-MVA *security experiment for MAC* $\mathcal{M} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ *against the adversary* $\mathcal{A}$ *consists of three phases:*

1. ***Setup****: The experiment, or challenger, runs* KeyGen *to generate a uniform MAC key* k*. The adversary,* $\mathcal{A}$*, performs any pre-computation they wish and outputs a message,* m *and a state vector* st*. Finally the message is then signed with the MAC key to produce a tag,* τ^**.*

2. ***Challenge****: The adversary receives as input* τ^* *and* st*. Furthermore,* $\mathcal{A}$ *is given access to a verification oracle that takes as input a message-tag pair, and returns* 1 *if it is a valid message-tag pair and* 0 *otherwise. This stage ends when* $\mathcal{A}$ *outputs a message-tag pair* (m', τ')*.*

3. ***Check****: Finally,* (m', τ') *is checked if it is a valid message-pair and not equal to* (m, τ^*)*. The adversary wins the security experiment if* (m', τ') *satisfies the above.*

$\underline{\mathrm{Expt}^{\mathsf{OTS\text{-}MVA}}_{\mathcal{M}}(\mathcal{A})}$:

1. $k \leftarrow\!\!\$\ \mathcal{M}.\mathsf{KeyGen}()$
2. $(m^*, st) \leftarrow \mathcal{A}()$
3. $\tau^* \leftarrow \mathcal{M}.\mathsf{MAC}_k(m^*)$
4. $(m', \tau) \leftarrow \mathcal{A}^{\mathcal{O}_V(k,\cdot,\cdot)}(\tau^*, st)$
5. if $[\mathsf{Verify}_k(m', \tau') = 1] \wedge [(m', \tau') \neq (m^*, \tau^*)]$:
6. return 1
7. else 0

$\underline{\mathcal{O}_V(k, \cdot, \cdot)}$:

1. return $\mathcal{M}.\mathsf{Verify}_k(m, \tau)$

Figure 2.3.3: Security experiment for one-time strong existential unforgeability (with multiple verifications) of a MAC $\mathcal{M} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$.

Note, that the adversary is able to choose any message they wish to receive a MAC tag on and can wins by forging any new message-tag pair, even on the initial message.

Chapter 3

Two Stage Adversary & Combiners

In this chapter we introduce the notions of a two stage adversary, and combiners. With the notion of the two stage adversary we also define the two stage adversary version of IND-CPA and IND-CCA security experiments for KEMs, as well as the OTS-MVA security experiment for MACs. Finally, we introduce the definition of a (k, n)-robust combiner which form the basis for the main result of this thesis.

3.1 Two Stage Adversary

Originally introduced by Bindel *et al.* [7], the notion of a two stage adversary was a method to model security during the transition from known classically secure to *post quantum* secure digital signatures. We adapt this notion of a two stage adversary from digital signature to KEMs as a way to model the same transition from known classically secure to *post quantum* security for KEMs. In this model, the challenge phase of the security experiment is split into two separate phases and correspondingly we model the adversary as being split into two algorithms, $\mathcal{A}_1$ and $\mathcal{A}_2$. While the adversary is interacting with the experiment $\mathcal{A}_1$ is ran, having access to any relevant oracles and choice of inputs the experiment requires, before terminating and passing some state information to $\mathcal{A}_2$, which does not have access to the attack oracle and then returns the final value for the experiment to check against. For each stage the model specifies the computational power of both adversary algorithms, as well as the type of oracle access of each. Let $\mathsf{X}, \mathsf{Z} \in \{\mathsf{C}, \mathsf{Q}\}$ and $\mathsf{y} \in \{\mathsf{c}, \mathsf{q}\}$, then the two-stage model the adversary's computational abilities can be described using the notation $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$. In $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ notation X denotes whether $\mathcal{A}_1$ is classical (C) or quantum (Q), y denotes whether $\mathcal{A}_1$ has classical (C) or quantum (Q) access to the decapsulation oracle, and lastly Z whether $\mathcal{A}_2$ is classical (C) or quantum (Q). It is important to note that not all combinations of $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ are meaningful, such as C^qZ, thus this thesis is focused on four types of $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ adversaries that will reflect the real world transition adopting the names previously suggested by [7].

1. $\mathsf{C}^{\mathsf{c}}\mathsf{C}$: Fully classical adversary. The adversary has classical computational powers at all times and can classically interact with a decapsulation oracle. This is equivalent to the traditional adversary in the IND-CCA security experiment.

2. $\mathsf{C^cQ}$: Future quantum adversary. The adversary has classical computational powers during their time interacting with decapsulation oracle. They are able to perform classical queries to the decapsulation oracle. At a later time the adversary gains quantum computational powers but loses access to the decapsulation oracle. This models the scenario of a KEM that is being used to establish a shared ephemeral key between senders and receivers today, but eventually is no longer used to establish shared ephemeral keys; however, those keys need to remain secure even after quantum computer become available.

3. $\mathsf{Q^cQ}$: Quantum adversary with classical queries. The adversary has quantum computational powers at all times. However non-malicious, or honest, users of the KEM only ever decapsulate ciphertexts on classical computers, thus the decapsulation oracle is always classical.

4. $\mathsf{Q^qQ}$: Fully quantum adversary. The adversary has quantum computational powers at all times and can quantumly interact with a decapsulation oracle.

When working with two stage $\mathsf{X^yZ}$ adversaries in the goal-attack model the notation $\mathsf{X^yZ}$-*goal* will be used. This notation is chosen as for KEMs the primary attacks are CPA or CCA, and by definition the CPA attack the adversary does not have any decapsulation oracle access. Thus for CPA security, a $\mathsf{C^cC}$ two stage adversary becomes equivalent to a single classical, or C , adversary, and both $\mathsf{Q^cQ}$, $\mathsf{Q^qQ}$ types of two stage adversaries each become equivalent to a single quantum, or Q, adversary. Lastly, a $\mathsf{C^cQ}$ two stage adversary is also equivalent to a single Q adversary as anything the first stage classical adversary algorithm can compute so too can the second stage quantum adversary algorithm, but being quantum provides more computational power as such there would be no reason to remain in the first stage.

3.2 $\mathsf{X^yZ}$ Security Experiments

We now define the two stage adversary security experiments for KEMs and MACs.

First, we define the $\mathsf{X^yZ\text{-}IND\text{-}CCA}$, which we shorten to $\mathsf{X^yZ\text{-}IND}$ because of the discussion at the end of section 3.1, security experiment for KEMs.

Definition 12 ($\mathsf{X^yZ\text{-}IND}$ **Security Experiment**). *The* $\mathsf{X^yZ\text{-}IND}$ *experiment for KEM* $\mathcal{K} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *against the adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, consists of four phases:*

1. ***Setup****: The experiment, or challenger, runs* KeyGen *to generate a public-secret key pair. A ciphertext-key pair is encapsulated on the public key as well as another key selected uniformly at random from the key space for* $\mathcal{K}$*. Lastly, a bit* b *is chosen uniformly from* $\{0, 1\}$*.*

2. ***Query****: The adversary,* $\mathcal{A}_1$*, receives the public key, and ciphertext generated in* ***setup****. They also receive a key depending on the bit chosen. If the bit was* 0 *then the adversary receives the key encapsulated by the ciphertext, otherwise they receive the key selected at*

random. During this stage the adversary has computational powers denoted by X*. The adversary also has* y *type access to the decapsulation oracle. This stage once the adversary outputs a state vector st containing any information they choose.*

3. ***Challenge****: The adversary,* $\mathcal{A}_2$*, receives as input the state vector st produced by* $\mathcal{A}_1$*.* $\mathcal{A}_2$ *has computational power denoted by* Z *as well as having matching random oracle access. This stage ends when the adversary,* $\mathcal{A}_2$*, outputs a guess bit* b'*.*

4. ***Check****: Finally,* b' *is checked against* b*. If equal then the adversary has won the security experiment.*

$\underline{\mathrm{Expt}^{\mathsf{X}^{\mathsf{y}}\mathsf{Z}\text{-}\mathsf{IND}}_{\mathcal{K}}(\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)):}$

1. $q_D \leftarrow 0$
2. $(pk, sk) \leftarrow\!\!\$\ \mathcal{K}.\mathsf{KeyGen}(1^n)$
3. $(c^*, \kappa_0^*) \leftarrow \mathcal{K}.\mathsf{Encaps}(pk)$
4. $\kappa_1^* \leftarrow\!\!\$\ KS_{\mathcal{K}}$
5. $b \leftarrow\!\!\$\ \{0,1\}$
6. $st \leftarrow \mathcal{A}_1^{\mathcal{O}_D^{\mathsf{y}}(sk,c^*,\cdot)}(pk, c^*, \kappa_b^*)$
7. $b' \leftarrow \mathcal{A}_2(st)$
8. return $[b = b']$

$\underline{\mathcal{O}_D^{\mathsf{c}}(sk, c^*, c)}$

1. $q_D \leftarrow q_D + 1$
2. if $c = c^*$: return $\perp$
3. else: return $\mathcal{K}.\mathsf{Decaps}(sk, c)$

$\underline{\mathcal{O}_D^{\mathsf{q}}(sk, c^*, \sum_{c,t,z} \psi_{c,t,z} \left|c,t,z\right\rangle):}$

1. $q_D \leftarrow q_D + 1$
2. Return state $\sum_{c,t} \psi_{c,t} \left|c, t \oplus \mathcal{O}_D^{\mathsf{c}}(sk, c, c^*), z\right\rangle$

Figure 3.2.1: Security experiment for indistinguishable under chosen message attack of a KEM $\mathcal{K} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ against an $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Next we define the $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-OTS-MVA, which we will shorten to $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-OTS, security experiment for MACs.

Definition 13 ($\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-OTS **Security Experiment**)**.** *The* $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-OTS *security experiment for MAC* $\mathcal{M} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ *against the adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *consists of four phases:*

1. ***Setup****: The experiment, or challenger, runs* KeyGen *to generate a uniform MAC key* k*. The adversary,* $\mathcal{A}$*, performs any pre-computation they wish and outputs a message* m *and a state vector st. Finally message is then signed with the MAC key to produce a tag,* τ^**.*

2. ***Query****: The adversary,* $\mathcal{A}_1$*, receives as input* τ^* *and st. During this stage the adversary has computational powers denoted by* X*. Furthermore,* $\mathcal{A}_1$ *is given* y *type access to a verification*

oracle that takes as input a message-tag pair, and returns 1 *if it is a valid message-tag pair and* 0 *otherwise. This stage ends when* $\mathcal{A}_1$ *outputs a state vector* st'*.*

3. ***Challenge****: The adversary,* $\mathcal{A}_2$*, receives as input* st'*. During this stage the adversary has computational powers denoted by* Z *and no access to the verification oracle. This stage ends when* $\mathcal{A}_2$ *outputs a message-tag pair* (m', τ')*.*

4. ***Check****: Finally,* (m', τ') *is checked if it is a valid message-pair and not equal to* (m, τ^*)*. The adversary wins the security experiment if* (m', τ') *satisfies the above.*

$\underline{\text{Expt}^{\mathsf{X}^y\mathsf{Z}\text{-OTS}}_{\mathcal{M}}(\mathcal{A})}$:

1. $k \leftarrow\!\!\$\ \mathcal{M}.\mathsf{KeyGen}()$
2. $(m^*, st) \leftarrow \mathcal{A}_1()$
3. $\tau^* \leftarrow \mathcal{M}.\mathsf{MAC}_k(m^*)$
4. $st' \leftarrow \mathcal{A}_1^{\mathcal{O}_V^y(k,\cdot,\cdot)}(\tau^*, st)$
5. $(m', \tau') \leftarrow \mathcal{A}_2(st')$
6. if $[\mathsf{Verify}_k(m', \tau') = 1] \wedge [(m', \tau') \neq (m^*, \tau^*)]$:
7. return 1
8. else: return 0

$\underline{\mathcal{O}_V^c(k, m, \tau)}$:

1. return $\mathcal{M}.\mathsf{Verify}_k(m, \tau)$

$\underline{\mathcal{O}_V^q(k, \sum_{m,\tau,t,z} \psi_{m,\tau,t,z} |m, \tau, t, z\rangle)}$:

1. return $\sum_{m,\tau,t,z} \psi_{m,\tau,t,z} |m, \tau, t \oplus \mathcal{M}.\mathsf{Verify}_k(m, \tau), z\rangle$

Figure 3.2.2: Security experiment for one-time strong existential unforgeability (with multiple verifications) of a MAC $\mathcal{M} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ against an $\mathsf{X}^y\mathsf{Z}$ adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

3.3 Robust Combiners

We now define the main constructive tool used in this thesis, a (k, n)-robust combiner introduced by Harnik *et al.* [15] as well as discuss some previous work done with them. Informally, a (k, n)-robust combiner is an algorithm that accepts as input n of the same type of cryptographic schemes, such as KEMs, to produce a new cryptographic scheme of the same type, and so long as at least k of the input satisfies the same security notion the output also equally secure.

