Signature Schemes in the Quantum Random-Oracle Model

by

Edward Eaton

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

Waterloo, Ontario, Canada, 2017

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

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

Abstract

A signature scheme is a fundamental component in modern digital communication. It allows for authenticated messages, without which it would be nearly impossible to ensure security when using most modern technologies. However, there is a growing threat to this fundamental piece of electronic infrastructure. Universal quantum computers, which were originally envisioned by Richard Feynman, have moved from being a theoretical future technology into one that could realistically be available in a matter of decades. In 1994, Peter Shor devised an algorithm that would run on a quantum computer that could be used to solve mathematical problems that formed the foundation of public-key cryptography.

While Shor's algorithm clearly establishes that new mathematical problems must be found and studied that can admit efficient cryptographic protocols, it is equally important that the models in which we consider security are also updated to consider the possibility of a malicious adversary having a quantum computer.

In the random-oracle model, a hash function is replaced by a truly random function that any relevant party is able to query. This model can enable security reductions where otherwise none are known. However, it has been noted that this model does not properly consider the possibility of a quantum computer. For this, we must instead consider the quantum random-oracle model.

In this thesis, we explain the basics of quantum physics and quantum computation in order to give a complete motivation for the quantum random-oracle model. We explain many of the difficulties that may be encountered in the quantum random-oracle model, and how some of these problems may be solved. We then show prove three signature schemes secure in the quantum random-oracle model: the LMS hash-based scheme, TESLA, a lattice-based scheme, and the TOO transformation using chameleon hashes. The first two schemes are strong candidates for post-quantum standardization.

Acknowledgements

I would first like to thank my advisor, Alfred Menezes, for his support, helpful discussions and comments, and editorial assistance in the creation of this thesis.

For chapter 3 I would like to thank my co-authors of the paper [1] from which this chapter lends itself. They are Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, Juliane Krämer, and especially Gus Gutoski and Filip Pawlega.

For chapter 4 I would like to thank my co-author of the paper [21] from which this chapter lends itself, Fang Song, as well as Andrew Childs for helpful discussions.

Lastly I would like to thank my readers for their helpful comments on an earlier draft of this thesis.

Dedication

For Ellen, John, and Maggie.

Table of Contents

1 Introduction **1**

1.1 Quantum Computing Basics 1

1.1.1 The Superposition 1

1.1.2 Composite Systems and Entanglement 3

1.1.3 Transformations of States 4

1.1.4 Measurement 6

1.2 The Random-Oracle Model 6

1.3 Common Mathematical Definitions and Results 7

1.4 ROM Example — Full Domain Hash 8

1.5 Quantum Security Models 11

1.6 QROM Basics and Challenges 14

1.6.1 Oracle Simulation 14

1.6.2 Query Lookup 16

1.6.3 Rewinding 17

1.6.4 Reprogramming 18

1.6.5 Challenge Injection 19

1.6.6 Query Categorization 20

1.7 FDH in the QROM 21

1.8 Thesis Roadmap 22

2 LMS in the QROM **24**
2.1 Scheme Description . . . 24
2.1.1 One-Time Scheme . . . 24
2.1.2 Full Scheme . . . 27
2.2 LMS in the (Quantum) Random-Oracle Model . . . 28
2.2.1 Second-preimage Resistance with Non-uniform First-preimage . . . 29
2.2.2 Second-preimage Resistance with Adversary Prefixes . . . 31
2.2.3 Random Oracle Composition . . . 33
2.2.4 Merkle Trees in the Random-Oracle Model . . . 34
2.3 Security Proof for OTLMS in the QROM . . . 35
2.4 Security Proof for LMS in the QROM . . . 38

3 TESLA **42**
3.1 Chapter Notation . . . 43
3.2 The Learning with Errors Problem . . . 44
3.3 The Signature Scheme TESLA . . . 45
3.4 Brief Sketch of Security Proof for TESLA . . . 47
3.4.1 Yes-Instances of LWE . . . 48
3.4.2 No-Instances of LWE . . . 49
3.5 Overview of Security Proof . . . 50
3.5.1 Notation and Sizes for Various Sets of Vectors . . . 52
3.6 Yes-Instances of LWE . . . 53
3.6.1 Adaptively Chosen Queries . . . 53
3.6.2 The Distinguisher's State, a First Look . . . 54
3.6.3 Mid-Sign . . . 55
3.6.4 Consistent-Mid-Sign . . . 57
3.6.5 Consistent-Mid-Sign is a Mixture of Real Sign Oracles . . . 61
3.6.6 Re-Programming of Hash Oracles is Hard to Detect . . . 62