Definition 14. ***(k, n)-Robust Combiner*** *[15] Let* $\mathbb{P}$ *be a set of cryptographic primitives. A* (k, n)*-robust combiner is an algorithm that gets* n *candidate schemes from* $\mathbb{P}$ *as inputs, and whose output is a single algorithm that is secure to some security specification* s *if the following hold:*

1. *If at least* k *candidates securely implement the security specification* s *then the result of the combiner also securely implements* s*.*

2. *The running time of the result of the combiner is polynomial in the security parameter m, in n and in the lengths of the inputs to $\mathbb{P}$.*

We now discuss some of the previous work done with robust combiners, specifically for KEMs and PKEs, as they can easily be turned into KEMs.

First, we discuss Zhang *et al.*'s [26] results on combiners for PKEs. Their initial attempt to construct an IND-CCA $(1,2)$-robust combiner is described in Figure 3.3.1. Unfortunately, such a combiner does not preserve IND-CCA security regardless of which scheme is IND-CCA secure, but rather achieves a weaker security notion called "detectable IND-CCA". Zhang *et al.* do construct an IND-CCA combiner for PKEs but their construction requires 3 input PKEs, each of which is required to have a minimum security assumption on them to ensure IND-CCA security of the resulting PKE, Figure 3.3.2. However, this construction has an important drawback: if any of the PKEs are insecure the resulting PKE will not be IND-CCA secure, and thus not X^{y}Z-IND secure. Thus, this combiner is not well suited for use with schemes based on computational difficulty assumptions that are not fully trusted.

Π.KeyGen(1^n):	Π.Enc(pk, m):	Π.Dec$(sk, (c_1, c_2))$:
1. $(pk_1, sk_1) \leftarrow\$ \mathsf{KeyGen}_1(1^n)$	1. $r \leftarrow\$ M_{\Pi_2}$	1. return $m \leftarrow \mathsf{Dec}_1(sk_1 c_1) \oplus \mathsf{Dec}_2(sk_2, c_2)$
2. $(pk_2, sk_2) \leftarrow\$ \mathsf{KeyGen}_2(1^n)$	2. $c_1 \leftarrow \mathsf{Enc}_1(pk_1, r)$	
3. $pk \leftarrow (pk_1, pk_2)$	3. $c_2 \leftarrow \mathsf{Enc}_2(pk_2, r \oplus m)$	
4. $sk \leftarrow (sk_1, sk_2)$	4. $c \leftarrow (c_1, c_2)$	
5. return (pk, sk)	5. return c	

Figure 3.3.1: Zhang *et al.*'s first combiner that preserves "detectable IND-CCA" security on input PKEs Π_1 and Π_2.

Π.KeyGen(1^n):	Π.Enc(pk, m):
1. $(pk_d, sk_d) \leftarrow\$ \mathsf{KeyGen}_d(1^n)$	1. $r_d, r_q, r_c \leftarrow\$ 0, 1^n$
2. $(pk_q, sk_q) \leftarrow\$ \mathsf{KeyGen}_q(1^n)$	2. $c_d \leftarrow \mathsf{Encaps}_d(pk_d, r_c \| r_q \| m; r_d)$
3. $(pk_c, sk_c) \leftarrow\$ \mathsf{KeyGen}_c(1^n)$	3. $c_q \leftarrow \mathsf{Encaps}_q(pk_q, c_d; r_q)$
4. $pk \leftarrow (pk_d, pk_q, pk_c)$	4. $c_c \leftarrow \mathsf{Encaps}_c(pk_c, c_d; r_c)$
5. $sk \leftarrow (sk_d, sk_q, sk_c)$	5. $c \leftarrow (c_q, c_c)$
6. return (pk, sk)	6. return c

Π.Dec$((sk_d, sk_q, sk_c), (c_q, c_c))$:

1. $c'_d \leftarrow \mathsf{Dec}_q(sk_q, c_q)$
2. $r'_c \| r'_q \| m' \leftarrow \mathsf{Dec}_d(sk_d, c'_d)$
3. If: $c_q = \mathsf{Enc}_q(pk_q, c'_d; r_q) \wedge c_c = \mathsf{Enc}_c(pk_c, c'_d; r_c)$ return m'
4. else: $\perp$

Figure 3.3.2: Zhang *et al.*'s second combiner that preserves IND-CCA security on inputs Π_d, the detectable IND-CCA PKE, Π_q, the 1-bounded PKE, and Π_c, a IND-CPA PKE.

Next we discuss the results of Herzberg on the nested, or *cascade*, combiner for PKEs [16] as defined in Figure 3.3.3. In particular Herzberg proved that cascade encryption is not a robust combiner for IND-CCA security of PKEs, but rather the cascade combiner only retains weaker

security notions such as IND-CPA security for so called length-uniform PKEs. The cascade combiner not preserving IND-CCA means that this combiner is not secure to use against any $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-IND adversary as all $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ adversaries are at least $\mathsf{C}^{\mathsf{c}}\mathsf{C}$ adversaries and thus equivalent to an IND-CCA adversary. This means that this combiner is also not well suited for the purposes of this thesis as we model against adaptive adversaries in the two stage $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ model.

$\Pi.\mathsf{KeyGen}(1^n)$:	$\Pi.\mathsf{Enc}(pk_1, pk_2, m)$:	$\Pi.\mathsf{Dec}((sk_1, sk_2), c)$:
1. $(pk_1, sk_1) \leftarrow\!\!\$\ \mathsf{KeyGen}_1(1^n; r_1)$	1. $c_1 \leftarrow \mathsf{Enc}_1(pk_1, m; r_1)$	1. $c_1' \leftarrow \mathsf{Dec}_2(sk_2, c)$
2. $(pk_2, sk_2) \leftarrow\!\!\$\ \mathsf{KeyGen}_2(1^n; r_2)$	2. $c \leftarrow \mathsf{Enc}_2(pk_2, c_1; r_2)$	2. $m' \leftarrow \mathsf{Dec}_1(sk_1, c_1')$
3. $pk \leftarrow (pk_1, pk_2)$	3. return c	3. return m'
4. $sk \leftarrow (sk_1, sk_2)$		
5. return (pk, sk)		

Figure 3.3.3: Herzberg's cascade combiner on PKEs Π_1 and Π_2.

Lastly, we discuss Giacon *et al.* work on robust KEM combiners [13]. Giacon *et al.* first prove that two specific combiners do not preserve IND-CCA, specifically that the simple combiner that XORs all the keys, Figure 3.3.4, and the combiner which XORs all the keys then applies a pseudorandom function on the XOR and all ciphertexts, Figure 3.2.5, are only IND-CPA secure. To achieve full IND-CCA security Giacon *et al.* define a special variant of a pseudorandom function called a split key pseudorandom function. However, to construct and prove security with the so called split key pseudorandom functions Giacon *et al.* model them as a truly random function, or random oracle. The requirement that the pseudorandom function used is a split key pseudorandom function places a restriction that limits the potential choices that can be used, whereas the main result of this thesis uses a more general and more easily realized universal hash function. The use of universal hash functions means that the combiner presented in this thesis uses weaker assumptions than previous work. Furthermore, Giacon *et al.* exclusively look at classical IND-CCA which does not ensure security against a quantum adversary, meaning that quantum version of their results would require new proofs. Thus the results of Giacon *et al.* may not hold against quantum adversaries and as such their combiner may be insecure against the majority of $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-IND adversaries considered in this thesis.

$\mathcal{K}.\mathsf{KeyGen}(1^n)$:	$\mathcal{K}.\mathsf{Encaps}(pk_1, pk_2)$:	$\mathcal{K}.\mathsf{Decaps}((sk_1, sk_2), ((c_1, c_2))$:
1. $(pk_1, sk_1) \leftarrow\!\!\$\ \mathsf{KeyGen}_1(1^n)$	1. $(c_1, k_1) \leftarrow \mathsf{Encaps}_1(pk_1)$	1. $k_1' \leftarrow \mathsf{Decaps}_1(sk_1, c_1)$
2. $(pk_2, sk_2) \leftarrow\!\!\$\ \mathsf{KeyGen}_2(1^n)$	2. $(c_2, k_2) \leftarrow \mathsf{Encaps}_2(pk_2)$	2. $k_2' \leftarrow \mathsf{Decaps}_2(sk_2, c_2)$
3. $pk \leftarrow (pk_1, pk_2)$	3. $c \leftarrow (c_1, c_2)$	3. $k' \leftarrow k_1' \oplus k_2'$
4. $sk \leftarrow (sk_1, sk_2)$	4. $k \leftarrow k_1 \oplus k_2$	4. return k'
5. return (pk, sk)	5. return (c, k)	

Figure 3.3.4: The XOR combiner for KEMs on input $\mathcal{K}_1$ and $\mathcal{K}_2$.

In the main result of this thesis we construct a combiner which does not have the drawbacks that previous works have. The combiner constructed in Chapter 5 of this thesis preserves the strong security notions of $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-IND and only requires that at least one input scheme is secure, avoiding the requirements of multiple security assumptions. Moreover, it utilizes less cryptographic assumptions by using a universal hash function. Furthermore, by working in the two stage model it accounts for quantum adversaries and proves security against such adversaries.

$\mathcal{K}.\mathsf{KeyGen}(1^n)$:	$\mathcal{K}.\mathsf{Encaps}(pk_1, pk_2)$:	$\mathsf{XtM}.\mathsf{Decaps}((sk_1, sk_2), (c_1, c_2))$:
1. $(pk_1, sk_1) \leftarrow\!\!\$\ \mathsf{KeyGen}_1(1^n)$	1. $(c_1, k_1) \leftarrow \mathsf{Encaps}_1(pk_1)$	1. $k_1' \leftarrow \mathsf{Decaps}_1(sk_1, c_1)$
2. $(pk_2, sk_2) \leftarrow\!\!\$\ \mathsf{KeyGen}_2(1^n)$	2. $(c_2, k_2) \leftarrow \mathsf{Encaps}_2(pk_2)$	2. $k_2' \leftarrow \mathsf{Decaps}_2(sk_2, c_2)$
3. $pk \leftarrow (pk_1, pk_2)$	3. $k \leftarrow F(k_1, k_2, c_1 \Vert c_2)$	3. return $k \leftarrow F(k_1, k_2, c_1 \Vert c_2)$
4. $sk \leftarrow (sk_1, sk_2)$	4. $c \leftarrow (c_1, c_2)$	
5. return (pk, sk)	5. return (c, k)	

Figure 3.3.5: The split key pseudorandom function combiner on input KEMs $\mathcal{K}_1$ and $\mathcal{K}_2$ and split key pseudorandom function F.

Chapter 4

Separations and Implications

Bindel *et al.* [7] showed that there exists a hierarchy of security against two stage adversaries for digital signatures. Likewise,the four $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-IND KEM security notions defined in the previous chapter form a hierarchy between them. Furthermore, it shows that there exists separations between these notions and construct examples of KEMs that separate the attacks. In this chapter we prove these separations between definitions as well as the implications between them.

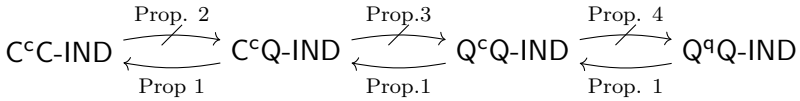

Figure 4.0.1: Implications ($\rightarrow$) and separations ($\not\rightarrow$) between indistinguishability-based security notions for KEMs wrt. two-stage adversaries.

4.1 Implications

We begin by proving the implications between the notions. That is if a KEM is Q^{q}Q-IND secure, it is also Q^{c}Q-IND secure. Likewise if a KEM is Q^{c}Q-IND secure, it is also C^{c}Q-IND secure, and ik a KEM is C^{c}Q-IND it is also C^{c}C-IND secure.

Proposition 1. ***(Implications)*** *Let $\mathcal{K}$ be a KEM. If $\mathcal{K}$ is* Q^{q}Q-IND *secure, then $\mathcal{K}$ is also* Q^{c}Q-IND *secure. If $\mathcal{K}$ is* Q^{c}Q-IND *secure, then $\mathcal{K}$ is also* C^{c}Q-IND *secure. If $\mathcal{K}$ is* C^{c}Q-IND *secure, then $\mathcal{K}$ is also* C^{c}C-IND *secure. Formally, we have*

$$\mathsf{Adv}_{\mathcal{K}}^{\mathsf{Q^qQ\text{-}IND}}(\mathcal{A}_1,\mathcal{A}_2) \geq \mathsf{Adv}_{\mathcal{K}}^{\mathsf{Q^cQ\text{-}IND}}(\mathcal{A}_1,\mathcal{A}_2) \geq \mathsf{Adv}_{\mathcal{K}}^{\mathsf{C^cQ\text{-}IND}}(\mathcal{A}_1,\mathcal{A}_2) \geq \mathsf{Adv}_{\mathcal{K}}^{\mathsf{C^cC\text{-}IND}}(\mathcal{A}_1,\mathcal{A}_2).$$

Proof. This is straightforward as every classical algorithm can also be implemented on a quantum

computer that does not utilizes its additional quantum power. Furthermore, classical queries can also be simulated by superposition queries where all values are identical. □

4.2 Separations

We now prove that this hierarchy of security is strict by constructing the separating KEMs in the following propositions. We first begin with the fully classical- future quantum separation which follows from Shor's algorithm.

Proposition 2. *(*$\mathsf{C^cC\text{-}IND} \not\Longrightarrow \mathsf{C^cQ\text{-}IND}$*) In the classical model, assuming RSA is a one-way function for classical algorithms, then there exists a* $\mathsf{C^cC\text{-}IND}$ *KEM that is not* $\mathsf{C^cQ\text{-}IND}$*.*

Proof. Bellare and Rogaway construct a KEM based on RSA-OAEP which is IND-CCA secure and thus equivalent to $\mathsf{C^cC\text{-}IND}$ secure [4]. However, a $\mathsf{C^cQ}$ adversary has local quantum computing power in their second stage and can run Shor's algorithm on the RSA modulus to factor it and recover the decapsulation key and win the $\mathsf{C^cQ\text{-}IND}$ experiment and win with probability 1.

□

Next, we separate future quantum and quantum with classical queries.

Proposition 3. *(*$\mathsf{C^cQ\text{-}IND} \not\Longrightarrow \mathsf{Q^cQ\text{-}IND}$*) Assume* $\mathcal{K}$ *is a* $\mathsf{C^cQ\text{-}IND}$ *secure KEM and* Π *is a* OW-CPA *secure PKE that is not* OW-qCPA*. Then there exits a KEM* $\mathcal{K}'$ *that is* $\mathsf{C^cQ\text{-}IND}$ *secure but not* $\mathsf{Q^cQ\text{-}IND}$ *secure.*

Proof. First we note that an RSA based PKE may be used for Π as it is OW-CPA, but not OW-qCPA. To prove this proposition a backdoor is built into $\mathcal{K}'$ so that a quantum adversary with access to the decapsulation oracle is able to recover the secret key so that they may compute decapsulation queries themselves, including on the challenge ciphertext and thus win the $\mathsf{Q^cQ\text{-}IND}$ experiment with probability 1. $\mathcal{K}'$ is shown in Figure 4.2.1.

$\mathcal{K}'.\mathsf{KeyGen}()$:

1. $(pk_{\mathcal{K}}, sk_{\mathcal{K}}) \leftarrow\!\!\$\ \mathcal{K}.\mathsf{KeyGen}()$
2. $(pk_{\Pi}, sk_{\Pi}) \leftarrow\!\!\$\ \Pi.\mathsf{KeyGen}()$
3. $\delta^* \leftarrow \$\{0,1\}^{256}$
4. $bd \leftarrow \Pi.\mathsf{Enc}(\delta^*, pk_{\Pi})$
5. return $((pk_{\mathcal{K}}, pk_{\Pi}, bd), (sk_{\mathcal{K}}, \delta^*)) = (pk', sk')$

$\mathcal{K}'.\mathsf{Encaps}(pk')$:

1. $(c, k) \leftarrow \mathcal{K}.\mathsf{Encaps}(pk_{\mathcal{K}})$
2. return $((c, k)$

$\mathcal{K}'.\mathsf{Decaps}((sk', c)$:

1. If $c = \delta^*$ return $sk_{\mathcal{K}}$
2. Else: $\mathcal{K}.\mathsf{Decaps}(sk_{\mathcal{K}}, c)$

Figure 4.2.1: A KEM $\mathcal{K}'$ that is $\mathsf{C^cQ\text{-}IND}$ secure but not $\mathsf{Q^cQ\text{-}IND}$ secure

Clearly $\mathcal{K}'$ cannot be $\mathsf{Q^cQ\text{-}IND}$ secure: in stage 1 the adversary is locally quantum and as Π is not secure against a passive quantum adversary, the secret value δ^* can be recovered and queried to the decapsulation oracle to obtain the secret decapsulation key. Thus $\mathcal{K}'$ is not $\mathsf{Q^cQ\text{-}IND}$-secure.

To show $\mathcal{K}'$ is $\mathsf{C^cQ\text{-}IND}$ suppose that it is not, that there exists an efficient adversary, $\mathcal{A}$, that can break the $\mathsf{C^cQ\text{-}IND}$ security of $\mathcal{K}'$. Then there exists an adversary $\mathcal{B}$ that can break the $\mathsf{C^cQ\text{-}IND}$ security of $\mathcal{K}$. In the $\mathsf{C^cQ\text{-}IND}$ experiment against $\mathcal{B}$, once $\mathcal{B}$ is given $(pk^*_{\mathcal{K}}, c^*, \kappa^*_b)$, $\mathcal{B}$ then runs its own instance of Π generating a pair of keys (pk_{Π}, sk_{Π}), selects δ^* at uniform and encrypts it, and sends $((pk^*_{\mathcal{K}}, pk_{\Pi}, bd), c^*, \kappa^*_b)$ to $\mathcal{A}$. Now whenever $\mathcal{A}$, in its first stage, queries the decapsulation oracle of $\mathcal{K}'$on a ciphertext $c \neq \delta^*$, $\mathcal{B}$ forwards that query to its own decapsulation oracle for $\mathcal{K}$ and returns the answer to $\mathcal{A}$. If $\mathcal{A}$ queries the decapsulation oracle on δ^*, $\mathcal{B}$ returns $\perp$. However this only happens with probability $\frac{q}{2^{256}}$, where q is the number of queries $\mathcal{A}$ makes. Now since Π is $\mathsf{OW\text{-}CPA}$ and both $\mathcal{A}$ and $\mathcal{B}$ are in their first stage, which are classical, any query about δ^* would contradict the assumption of $\mathsf{OW\text{-}CPA}$ security. Thus making the first stage of $\mathcal{A}$, $\mathcal{A}_1$, an oracle algorithm which contradicts the $\mathsf{OW\text{-}CPA}$ security assumption of Π. Finally once an adversary transitions to its second stage $\mathcal{K}'$ and $\mathcal{K}'$ become identical as being able to recover δ^* provides no advantage in the second stage. Thus $\mathcal{K}'$ is $\mathsf{C^cQ\text{-}IND}$-secure.

□

Lastly, we separate quantum with classical queries and fully quantum. We do this is a manner mirroring Bindel *et al*'s [7] separation of $\mathsf{Q^cQ}$ and $\mathsf{Q^qQ}$ for existential unforgeability of digital signature schemes, using a quantum-secure family of pseudorandom permutations, and the hidden linear structure problem to take advantage of the ability to perform decapsulation queries in superposition.

Proposition 4. *($\mathsf{Q^cQ\text{-}IND} \not\Longrightarrow \mathsf{Q^qQ\text{-}IND}$) Assume that there exists a quantum-secure family*

of pseudorandom permutations. Furthermore, assume there exists a $\mathsf{Q^cQ}$ *secure KEM* $\mathcal{K}$ *whose ciphertext are at least* 3λ *bits longs, where* $2^{-\lambda}$ *is considered intractable. Then there exists a KEM* $\mathcal{K}'$ *that is* $\mathsf{Q^cQ}$-IND *secure but not* $\mathsf{Q^qQ}$-IND *secure.*

As in the previous proof, the key is to construct a KEM with a backdoor that is exploitable to adversaries with quantum access to the decapsulation oracle but unexploitable to adversaries with classical access to the oracle. To do this the quantum safe hidden linear structure problem is used as it has constant query complexity with quantum oracle access, but has exponential query complexity with only classical oracle access. The requirement that the ciphertext be at least 3λ bits long is due to parsing bits of the ciphertext into three parts each of length at least λ to hide the key to the backdoor.

Definition 15 ([11]). *Let* $Perm(S)$ *denote the set of all permutations on a set* S*. Given oracle access to* $\mathcal{B}_{s,\pi}(x,y) = (x, \pi(y \oplus sx))$*, where* $x, y, s \in GF(2^n)$ *and* $\pi \in Perm(\{0,1\}^n)$ *with* s *and* π *chosen uniformly at random. The hidden linear structure problem is to determine* s*.*

The hidden linear structure problem has constant query complexity with quantum oracle access for a $\mathsf{Q^qQ}$ adversary but it is hard for a $\mathsf{Q^cQ}$ adversary as it is required for our proof.

Theorem 1 ([11], [7]). *The hidden linear structure problem has query complexity* $\Omega(2^{n/2})$ *for classical queries, and* 1 *for quantum queries. More specifically, there exists a quantum algorithm which solves the hidden linear structure problem with 1 query and probability 1, while any algorithm which queries the oracle classically and uses* 2^b *queries with* $2b \leq n-2$ *outputs the correct* s *with probability at most* 2^{2b-n+1}*.*

Following the idea of [7], we use a restricted version of the hidden linear structure problem which replaces π with a pseudorandom permutation: the quantum-safe hidden linear structure problem. It is indistinguishable from the hidden linear structure problem in time d with advantage greater than δ if there exists a (d,δ)-quantum indistinguishable family of secure pseudorandom permutations as defined next [7]. Let $\mathcal{P} = \{\pi_t : t \in \{0,1\}^k\}$ be a family of pseudorandom permutations on $\{0,1\}^l$. We say that $\mathcal{P}$ is of pseudorandom permutations on the set $\{0,1\}^l$ is $(c_\mathcal{P}, p_\mathcal{P})$-quantum-indistinguishable if no quantum algorithm with running time less than $c_\mathcal{P}$ can win the following indistinguishability game with advantage more than $p_\mathcal{P}$ when using $c_\mathcal{P}$ quantum oracle queries:

1. $a \leftarrow_\$ \{0,1\}$
2. If $a = 0$, then $\pi \leftarrow_\$ \mathcal{P}$. Else $\pi \leftarrow_\$ Perm(\{0,1\}^l)$.
3. $a' \leftarrow_\$ \mathcal{A}^{\pi(\cdot)}$
4. $\mathcal{A}$ wins if $a' = a$

Definition 16 ([7]). *The quantum-safe hidden linear structure problem is a hidden linear structure problem where* π *is drawn from a set* $\mathcal{P}$ *of quantum-indistinguishable pseudorandom permutations.*

Finally, we can prove the separation of $\mathsf{Q^qQ}$ and $\mathsf{Q^cQ}$.