3.6.7 The Distinguisher's State, Revisited . . . 64
3.6.8 Probability of Well-Roundedness . . . 66
3.6.9 Probability of Repetition . . . 68
3.7 No-Instances of LWE . . . 71
3.7.1 Correspondence Between Valid Signatures and Good Hash Inputs . . . 71
3.7.2 The Fraction of Hood Hash Inputs . . . 72
3.7.3 Good Hash Inputs are Rare . . . 73
3.7.4 Forgers Cannot Forge on LWE No-Instances . . . 75
3.8 Security: Putting it all Together . . . 76

4 Strong Unforgeability in the QROM **78**
4.1 Chameleon Hash functions . . . 79
4.2 The TOO Transformation . . . 80
4.3 TOO Classical Proof . . . 81
4.4 Quantum Challenges and Tools . . . 84
4.5 TOO Quantum Proof . . . 90
4.5.1 Case 1 . . . 90
4.5.2 Case 2 . . . 91
4.6 Instantiation . . . 93

5 Conclusion **94**
5.1 Future Work . . . 95

References **96**

Chapter 1

Introduction

1.1 Quantum Computing Basics

Models of computation are varied and complex, but ultimately they all come down to three steps: information is encoded onto some physical construction, some physical transformations take place on this construction, and then information is somehow extracted from the construction.

For example, in the Turing machine model of computation, information is initially encoded by symbols on a tape, and then a machine reads and modifies the symbols on the tape according to some instruction set, and when it is finished, the symbols on the tape are read off.

Quantum computing is simply the idea that the physical construction that information can be encoded into can be quantum mechanical in nature, as can the physical transformation that changes that construction. As this computation model is rooted in quantum mechanics, a small amount of quantum mechanics basics is needed to understand this model. For a more complete discussion of the principles of quantum mechanics and how it leads to a formalism of quantum computing we refer to [29].

1.1.1 The Superposition

In classical physics, the *state* of a system is described by the values of a set of variables. For example, a particle travelling through empty space has some position and momentum that describes the system. For the purpose of computation, we generally consider systems in

which the relevant components take on binary values. A wire either has current travelling through it, or doesn't, a ferromagnetic film has one of two polarizations, or a cat in a box is either alive or dead.

The concept of a superposition is a fundamentally new way of describing a system. It is sometimes inaccurately described as being a probabilistic description of a system (e.g., the cat is alive with probability 1/2 and dead with probability 1/2) or a continuous description of a system (e.g., half of a unit of current travels through a wire). While the concept of a superposition is related to these ideas, it cannot be fully described by either.

A superposition of a binary system corresponds to a two-dimensional Hilbert space where the basis states correspond to the two 'classical' states. Consider a particle of light propagating through space. This particle can either oscillate vertically, represented by the state $|\updownarrow\rangle$, or horizontally, represented by the state $|\leftrightarrow\rangle$ (here we are using Dirac's bra-ket notation for quantum mechanical systems).

Working classically, we could only say that the polarization of the light is in the state $|\updownarrow\rangle$ or $|\leftrightarrow\rangle$. But in fact the polarization of light allows for a superposition of these two states. So the state of the system can also be mathematically described by expressions such as $\frac{1}{\sqrt{2}}|\updownarrow\rangle + \frac{1}{\sqrt{2}}|\leftrightarrow\rangle$, or $\frac{1}{2}|\updownarrow\rangle - \frac{\sqrt{3}i}{2}|\leftrightarrow\rangle$, or even $\left(\frac{1}{\sqrt{3}} + \frac{i}{\sqrt{3}}\right)|\updownarrow\rangle + \left(\frac{\sqrt{2}}{3} - \frac{i}{3}\right)|\leftrightarrow\rangle$. The general superposition is

$$\alpha|\updownarrow\rangle + \beta|\leftrightarrow\rangle$$

where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$.

While all superposition of the polarization of light can be described in this way, not all values of α and β correspond to different superpositions. Superpositions are only unique up to a unitary constant. So $\frac{1}{\sqrt{2}}|\updownarrow\rangle + \frac{i}{\sqrt{2}}|\leftrightarrow\rangle$ and $\frac{i}{\sqrt{2}}|\updownarrow\rangle - \frac{1}{\sqrt{2}}|\leftrightarrow\rangle$ correspond to the same superposition, as they are equal to each other up to a multiplication of i. For this reason, we can write the general state instead as

$$\cos\frac{\theta}{2}|\updownarrow\rangle + \sin\frac{\theta}{2}e^{i\phi}|\leftrightarrow\rangle$$

where $\theta, \phi \in [0, 2\pi)$.