Proof of Proposition 4($\mathsf{Q^cQ} \not\Longrightarrow \mathsf{Q^qQ}$). Suppose that there exists a quantum-secure pseudorandom family of permutations. Furthermore, assume there exists a $\mathsf{Q^cQ}$-IND secure KEM $\mathcal{K} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ whose ciphertexts are at least 3λ bits long, where $2^{-\lambda}$ is considered intractable. We define $c.x$ to be the first λ bits, $c.y$ to be the second λ bits, and $c.z$ to be the remaining bits of c's bit representation.

Let $\mathcal{B}_{s,t}$ be a classical oracle for the quantum-safe hidden linear structure problem that is trying to guess s. Let $\mathcal{B}_{s,t} : GF(2^k) \times GF(2^k) \to GF(2^k) \times GF(2^k)$ be a family of functions such that, given oracle access to $\mathcal{B}_{s,t}$, at least $c_{\mathcal{B}}$ classical queries are required to determine s with probability greater than $p_{\mathcal{B}}$, whereas a single query suffices when given access to $\mathcal{B}_{s,t}$ via a quantum oracle. In our construction, s will be a secret which will unlock access to the decapsulation key sk'. The second parameter, t, needs to be secret but is otherwise not important for our application. The KEM $\mathcal{K}' = (\mathsf{KeyGen}', \mathsf{Encaps}', \mathsf{Decaps}')$ that is $\mathsf{Q^cQ}$-IND secure but not $\mathsf{Q^qQ}$-IND secure is defined in Figure 4.2.2.

$\mathcal{K}'.\mathsf{KeyGen}()$:

1. $(pk, sk) \leftarrow\!\!\$\ \mathcal{K}.\mathsf{KeyGen}()$
2. $s \leftarrow\!\!\$\ \{0,1\}^\lambda$
3. $t \leftarrow\!\!\$\ \{0,1\}^\lambda$
4. $pk' \leftarrow pk$
5. $sk' \leftarrow (sk, s, t)$
6. return (pk', sk')

$\mathcal{K}'.\mathsf{Encaps}(pk)$:

1. $(c, k) \leftarrow \mathcal{K}.\mathsf{Encaps}(pk)$
2. return (c, k)

$\mathcal{K}'.\mathsf{Decaps^c}(sk, s, t, c)$:

1. $k \leftarrow \mathcal{K}.\mathsf{Decaps}(sk, c)$
2. $(u, v) \leftarrow \mathcal{B}_{s,t}(c.x, c.y)$
3. $w \leftarrow sk \cdot \delta_{s,c.z}$
4. return $(k, (u, v), w)$

$\mathcal{K}'.\mathsf{Decaps^q}(sk, s, t, |c, \alpha, \beta, \gamma, \epsilon, z\rangle)$:

1. return $|c, \alpha \oplus k, \beta \oplus u, \gamma \oplus v, \epsilon \oplus w, z\rangle$

Figure 4.2.2: Description of separating KEM $\mathcal{K}'$ that is $\mathsf{Q^cQ}$ -secure, but not $\mathsf{Q^qQ}$ -secure and the quantum oracle for $\mathcal{B}_{s,t}(\cdot)$.

Oracle $\mathcal{B}_{s,t}(|c, 0, 0, 0, 0, \theta, \eta, z\rangle)$:

1. $|c, k, u, v, w, z\rangle \leftarrow \mathcal{K}'.\mathsf{Decaps^q}(sk, s, t, |c, 0, 0, 0, 0, z\rangle)$
2. return $|c, 0, 0, 0, 0, \theta \oplus u, \eta \oplus v, z\rangle$

Figure 4.2.3: The description of the conversion of the quantum decapsulation algorithm of $\mathcal{K}'$ to a quantum $\mathcal{B}_{s,t}$.

We note that $\delta_{s,c.z}$ represents the Kronecker delta function on input s and $c.z$. We now show the $\mathsf{Q^cQ\text{-}IND}$ security of $\mathcal{K}'$. Suppose that it takes at least $q_{\mathcal{B}}$ queries to $\mathcal{B}_{s,t}$ to determine s with probability $p_{\mathcal{B}}$, and that it take at least t_K time for an adversary $\mathcal{A}$ to break the $\mathsf{Q^cQ\text{-}IND}$ security of $\mathcal{K}$ with probability p_K. Note that $\mathcal{K}$ and s are unrelated. Hence, knowledge of the public key pk and access to the classical decapsulation algorithm does not reduce the complexity of guessing s. Likewise, access to a classical oracle for $\mathcal{B}_{s,t}$ does not increase the advantage of an adversary during the $\mathsf{Q^cQ\text{-}IND}$ experiment of $\mathcal{K}$ as we explain next. Assume the contrary, i.e., the advantage of an adversary $\mathcal{A}$ against $\mathcal{K}$ would increase when having access to an oracle for $\mathcal{B}_{s,t}$. Then $\mathcal{A}$ could choose s, t uniformly at random and simulate $\mathcal{B}_{s,t}$.

So the only relation between them is the element $w = sk \cdot \delta_{s,c.z}$ in the decapsulation algorithm of $\mathcal{K}'$. $\mathcal{A}$ only learns information from w, if $w \neq 0$ and thus $w = sk$. By definition of $\delta_{s,c.z}$ this happens if and only if the input c to the classical decapsulation algorithm is such that $c.z = s$. But this only possible if $\mathcal{A}$ solves the quantum safe hidden linear structure problem or guesses s directly. Thus by the following lemma below, we know that $\mathcal{K}'$ is $\mathsf{Q^cQ\text{-}IND}$ secure.

Lemma 1. *([11], [7])*

Suppose that it takes at least $q_{\mathcal{B}}$ queries to $\mathcal{B}_{s,t}$ to determine s with probability $p_{\mathcal{B}}$, and that it takes at least $t_{\mathcal{K}}$ time for an adversary $\mathcal{A}$ to break the $\mathsf{Q^cQ\text{-}IND}$ security of $\mathcal{K}$ with probability $p_{\mathcal{K}}$. If $\mathcal{A}$ has access to a classical oracle $\mathcal{O}^{\mathsf{c}}_{D'}$, knows pk', and runs for time $t < \min\{q_{\mathcal{B}}, t_{\mathcal{K}}\}$, then $\mathcal{A}$ breaks the $\mathsf{Q^cQ\text{-}IND}$ security of $\mathcal{K}'$ with probability at most $p \leq p_{\mathcal{B}} + p_{\mathcal{K}} + 2^{-\lambda}t$.

Proof. Since $t < q_{\mathcal{B}}$, $\mathcal{A}$ has to have made fewer than $q_{\mathcal{B}}$ queries. Hence, $\mathcal{A}$ learns s with probability at most $p_{\mathcal{B}}$ by assumption. The probability of learning s by guessing is at most $2^{-\lambda}t$. So the decapsulation oracle the classical decapsulation algorithm returns an answer of the form $(\cdot, \cdot, \cdot, sk)$ (and hence $\mathcal{A}$ breaks $\mathcal{K}$) with probability no more than $p_{\mathcal{B}} + 2^{-\lambda}t$.

Since $t < t_K$, the probability that $\mathcal{A}$ distinguishes between a real and random key to win the $\mathsf{Q^cQ\text{-}IND}$ game for $\mathcal{K}$ is at most $p_{\mathcal{K}}$, unless one of the above cases applies. Distinguishing a real or random key for $\mathcal{K}'$ implies distinguishing a real or random key for $\mathcal{K}$. Thus the probability that $\mathcal{A}$ wins the $\mathsf{Q^cQ\text{-}IND}$ game for $\mathcal{K}'$ is at most $p \leq p_{\mathcal{B}} + p_{\mathcal{K}} + 2^{-\lambda}t$. □

Next we show that $\mathcal{K}'$ is not $\mathsf{Q^qQ\text{-}IND}$ secure. By the theorem, the quantum safe hidden linear structure problem is in fact solvable in one quantum query to a $\mathcal{B}_{s,t}$ oracle. We construct the

quantum oracle for $\mathcal{B}_{s,t}$ as given in Figure 4.2.3. The quantum $\mathcal{B}_{s,t}$ oracle runs on input of the form $c' = |c, 0, 0, 0, 0, \theta, \eta\rangle$ and returns $|c, 0, 0, 0, 0, \theta \oplus u, \eta \oplus v\rangle$. By construction of the quantum decapsulation algorithm, it holds that $(u, v) = \mathcal{B}_{s,t}(c.x, c.y)$.

Hence, the quantum decapsulation algorithm can be turned into a quantum $\mathcal{B}_{s,t}$ oracle. Thus, an adversary with quantum access can determine the secret s by solving the quantum safe hidden linear structure problem. Afterward the adversary can recover the decapsulation key sk, and win the $\mathsf{Q^cQ\text{-}IND}$ game with probability 1. Thus, $\mathcal{K}'$ is not $\mathsf{Q^qQ\text{-}IND}$ secure. □

Thus, the $\mathsf{X^yZ\text{-}IND}$ security notions for KEMs are nontrivial and separated. We can now order the four $\mathsf{X^yZ\text{-}IND}$ notions based on the separations and implications as such:

$$\mathsf{C^cC\text{-}IND} < \mathsf{C^cQ\text{-}IND} < \mathsf{Q^cQ\text{-}IND} < \mathsf{Q^qQ\text{-}IND}.$$

In Chapter 5 we construct a combiner that takes as input two KEMs and a MAC, and say that it preserves the max $\mathsf{X^yZ\text{-}IND}$ security of the two KEMs if the MAC is equally $\mathsf{X^yZ\text{-}OTS}$ secure, we use this order to determine the max security of the two input KEMs.

Chapter 5

XtM Combiner & $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ Security

In this chapter we construct a secure hybrid KEM combiner the using robust combiners, then prove the security of the resulting KEM in the two stage adversary model, and discuss how to construct the MAC.

It was shown by Giacon *et al.* [13] that the simplest KEM combiner, concatenation of the ciphertexts and XOR of the keys, does not preserve security past IND-CPA in general. This is easy to see: by isolating a single ciphertext component, c_i^* in the challenge ciphertext, $c^* = c_1^* \| ... \| c_i^* \| ... \| c_n^*$, and replacing all other components with ciphertexts with known keys, the adversary can query the modified ciphertext to the decapsulation oracle, and piecewise recover each key to compute the XOR themselves. The KEM combiner we build is a XOR-then-MAC combiner, denoted by XtM, prevents such attacks. It is built using an exclusive-or of the two keys k_1, k_2 of the two input KEMs, but then computes a MAC over the ciphertexts, whose key is derived from the XOR of the keys. The use of the MAC over the ciphertexts prevents the attack described as the adversary must forge a new tag for each of the queries to the decapsulation oracle. Note that in this construction the only additional assumption needed is that the MAC is OTS secure against the same type of adversary as the secure KEM.

5.1 XtM Combiner

We now prove the main result of this thesis, that the XOR-then-MAC combiner, described in Figure 5.1.1, is a robust combiner for KEMs that are up to $\mathsf{Q}^{\mathsf{c}}\mathsf{Q}$-IND secure. That is to say the resulting KEM is as secure as the strongest KEM input, provided the MAC is also equally secure.

Theorem 2. *Let $\mathcal{K}_1$ be a $\mathsf{X}^{\mathsf{c}}\mathsf{Z}$-secure KEM, $\mathcal{K}_2$ a $\mathsf{U}^{\mathsf{c}}\mathsf{W}$-secure KEM, and $\mathcal{M}$ a $\mathsf{R}^{\mathsf{c}}\mathsf{T}$-OTS-secure MAC, where $\mathsf{R}^{\mathsf{c}}\mathsf{T} = \max\{\mathsf{X}^{\mathsf{c}}\mathsf{Z}, \mathsf{U}^{\mathsf{c}}\mathsf{W}\}$. Then $\mathsf{XtM}[\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}]$ is a $(1,2)$- robust combiner for $\mathsf{R}^{\mathsf{c}}\mathsf{T}$-IND-security. More precisely, for any efficient adversary $\mathcal{A}$ that breaks the $\mathsf{R}^{\mathsf{c}}\mathsf{T}$-IND security*

of $\mathsf{XtM}[\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}]$*, then there exists efficient adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *and* $\mathcal{B}_3$ *such that*

$$\mathsf{Adv}^{\mathsf{R^cT\text{-}IND}}_{\mathsf{XtM}[\mathcal{K}_1,\mathcal{K}_2,\mathcal{M}]} \leq 2 \cdot \min\{\mathsf{Adv}^{\mathsf{R^cT\text{-}IND}}_{\mathcal{K}_1}(\mathcal{B}_1), \mathsf{Adv}^{\mathsf{R^cT\text{-}IND}}_{\mathcal{K}_2}(\mathcal{B}_2)\} + \mathsf{Adv}^{\mathsf{R^cT\text{-}OTS}}_{\mathcal{M}}(\mathcal{B}_3),$$

where the run times of all $\mathcal{B}_i$ *are approximately equal to that of* $\mathcal{A}$*, and* $\mathcal{B}_3$ *makes at most as many queries to the verification oracle as* $\mathcal{A}$ *does to the decapsulation oracle.*

$\mathsf{XtM.KeyGen}(1^n)$:	$\mathsf{XtM.Encaps}(pk_1, pk_2)$:	$\mathsf{XtM.Decaps}((sk_1, sk_2), ((c_1, c_2), \tau))$:
1. $(pk_1, sk_1) \leftarrow\!\!\$\ \mathsf{KeyGen}_1(1^n)$	1. $(c_1, k_1) \leftarrow \mathsf{Encaps}_1(pk_1)$	1. $k'_1 \leftarrow \mathsf{Decaps}_1(sk_1, c_1)$
2. $(pk_2, sk_2) \leftarrow\!\!\$\ \mathsf{KeyGen}_2(1^n)$	2. $(c_2, k_2) \leftarrow \mathsf{Encaps}_2(pk_2)$	2. $k'_2 \leftarrow \mathsf{Decaps}_2(sk_2, c_2)$
3. $pk \leftarrow (pk_1, pk_2)$	3. $k_{\mathsf{kem}} \Vert k_{\mathsf{mac}} \leftarrow k_1 \oplus k_2$	3. $k'_{\mathsf{kem}} \Vert k'_{\mathsf{mac}} \leftarrow k'_1 \oplus k'_2$
4. $sk \leftarrow (sk_1, sk_2)$	4. $c \leftarrow (c_1, c_2)$	4. if $\mathsf{Verify}_{k'_{\mathsf{mac}}}((c_1, c_2), \tau) = 0$: return $\perp$
5. return (pk, sk)	5. $\tau \leftarrow \mathsf{MAC}_{k_{\mathsf{mac}}}(c)$	5. else: return k'_{kem}
	6. return $((c, \tau), k_{\mathsf{kem}})$	

Figure 5.1.1: KEM constructed by the XOR-then-MAC combiner $\mathsf{XtM}[\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}]$ with MAC $\mathcal{M} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$.

Proof. Assume there exists an adversary, $\mathcal{A}$, that does break $\mathsf{R^cT\text{-}IND}$ security of XtM, then it is possible to construct further adversaries that breaks the $\mathsf{R^cT\text{-}IND}$ security of either $\mathcal{K}_1, \mathcal{K}_2$, and the $\mathsf{R^cT\text{-}OTS}$ security of $\mathcal{M}$. Furthermore, assume $\mathcal{K}_1$ is $\mathsf{R^cT\text{-}IND}$ secure. By symmetry of the combiner it suffices to consider only the case of constructing an adversary that breaks the $\mathsf{R^cT\text{-}IND}$ security of $\mathcal{K}_1$ as the proof for $\mathcal{K}_2$ would be similar.

$\underline{G_0}$

1. $((pk_1, pk_2), (sk_1, sk_2)) \leftarrow \mathsf{XtM.KeyGen}()$
2. $((c_1^*, c_2^*, \tau^*), k_{\mathsf{kem}}) \leftarrow \mathsf{XtM.Encaps}(pk_1, pk_2)$
3. $k_{\mathsf{kem}} \| k_{\mathsf{mac}} \leftarrow k_1^* \oplus k_2^*$
4. $\tau^* \leftarrow \mathsf{MAC}_{k_{\mathsf{mac}}}(c_1^*, c_2^*)$
5. $\kappa_0^* \leftarrow k_{\mathsf{kem}}$
6. $\kappa_1^* \leftarrow\!\!\$\ KS_{\mathsf{XtM}}$
7. $b \leftarrow\!\!\$\ \in \{0, 1\}$
8. $st \leftarrow \mathcal{A}_1^{\mathcal{O}_D^{c}(\cdot)}(pk_1, pk_2, (c_1^*, c_2^*, \tau), \kappa_b^*)$
9. $b' \leftarrow \mathcal{A}_2(st)$
10. return $[b' = b]$

Figure 5.1.2: Game 0 for the proof of Theorem 2.

G_0: *Game 0* in Figure 5.1.2 is the $\mathsf{R^cT\text{-}IND}$ security experiment for XtM so

$$\mathsf{Adv}^{\mathsf{R^cT\text{-}IND}}_{\mathsf{XtM}[\mathcal{K}_1,\mathcal{K}_2,\mathcal{M}]} = \left| \Pr[G_0 \to 1] - \frac{1}{2} \right|.$$

G_1: The next game is *Game 1*, G_1, in Figure 5.1.3 is made by replacing the key, k_1, corresponding to $\mathcal{K}_1$'s component of the challenge ciphertext, c_1 with a uniformly random and independent value r_1^* from the same keyspace as $\mathcal{K}_1$. This means that the experiment first encapsulates (c_1, k_1) from the $\mathcal{K}_1$ part of the XtM encapsulation algorithm, and then uses (c_1, r_1^*) from then onward for the rest of the encapsulation algorithm, including in generating both the challenge key and the MAC key used to compute the tag, that is $k^*_{\mathsf{kem}} \| k^*_{\mathsf{mac}} \leftarrow r_1^* \oplus k_2^*$. Furthermore, this replacement is done consistently with all decapsulation queries involving the ciphertext c_1. This means the decapsulation oracle is modified so that the step '$k_1 \leftarrow\!\!\$\ \mathsf{Decaps}_1(sk_1, c_1)$" is replaced with the step "if $c_1 = c_1^*$ then $k_1 \leftarrow r_1^* \leftarrow\!\!\$\ K$ else $k_1 \leftarrow \mathsf{Decaps}_1(sk_1, c_1)$" We call this new decapsulation oracle $\mathcal{O}_D^{c^*}$.

$\underline{G_1}$

1. $((pk_1, pk_2), (sk_1, sk_2)) \leftarrow \mathsf{XtM.KeyGen}()$
2. $((c_1^*, c_2^*, \tau^*), k_{\mathsf{kem}}) \leftarrow \mathsf{XtM.Encaps}(pk_1, pk_2)$
3. $k_{\mathsf{kem}} \| k_{\mathsf{mac}} \leftarrow \boxed{r_1^*} \oplus k_2^*$ with $r_1 \leftarrow\!\!\$\ KS_{\mathcal{K}_1}$ // Replacement of the $\mathsf{X^cZ\text{-}IND}$ secure key k_1
4. $\tau^* \leftarrow \mathsf{MAC}_{k_{\mathsf{mac}}}(c_1^*, c_2^*)$
5. $\kappa_0^* \leftarrow k_{\mathsf{kem}}$
6. $\kappa_1^* \leftarrow\!\!\$\ KS_{\mathsf{XtM}}$
7. $b \leftarrow\!\!\$\ \in \{0, 1\}$
8. $st \leftarrow \mathcal{A}_1^{\mathcal{O}_D^{c_1^*}(\cdot)}(pk_1, pk_2, (c_1^*, c_2^*, \tau), \kappa_b^*)$ // Modification on Decap queries
9. $b' \leftarrow \mathcal{A}_2(st)$
10. return $[b' = b]$

Figure 5.1.3: *Game 1* for the proof of Theorem 2.

Claim 1.

$$|\Pr[G_0 \to 1] - \Pr[G_1 \to 1]| \leq 2 \cdot \mathsf{Adv}_{\mathcal{K}_1}^{\mathsf{R^cT\text{-}IND}}(\mathcal{B}_1).$$

Proof Of Claim 1. Suppose that $\mathcal{A}$ can efficiently distinguish between Games 0 and 1, then $\mathcal{A}$ can be used as an oracle algorithm for another adversary, $\mathcal{B}_1$, to break $\mathsf{R^cT\text{-}IND}$ security of $\mathcal{K}_1$ as follows: $\mathcal{B}_1$ is given as input the tuple of public key, challenge ciphertext, and challenge key, (pk_1, c_1^*, k_1^*), where k_1 is either real or uniformly random. $\mathcal{B}$ then simulates $\mathcal{A}$ by first generating the key pair (pk_2, sk_2) for $\mathcal{K}_2$, then runs the encapsulation algorithm of $\mathcal{K}_2$ to generate the second challenge ciphertext portion c_2^* and key share k_2^* itself. It then computes $k_{\mathsf{kem}}^* \| k_{\mathsf{mac}}^* \leftarrow k_1^* \oplus k_2^*$ and assembles the challenge ciphertext (c_1^*, c_2^*, τ^*) where $\tau^* \leftarrow \mathsf{MAC}_{k_{\mathsf{mac}}^*}((c_1^*, c_2^*))$. Finally $\mathcal{B}$ then runs $\mathcal{A}$ on input $((pk_1, pk_2), (c_1^*, c_2^*, \tau^*), k_{\mathsf{kem}}^*)$.

While simulating $\mathcal{A}$, any decapsulation queries of ciphertexts $c = (c_1, c_2, \tau)$ with $c_1 \neq c_1$, are answered as follows: c_1 is decapsulated by using $\mathcal{B}_1$'s decapsulation oracle for $\mathcal{K}_1$, c_2 is decapsulated by $\mathcal{B}_1$ using sk_2, and response, k_{kem}, is computed as the appropriately from the above decapsulations after the verification of the MAC tag by $\mathcal{B}_1$. If $\mathcal{A}$ queries $c = (c_1^*, c_2, \tau)$ then $\mathcal{B}_1$ uses k_1^* as the decapsulation of c_1^* and then continues as described above. Eventually, the distinguisher $\mathcal{A}$ terminates and outputs a guess bit b', and $\mathcal{B}_1$ outputs the same bit b'.

Clearly, $\mathcal{B}_1$ perfectly simulates $\mathcal{A}$ in Game 0 if the k_1^* used was the key encapsulated in c_1^* in the $\mathsf{R^cT\text{-}IND}$ experiment of $\mathcal{K}_1$, and it perfectly simulates Game 1 if k_1^* is random. Furthermore,

in Game 1 the adversary $\mathcal{A}$ always receives a challenge key that is uniformly random as it was either computed by and XOR with a random string r_1^* and k_2^* or selected uniformly at random, whereas, in Game 0 the adversary receives either the real key or a uniform key. This means that we must transition from the prediction based $\mathsf{R^cT\text{-}IND}$ attack, which distinguishes between a real or random key, with a random challenge bit b to an indistinguishability-based comparison between fixed games. Thus, we have

$$|\Pr[G_0 \to 1] - \Pr[G_1 \to 1]| \leq 2 \cdot \mathsf{Adv}^{\mathsf{R^cT\text{-}IND}}_{\mathcal{K}_1}(\mathcal{B}_1).$$

□

G_2: In *Game 2*, Figure 5.1.5, $r_1^* \oplus k_2^*$ is replaced by r^* an independent and uniformly random value, where k_2^* is the key encapsulated by the $\mathcal{K}_2$ challenge ciphertext. This is done consistently with the challenge key and MAC key. This means that challenge key k^*_{kem} the adversary receives is always random and independent of the bit b chosen during the experiment, as well as the MAC tag being computed over a random key. Since r^*, and r_1^* were each chosen independently and uniformly random this means that $r_1^* \oplus k_2^*$ and r^* have identical distributions, as $r_1^* \oplus k_2^*$ is a one time pad, and so the adversary's advantage does not change. Thus,

$$\Pr[G_1 \to 1] = \Pr[G_2 \to 1].$$

G_2

1. $((pk_1, pk_2), (sk_1, sk_2)) \leftarrow \mathsf{XtM.KeyGen}()$
2. $((c_1^*, c_2^*, \tau^*), k_{\mathsf{kem}}) \leftarrow \mathsf{XtM.Encaps}(pk_1, pk_2)$
3. $k_{\mathsf{kem}} \| k_{\mathsf{mac}} \leftarrow \boxed{r^*}$ with $r^* \leftarrow\!\!\$\ KS_{\mathsf{XtM}}$ //Replacement of the XOR, $r_1^* \oplus k_2$
4. $\tau^* \leftarrow \mathsf{MAC}_{k_{\mathsf{mac}}}(c_1^*, c_2^*)$
5. $\kappa_0^* \leftarrow k_{\mathsf{kem}}$
6. $\kappa_1^* \leftarrow\!\!\$\ KS_{\mathsf{XtM}}$
7. $b \leftarrow\!\!\$\ \in \{0, 1\}$
8. $st \leftarrow \mathcal{A}_1^{\mathcal{O}_D^{c^*}(\cdot,\cdot,\cdot)}(pk_1, pk_2, (c_1^*, c_2^*, \tau), \kappa_b^*)$
9. $b' \leftarrow \mathcal{A}_2(st)$
10. return $[b' = b]$

Figure 5.1.4: *Game 2* for the proof of Theorem 2.

G_3

1. $((pk_1, pk_2), (sk_1, sk_2)) \leftarrow \mathsf{XtM.KeyGen}()$
2. $((c_1^*, c_2^*, \tau^*), k_{\mathsf{kem}}) \leftarrow \mathsf{XtM.Encaps}(pk_1, pk_2)$
3. $k_{\mathsf{kem}} \| k_{\mathsf{mac}} \leftarrow r^*$ with $r^* \leftarrow\!\!\$\ KS_{\mathsf{XtM}}$
4. $\tau^* \leftarrow \mathsf{MAC}_{k_{\mathsf{mac}}}(c_1^*, c_2^*)$
5. $\kappa_0^* \leftarrow k_{\mathsf{kem}}$
6. $\kappa_1^* \leftarrow\!\!\$\ KS_{\mathsf{XtM}}$
7. $b \leftarrow\!\!\$\ \in \{0, 1\}$
8. $st \leftarrow \mathcal{A}_1^{\mathcal{O}_D^{c^{**}}(\cdot,\cdot,\cdot)}(pk_1, pk_2, (c_1^*, c_2^*, \tau), \kappa_b^*)$ //Replacement of Decaps oracle with a modified version
9. $b' \leftarrow \mathcal{A}_2(st)$
10. return $[b' = b]$

Decaps oracle $\mathcal{O}_D^{c^{**}}(sk, c_1, c_2, \tau)$:

1. if $c_1 = c_1^*$ then return $\perp$
2. else return $\mathcal{O}_D^c(sk, c_1, c_2, \tau)$

Figure 5.1.5: *Game 3* for the proof of Theorem 2.

G_3: In *Game 3*, Figure 5.1.5, the decapsulation oracle is modified so that it immediately rejects queries of the form $(c_1^*, *, *)$ with output $\perp$, where c_1^* is the $\mathcal{K}_1$ component of the challenge ciphertext. The adversary, $\mathcal{A}$, is only able to notice the difference between *Game 2* and *Game 3* if they query the decapsulation oracle on a ciphertext $(c_1^*, c_2, \tau) \neq (c_1^*, c_2^*, \tau^*)$ where c_2 is a $\mathcal{K}_2$ ciphertext of the adversary's choice and τ is a valid MAC tag. If τ is not a valid tag or the adversary queries the challenge ciphertext, then the decapsulation oracle in *Game 2* would also

return $\perp$. We then show that Claim 2 if an adversary is able to distinguish between *Game 2* and *Game 3* then they can be used as an oracle to break the $\mathsf{R^cT\text{-}OTS}$ security of $\mathcal{M}$.

Claim 2.

$$|\Pr[G_2 \to 1] - \Pr[G_3 \to 1]| \leq \mathsf{Adv}^{\mathsf{R^cT\text{-}OTS}}_{\mathcal{M}}(\mathcal{B}_3).$$

Proof Of Claim 2. Suppose that there exists a $\mathsf{R^cT}$ adversary, $\mathcal{A}$, that can distinguish between *Game 2* and *Game 3*. Now suppose that $\mathcal{B}_3$ is a $\mathsf{R^cT\text{-}OTS}$ adversary of $\mathcal{M}$. Then $\mathcal{B}_3$ can use $\mathcal{A}$ to break $\mathsf{R^cT\text{-}OTS}$ security of $\mathcal{M}$ as follows: $\mathcal{B}_3$ runs $\mathcal{A}$ according to *Game 2*, generating all components (pk_1, sk_1) and (pk_2, sk_2) and c_1^*, c_2^* of $\mathcal{K}_1$ and $\mathcal{K}_2$ respectively itself. To create the MAC tag of the challenge ciphertext, $\mathcal{B}_3$ makes its one-time MAC request on the message (c_1^*, c_2^*) and receives τ^*. It then runs $\mathcal{A}$ on (c_1^*, c_2^*, τ^*) and uniformly random string k^*_{kem}. If $\mathcal{A}$ makes a decapsulation query on (c_1, c_2, τ) where $c_1 \neq c_1^*$ then $\mathcal{B}_3$ is able to use its knowledge of the decapsulation keys to compute the answer itself and return an answer. For queries that contain c_1^* $\mathcal{B}_3$ calls its verification oracle on (c_1, c_2, τ) and returns $\perp$ to $\mathcal{A}$ to continue the simulation. As in *Game 2* the challenge key is either a independent random string r^* if the bit b is 0, or a uniformly random sample key, if the bit is 1. In both cases k^*_{kem} is a uniform key independent of b, as is $\mathcal{B}_3$'s choice of k^*_{kem} in the simulation of $\mathcal{A}$, and the MAC key part is also independent and uniform. The latter holds in $\mathcal{B}_3$ simulation as well as the $\mathsf{OTS\text{-}MVA}$ security experiment chooses a random MAC key. Thus the simulation is identical up until $\mathcal{A}$ makes a query on a fresh ciphertext with a valid MAC tag which would yield a response different from $\perp$. But then $\mathcal{B}_3$ would find a forgery against the MAC, $\mathcal{M}$, in one of its multiple verification attempts. Thus Claim 2 holds true and so

$$|\Pr[G_2 \to 1] - \Pr[G_3 \to 1]| \leq \mathsf{Adv}^{\mathsf{R^cT\text{-}OTS}}_{\mathcal{M}}(\mathcal{B}_3).$$

□

Finally in *Game 3* the challenge key, k^*_{kem}, is an independent string regardless of the secret bit b's value, and decapsulation queries are also independent of b, since the change in the oracles answer only depend on c_1^*. Hence $\mathcal{A}$'s output is independent of the secret bit, and thus

$$\Pr[G_3 \to 1] = \frac{1}{2}.$$

Thus the conclusion of the theorem is obtained by combining the above equalities, that is

$$\mathsf{Adv}^{\mathsf{R^cT\text{-}IND}}_{\mathsf{XtM}[\mathcal{K}_1,\mathcal{K}_2,\mathcal{M}]} \leq 2 \cdot \min\{\mathsf{Adv}^{\mathsf{R^cT\text{-}IND}}_{\mathcal{K}_1}(\mathcal{B}_1), \mathsf{Adv}^{\mathsf{R^cT\text{-}IND}}_{\mathcal{K}_2}(\mathcal{B}_2)\} + \mathsf{Adv}^{\mathsf{R^cT\text{-}OTS}}_{\mathcal{M}}(\mathcal{B}_3).$$

Proving the XOR-then-MAC $\mathsf{XtM}[\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}]$ combiner is robust in preserving $\mathsf{R^cT\text{-}IND}$ security.

□

5.2 Constructing the MAC

We now discuss a type of $\mathsf{Q^cQ\text{-}OTS}$ MAC which can be used for the XOR-then-MAC combiner in the $\mathsf{Q^cQ\text{-}IND}$ setting which does not rely on further cryptographic assumptions. These MACs can be built with universal hash functions using the Carter-Wegman paradigm [25]. This construction requires that the input ciphertexts be larger than the keys, such that the domain of the universal hash function must be extended. For a universal hash function which collides with probability at most ϵ, it is clear that an adversary cannot win after a single verification query after seeing one MAC tag, except with probability at most ϵ. And as verification is deterministic and consists of re-computing the MAC tag, it follows that the adversary cannot win with probability more than $q\epsilon$, where q is the number of verification queries [2].

5.3 $\mathsf{Q^qQ\text{-}IND}$ Security of XtM Combiner

We note that Theorem 2 proves that the XtM combiner preserves security of the input KEMs, if the MAC is equally secure, against attacks in which the adversary has classical access to the decapsulation oracle. However, a proof that the XOR-then-MAC combiner is a $\mathsf{Q^qQ\text{-}IND}$ robust combiner was proved in a joint paper with Fischlin *et al.* [6] on using hybridly secure KEM combiners to design a hybrid authenticated key exchange. We include the outline of the proof they provide in this section. Notably, the MAC need not be $\mathsf{Q^qQ\text{-}OTS}$ secure, but rather only needs $\mathsf{Q^cQ\text{-}OTS}$ security for the proof. This is because the MAC in the challenge is still computed classically, and in the reduction a potential forgery is measured from a decapsulation query in superposition and outputs a classical MAC forgery.

The outline of the proof is as follows: it is very similar to the main theorem with one difference. In *Game 3* the $\mathsf{Q^cQ\text{-}OTS}$ MAC adversary, $\mathcal{B}_3$, can no longer read off a potential MAC forgery from the decapsulation query, from $\mathcal{A}$, in the form of $(c_1^*, *, *)$ for the value c_1^* in the challenge because the query is in superposition. But the "measure-and-modify" approach of Boneh *et al.* [8] for proving quantum resistance of certain styles of encryptions can be adapted to overcome this issue. Specifically, if the amplitudes of entries (c_1^*, c_2, τ) with a valid MAC tag and fresh $(c_2, \tau) \neq (c_2^*, \tau^*)$ in the quantum decapsulation queries would be non-negligible, then we could measure for a randomly chosen query among the polynomial many decapsulation queries to get a classical MAC forgery with non-negligible probability. This contradicts the $\mathsf{Q^cQ\text{-}OTS}$ security of $\mathcal{M}$. If the amplitudes of such queries are negligible, then we can perform the change to the decapsulation oracle so that it returns $\perp$ on queries of the form $(c_1^*, *, *)$. Then following the reasoning by Boneh *et al.* [8], based on Bennett *et al.* [5], this cannot change the adversary's

output behaviour significantly. And since the MAC can be constructed for classical queries information-theoretically, the XtM is a secure KEM combiner in the fully quantum case, without any additional assumptions beyond full quantum resistance of one of the KEMs.

Thus we have that the XtM combiner is a robust combiner for all $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-IND adversaries.

Chapter 6

Summary, Application, & Conclusions

In this chapter we provide a brief summary of key points of the previous chapter, and the discuss an application of the XOR-then-MAC combiner.

6.1 Summary

We first summarize Chapter 2. We defined both *key encapsulation mechanisms* (KEMs) and *message authentication codes* (MACs) as tuples of algorithms with specific characteristics, as well as their purposes. A KEM as a tuple of algorithms used to encapsulate an ephemeral key within a ciphertext for parties to use when sending a message, while a MAC is a tuple of algorithms used to provide a tag on a message intended to authenticate the sender of the message.

We then gave a description of how security is formalized. The *goal-attack* model is used to specify what form of security a cryptographic primitive is attempting to achieve and against which attack powers from an adversary. To define security an *experiment* is created where the adversary is trying to win by achieving some security break condition, which implies they are able to break the security goal of the the primitive, with access to the powers specified by the attack the primitive is trying to be resistant against. We then express the probability of an adversary winning the experiment as an *advantage*, which represents the probability of the adversary winning the security experiment. For indistinguishability based security experiments the advantage is measured as a distance from probability $\frac{1}{2}$.

Next we defined the relevant security notions for KEMs and MACs. For KEMs the primary *goal-attack* is *indistinguishability under chosen ciphertext attack*, denoted as IND-CCA: the adversary must distinguish a key generated using the encapsulation algorithm on a public key from a key selected uniformly at random; the attacker is given the public key, the ciphertext in which the generated key is encapsulated, and access to a decapsulation oracle programmed with the related secret key which they can query with any ciphertext and receive the decapsulation,

with the exception of the ciphertext the adversary receives. For MACs we consider the *one time strongly unforgeable under multiple verification attack*, denoted by OTS-MVA, *goal-attack*, where the adversary is attempting to produce a new valid message-tag pair after receiving a valid tag on a message of their choice; the attacker is given access to a verification oracle that determines if a message-tag is valid.

In Chapter 3 we defined the two stage $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ adversary model, and introduced the $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-IND and $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-OTS security experiments for KEMs and MACs respectively. We use the $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ two stage model as a way to model the transition from classical security to *quantum* security. More specifically, it models an adversary and their interaction with an experiment over time by considering two stages. The first stage is when the adversary has access to the oracles of the experiment and the second stage when they no longer have this access. Furthermore, the $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$ two stage model specifies whether the adversary is classical or quantum while they have access to the experiment oracle, X, whether they have classical or quantum to the experiment oracles, y, and finally whether they are classical or quantum after losing oracle access, Z. This leads to four notions: C^{c}C-IND, C^{c}Q-IND, Q^{c}Q-IND, and Q^{q}Q-IND. We then define the $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-IND and $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-OTS security experiments for KEMs and MACs by relabeling the challenge phase to the query phase in each experiment and modified it so that the first stage adversary outputs a state value, *st*, which is then passed on to the second stage adversary, and then added a challenge phase in which the second stage adversary outputs the guess.

Furthermore, we defined what a robust combiner is and discuss some previous work done with combiners on PKEs and KEMs and the drawbacks of those works. A combiner is an algorithm that accepts as input n of the type of cryptographic schemes to produce a new scheme type, with the property that the combiner preserves some specified *goal-attack* security if at least k of the input are satisfy that security notion. Both Zhang *et al.* [26] and Herzberg [16] constructed combiner for PKEs but neither were able to retain IND-CCA security of PKEs, except in a special case of length preserving PKEs. Giacon *et al.* [13] constructed a KEM combiner that did preserve IND-CCA security, and thus C^{c}C-IND security, but required a special type of pseudorandom functions called split key pseudorandom and lacked a quantum version of their security proof. Our goal was to resolved these gaps while also extending results to the new security notion of $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-IND security.

In Chapter 4 we proved the hierarchy of two stage $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-IND adversaries for KEMs by proving the implications and separations between them. The implications of the hierarchy follow from the fact all classical algorithms and queries can be performed on a quantum computer as well. To show the separations, we constructed KEMs with backdoors that are exploitable to the next adversary in the hierarchy. The separation of C^{c}C-IND and C^{c}Q-IND follows from a concrete example of a C^{c}C-IND KEM that is RSA based and thus any quantum adversary can factor the modulus and recover the decapsulation key. To see the separation of C^{c}Q-IND and Q^{c}Q-IND we constructed a KEM with a decapsulation algorithm that returns the decapsulation key when queried on a secret value encrypted by a PKE that is OW-CPA but not OW-qCPA so that a quantum first stage adversary can exploit this, while a classical first stage adversary cannot. Lastly, for the separation between Q^{c}Q-IND and Q^{q}Q-IND, we constructed a KEM with an exploitable backdoor, based on the quantum-safe hidden linear structure problem, so that an adversary that can decapsulate in superposition could recover the decapsulation key in a single query.

Lastly, in Chapter 5 we constructed the XOR-then-MAC combiner, XtM, for KEMs and proved that it is a robust combiner for all $\mathsf{X}^{\mathsf{y}}\mathsf{Z}$-IND adversaries. We describe the XOR-then-MAC again in Figure 6.1.1 on input KEMs $\mathcal{K}_1$ and $\mathcal{K}_2$, and MAC $\mathcal{M}$.

$\mathsf{XtM.KeyGen}(1^n)$:	$\mathsf{XtM.Encaps}(pk_1, pk_2)$:	$\mathsf{XtM.Decaps}((sk_1, sk_2), ((c_1, c_2), \tau))$:
1. $(pk_1, sk_1) \leftarrow\!\!\$\ \mathsf{KeyGen}_1(1^n)$	1. $(c_1, k_1) \leftarrow \mathsf{Encaps}_1(pk_1)$	1. $k'_1 \leftarrow \mathsf{Decaps}_1(sk_1, c_1)$
2. $(pk_2, sk_2) \leftarrow\!\!\$\ \mathsf{KeyGen}_2(1^n)$	2. $(c_2, k_2) \leftarrow \mathsf{Encaps}_2(pk_2)$	2. $k'_2 \leftarrow \mathsf{Decaps}_2(sk_2, c_2)$
3. $pk \leftarrow (pk_1, pk_2)$	3. $k_{\mathsf{kem}} \| k_{\mathsf{mac}} \leftarrow k_1 \oplus k_2$	3. $k'_{\mathsf{kem}} \| k'_{\mathsf{mac}} \leftarrow k'_1 \oplus k'_2$
4. $sk \leftarrow (sk_1, sk_2)$	4. $c \leftarrow (c_1, c_2)$	4. if $\mathsf{Verify}_{k'_{\mathsf{mac}}}((c_1, c_2), \tau) = 0$: return $\perp$
5. return (pk, sk)	5. $\tau \leftarrow \mathsf{MAC}_{k_{\mathsf{mac}}}(c)$	5. else: return k'_{kem}
	6. return $((c, \tau), k_{\mathsf{kem}})$	

Figure 6.1.1: KEM constructed by the XOR-then-MAC combiner $\mathsf{XtM}[\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}]$ with MAC $\mathcal{M} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$.

6.2 Applications

We now discuss an application of the XOR-then-MAC, done jointly with Fischlin, Bindel, and Brendel to construct a hybrid *authenticated key exchange protocol*. The work in this section was primarily done by my coauthors in [6] but is included here as an example of how the work in this thesis has been applied.

6.2.1 Authenticated Key Exchange

We begin with a description of what an authenticated key exchange (AKE) protocol is and the security definition for AKEs against active attackers. Informally an authenticated key exchange protocol is a method for a set of users to establish a shared session key, while also authenticating one another. Security is modeled against active attackers by granting the adversary a series of oracles which allow them to control all communications and compromise certain secret values, with the goal of distinguishing a session key of an uncompromised session of its choice from random. We will now formally define authenticated key exchange against active attackers using the model of Bellare and Rogaway [3].

Definition 17. *(Parties and sessions) Let* KE *be a key exchange protocol. We denote the set of all participants in the protocol by* $\mathcal{U}$*. Each participant* $U \in \mathcal{U}$ *is associated with a long-term key pair* (pk_U, sk_U)*, created in advance; we assume every participant receives an authentic copy of every other party's public key through some trusted out-of-band mechanism. In a single run of the protocol (referred to as a* session*),* U *may act as either initiator or responder. Any participant* U *may execute multiple sessions in parallel or sequentially.*

We denote by $\pi^j_{U,V}$ *the* j*th session of user* $U \in \mathcal{U}$ *(called the session* owner*) with intended communication partner* V*. Associated to each session are the following per-session variables; we often write* $\pi^j_{U,V}.\mathsf{var}$ *to refer to the variable* var *of session* $\pi^j_{U,V}$*.*

- $\mathsf{role} \in \{\mathsf{initiator}, \mathsf{responder}\}$ *is the role of the session owner in this session.*

- $\mathsf{st}_{\mathsf{exec}} \in \{\mathsf{running}, \mathsf{accepted}, \mathsf{rejected}\}$ *reflects the current status of execution. The initial value at session creation is* $\mathsf{running}$.
- $\mathsf{sid} \in \{0,1\}^* \cup \{\perp\}$ *denotes the session identifier. The initial value is* $\perp$.
- $\mathsf{st}_{\mathsf{key}} \in \{\mathsf{fresh}, \mathsf{revealed}\}$ *indicates the status of the session key* K. *The initial value is* fresh.
- $\mathsf{K} \in \mathcal{D} \cup \{\perp\}$ *denotes the established session key. The initial value is* $\perp$.
- $\mathsf{tested} \in \{\mathsf{true}, \mathsf{false}\}$ *marks whether the session key* K *has been tested or not. The initial value is* false.

To identify related sessions which might compute the same session key, we rely on the notion of partnering using session identifiers. Two sessions $\pi^i_{S,T}$ *and* $\pi^j_{U,V}$ *are said to be* partnered *if* $\pi^i_{S,T}.\mathsf{sid} = \pi^j_{U,V}.\mathsf{sid} \neq \perp$.

We assume that if the adversary has not interfered, sessions in a protocol run between two honest participants are partnered.

The following queries model the adversary's control over normal operations by honest parties:

$\mathsf{NewSession}(U, V, role)$**:** Creates a new session $\pi^j_{U,V}$ for U (with j being the next unused counter value for sessions between U and intended communication partner $V \in \mathcal{U} \cup \{\star\}$) and sets $\pi^j_{U,V}.\mathsf{role} \leftarrow role$.

$\mathsf{Send}(\pi^j_{U,V}, m)$**:** Sends the message m to the session $\pi^j_{U,V}$. If no session $\pi^j_{U,V}$ exists or does not have $\pi^j_{U,V}.\mathsf{st}_{\mathsf{exec}} = \mathsf{running}$, return $\perp$. Otherwise, the party U executes the next step of the key agreement protocol based on its local state, updates the execution status $\pi^j_{U,V}.\mathsf{st}_{\mathsf{exec}}$, and returns any outgoing messages. If $\mathsf{st}_{\mathsf{exec}}$ changes to $\mathsf{accepted}$ and the intended partner V has previously been corrupted, we mark the session key as revealed: $\pi^j_{U,V}.\mathsf{st}_{\mathsf{key}} \leftarrow \mathsf{revealed}$.

The next queries model the adversary's ability to compromise secret values:

$\mathsf{Reveal}(\pi^j_{U,V})$**:** If $\pi^j_{U,V}.\mathsf{st}_{\mathsf{exec}} = \mathsf{accepted}$, $\mathsf{Reveal}(\pi^j_{U,V})$ returns the session key $\pi^j_{U,V}.\mathsf{K}$ and marks the session key as revealed: $\pi^j_{U,V}.\mathsf{st}_{\mathsf{key}} \leftarrow \mathsf{revealed}$. Otherwise, it returns $\perp$.

$\mathsf{Corrupt}(U)$**:** Returns the long-term secret key sk_U of U.

Set $\pi^j_{V,W}.\mathsf{st}_{\mathsf{key}} \leftarrow \mathsf{revealed}$ in all sessions where $V = U$ or $W = U$. (If the security definition is meant to capture forward secrecy, this last operation is omitted.)

The final query is used to define the indistinguishability property of session keys:

$\mathsf{Test}(\pi^j_{U,V})$**:** At the start of the experiment, a test bit b_{test} is chosen uniformly and random and fixed through the experiment. If $\pi^j_{U,V}.\mathsf{st}_{\mathsf{exec}} \neq \mathsf{accepted}$, the query returns $\perp$. Otherwise, it sets $\pi^j_{U,V}.\mathsf{tested} \leftarrow \mathsf{true}$ and proceeds as follows. If $b_{\mathsf{test}} = 0$, a key $\mathsf{K}^* \leftarrow_{\$} \mathcal{D}$ is sampled uniformly at random from the session key distribution $\mathcal{D}$. If $b_{\mathsf{test}} = 1$, K^* is set to the real session key $\pi^j_{U,V}.\mathsf{K}$. Return K^*. The Test query may be asked only once.

6.2.2 Authenticated Key Exchange Security

We now will give a description of the three security notions for AKEs used by Fischlin *et al.* [6] by describing the adversaries win condition in the respective security experiment.

First we discuss two stage BR key secrecy. The adversary is said to have broken BR key secrecy if the adversary is able to distinguish a real session key from a uniformly random key in a session of their choice such that: the adversary has not tested and revealed the session key in a single session or in two partnered sessions. The notion corresponds to the the situation of the adversary successfully obtaining the session key from either one or both parties in a session.

Next we describe the idea of BR match security. The adversary breaks BR match security if they achieve any one of the following:

1. There exist two distinct sessions with the same valid session identifier that have partnered with each other but do not have the same session key.
2. There exist two sessions with the same valid session identifier but do not share the same initiator and responder. That is there exists two sessions with different intended partners.
3. There exist at least three sessions that are pairwise distinct but all three sessions have the same valid session identifier.

BR match security captures the notion that the adversary can not exploit the protocol to match users with someone other than the person they intended.

Lastly, we define the main security definition consider in [6], two stage BR security.

Definition 18 (Two-Stage BR Security)**.** *We call a key exchange protocol* KE $\mathsf{X^cZ}$-BR secure *(with/without forward secrecy) if* KE *provides* BR-Match *security and* $\mathsf{X^cZ}$-BR *key secrecy (with/without forward secrecy).*

6.2.3 Hybrid Authenticated Key Exchange

We will now provide a high-level overview the construction of the hybrid AKE, the main theorem statement from [6], and the role of the XtM combiner.

First we describe the AKE, $\mathcal{C}_{\mathsf{SigMA}}$, which takes as input a KEM $\mathcal{K}$, signature scheme $\mathcal{S}$, a MAC $\mathcal{M}$ and a key derivation function, (KDF) KDF. $\mathcal{C}_{\mathsf{SigMA}}$ works as follows between two parties "Alice" and "Bob" each with long-term public and secret key pairs (pk_A, sk_A) and (pk_B, sk_B) for each signature scheme.

1. Alice initially runs $\mathcal{K}.\mathsf{KeyGen}()$ to obtain an encapsulation-decapsulation key pair (ek_A, dk_A) and send ek_A to Bob.
2. Bob receives ek_A and runs $\mathcal{K}.\mathsf{Encaps}(ek_A)$ to obtain (c, k) then sends Alice c.
3. Alice receives c and decapsulates it to obtain k. Then both Alice and Bob compute the MAC key $k_{\mathsf{mac}} \leftarrow \mathsf{KDF}(k, ek_A, c)$.

4. Alice then sends Bob a random value r_A.
5. Bob receives r_A, and selects their own random value r_B. Then Bob computes a signature using his long-term key, $\sigma_B \leftarrow \mathsf{Sig}(sk_B, r_A \| r_B)$, as well as a MAC on his identity $\tau_B \leftarrow \mathsf{MAC}(k_{\mathsf{mac}}, Bob)$. Finally, Bob sends $(Bob, r_B, \sigma_B, \tau_B)$
6. Alice then verifies the signature and MAC tag using the random values r_A, r_B, Bob's long-term public key pk_B, and the MAC key k_{mac}. Once Alice has validated the message, Alice computes her own signature $\sigma_A \leftarrow \mathsf{Sig}(sk_A, r_A \| r_B)$ and MAC tag on her identity $\tau_A \leftarrow \mathsf{MAC}(k_{\mathsf{mac}}, Alice)$. Then finally sends sends $(Alice, \sigma_A, \tau_A)$.
7. Bob then verifies the signature and MAC tag using the random values r_A, r_B, Alice's long-term public key pk_A, and the MAC key k_{mac}. Once verified Bob computes the session key $K \leftarrow \mathsf{KDF}(k, r_A \| r_B \| Alice \| Bob)$.

We note that the protocol uses the initial KEM interact to establish a MAC key which is later used to authenticate Alice and Bob to one another. Thus ensuring that the KEM is secure is important since if a vulnerable KEM was used a malicious party could obtain the MAC key and forge tags successfully.

We now state the main theorem about the security of the $\mathcal{C}_{\mathsf{SigMA}}$ protocol from [6].

Theorem 3. *Let $\mathcal{K}$ be an* IND-CPA *or* IND-qCPA *key encapsulation mechanism, $\mathcal{S}$ be an* $\mathsf{R^cT}$*-unforgeable signature scheme, $\mathcal{M}$ be a* $\mathsf{U^cW}$*-unforgeable message authentication scheme, and* KDF *be a* $\mathsf{L^cN}$*-secure key derivation function. Then the compiled protocol* $\mathcal{C}_{\mathsf{SigMA}}$ *is* $\mathsf{X^cZ}$*-BR secure with forward secrecy, where*

- $\mathsf{X^cZ} = \mathsf{C^cC}$*, if either the key encapsulation mechanism $\mathcal{K}$ or the key derivation function* KDF *are only classically secure, i.e., if either $\mathcal{K}$ is* IND-CPA *or* $\mathsf{L^cN} = \mathsf{C^cC}$.
- $\mathsf{X^cZ} = \mathsf{Q^cQ}$*, if all components are resistant against fully quantum adversaries, i.e,* $\mathsf{R^cT} = \mathsf{U^cW} = \mathsf{L^cN} = \mathsf{Q^cQ}$ *(and $\mathcal{K}$ is* IND-qCPA*).*
- $\mathsf{X^cZ} = \mathsf{C^cQ}$*, if the employed signature and MAC scheme are at most future-quantum secure, i.e., if* $\mathsf{R^cT}, \mathsf{U^cW} \in \{\mathsf{C^cC}, \mathsf{C^cQ}\}$ *(and $\mathcal{K}$ is* IND-qCPA*,* $\mathsf{L^cN} \geq \mathsf{Q^cQ}$*).*

We note that Theorem 3 only requires the KEM to be either IND-CPA or IND-qCPA secure and not $\mathsf{X^yZ}$-IND secure. Fischlin *et al.* note that to obtain a hybrid AKE, the KEM $\mathcal{K}$ may be instantiated with any hybrid KEM, this includes the XtM combiner. Moreover, the XtM combiner also allows for a simpler choice in KEMs. The resulting security of $\mathcal{C}_{\mathsf{SigMA}}$ varies depending on the security of each of its inputs including the KEM, with the result being strong should the KEM be IND-qCPA, however the XtM can result in a $\mathsf{C^cQ}$ which is both IND-CPA and IND-qCPA secure. This means that such KEMs are ideal choices for this AKE and can be constructed efficiently with the XtM combiner.

We also note that results such as Theorem 3 are important as early designs often become templates for future work, and designing a provably sound method for hybrid key exchange is vital for the transition to post-quantum cryptography.

6.3 Conclusion

In this thesis we have established a series of security notions that consider adversaries with varying levels of quantum power over time, to represent the transition from classical to quantum computers. We have also introduced the XOR-then-MAC combiner, which was built with minimal assumptions and is the first hybrid KEM construct which is secure against a fully quantum adversary.

There is still open questions on this subject, specifically what other KEM combiners are there, what is the $\mathsf{X}^{\mathsf{y}}\mathsf{Z}\text{-}\mathsf{IND}$ security of other combiners, and how efficient are they compared to the XtM combiner.

Bibliography

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 26–45, 1998.

[2] M. Bellare, O. Goldreich, and A. Mityagin. The Power of Verification Queries in Message Authentication and Authenticated Encryption. Cryptology ePrint Archive, Report 2004/309, 2004.

[3] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 232–249, 1994.

[4] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, pages 92–111, 1995.

[5] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and Weaknesses of Quantum Computing. *SIAM Journal on Computing*, 26(5):1510–1523, October 1997.

[6] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila. Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange. Under review, 2018.

[7] N. Bindel, U. Herath, M. McKague, and D. Stebila. Transitioning to a Quantum-Resistant Public Key Infrastructure. PQCrypto, 2017.

[8] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random Oracles in a Quantum World. In D. H. Lee and X. Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 41–69, 2011.

[9] J.L. Carter and M. N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, 18(2):143 – 154, 1979.

[10] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[11] N. de Beaudrap, R. Cleve, and J. Watrous. Sharp Quantum versus Classical Query Complexity Separations. *Algorithmica*, 34(4):449–461, 2002.

[12] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[13] F. Giacon, F. Heuer, and B. Poettering. KEM Combiners. PKC, 2018.

[14] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984.

[15] D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On Robust Combiners for Oblivious Transfer and Other Primitives. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 96–113, 2005.

[16] A. Herzberg. Folklore, Practice and Theory of Robust Combiners. Cryptology ePrint Archive, Report 2002/135, 2002.

[17] H. Jiang, Z. Zhang, L. Chen, H. Wang, and Z. Ma. IND-CCA-Secure Key Encapsulation Mechanism in the Quantum Random Oracle Model, Revisited. Cryptology ePrint Archive, Report 2017/1096, 2017.

[18] J. Katz and Y. Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[19] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 10th edition, 2011.

[20] C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 433–444, 1992.

[21] O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93. ACM, 2005.

[22] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *CACM*, 26(1):96–99, 1983.

[23] P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, FOCS '94, pages 124–134, 1994.

[24] V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332, 2004.

[25] M. N. Wegman and J.L. Carter. New Hash Functions and Their Use in Authentication and Set Equality. *Journal of Computer and System Sciences*, 22(3):265 – 279, 1981.

[26] C. Zhang, D. Cash, X. Wang, X. Yu, and S. S. M. Chow. Combiners for Chosen-Ciphertext Security. In T. N. Dinh and M. T. Thai, editors, *Computing and Combinatorics*, pages 257–268. Springer International Publishing, 2016.