A natural question to ask is what a superposition corresponds to in physical reality. For this example, this is not a complicated question — θ and ϕ correspond to the elliptical polarization of the light. For general quantum mechanical systems however, this is a philosophically complicated issue. The concept of a superposition does not correspond to any notion that we, as macroscopic beings with a macroscopic view of the universe,

are comfortable with. Nonetheless, experiments have consistently confirmed it to be the underpinning formalism of many systems, and so we are forced to accept and work with it.

While questions such as what it means for a cat to be in the state $\frac{1}{\sqrt{2}}|alive\rangle + \frac{1}{\sqrt{2}}|dead\rangle$ are interesting, they are moot for our purposes, as we are more interested in the question of what such a cat means for the purpose of computation. For encoding classical information, a bit is encoded in some discrete two-state system. For quantum information, the fundamental unit is a two-state quantum system, or a *qubit*.

1.1.2 Composite Systems and Entanglement

While we now have some formalism to describe a two-state system, we want to extend this formalism for describing composites of these systems. If prepared in a certain way, the current travelling through a loop of superconducting material forms a two-state quantum system. The two basis states are whether the current travels clockwise, $|\circlearrowright\rangle$ or counter-clockwise, $|\circlearrowleft\rangle$. So we may have a system, consisting of a loop prepared in the state $\frac{1}{\sqrt{2}}|\circlearrowright\rangle - \frac{1}{\sqrt{2}}|\circlearrowleft\rangle$ This system may also have a particle of light, polarized to be in the state $\frac{1}{\sqrt{2}}|\updownarrow\rangle + \frac{i}{\sqrt{2}}|\leftrightarrow\rangle$.

Then the overall system is described by the state

$$\left(\frac{1}{\sqrt{2}}|\circlearrowright\rangle - \frac{1}{\sqrt{2}}|\circlearrowleft\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|\updownarrow\rangle + \frac{i}{\sqrt{2}}|\leftrightarrow\rangle\right)$$

where $\otimes$ represents the tensor product. We will write the tensor product of two basis states $|\circlearrowright\rangle \otimes |\updownarrow\rangle$ as $|\circlearrowright\rangle|\updownarrow\rangle$ or as $|\circlearrowright\updownarrow\rangle$ depending on context (the difference is only notational). This allows us to expand the tensor product as

$$\frac{1}{2}|\circlearrowright\rangle|\updownarrow\rangle + \frac{i}{2}|\circlearrowright\rangle|\leftrightarrow\rangle - \frac{1}{2}|\circlearrowleft\rangle|\updownarrow\rangle - \frac{i}{2}|\circlearrowleft\rangle|\leftrightarrow\rangle.$$

From this representation, it is natural to assume that the general system described by this superconducting loop and particle of light is described by

$$\alpha|\circlearrowright\rangle|\updownarrow\rangle + \beta|\circlearrowright\rangle|\leftrightarrow\rangle + \gamma|\circlearrowleft\rangle|\updownarrow\rangle + \delta|\circlearrowleft\rangle|\leftrightarrow\rangle$$

where $\alpha, \beta, \gamma, \delta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$ (as before, a global phase makes no meaningful difference). So our composition of two two-state systems has resulted in a four-dimensional system. Note that the choice of writing the state with the superconducting loop before the polarized particle was arbitrary — writing the basis states with the polarized particle first is entirely equivalent as long as we are consistent.

But note that this formalism allows for very strange effects. While we arrived at this formalism by considering the tensor product of two two-state systems, this arbitrary four dimensional system cannot always be decomposed into such a tensor product. For example, the state $\frac{1}{\sqrt{2}}|\circlearrowleft\rangle|\updownarrow\rangle + \frac{1}{\sqrt{2}}|\circlearrowright\rangle|\leftrightarrow\rangle$ cannot possibly be written as the tensor product of two states.

At first, one may assume that such a state cannot exist, and that any system that can be physically realised can be decomposed into its individual components. However this is not true. Quantum mechanics allows for systems to be in these strange states where the components cannot be separated out. This phenomenon is called entanglement, and its use is an important reason why quantum mechanics allows for efficient computations not possible classically.

The question of what entanglement means is similarly distressing as the concept of a superposition. One can imagine an entangled system where the individual components are separated by large amounts of space so that it is difficult to imagine a physical process by which the components may interact. That quantum mechanics suggests a strong connection between these objects despite the separation seems deeply troubling, and was the inspiration for the EPR 'paradox', published in 1935 by Einstein, Podolsky, and Rosen. But this formalism of quantum mechanics is correct, and these connections, called 'spooky action at a distance' by Einstein, are a fact of quantum mechanics.

1.1.3 Transformations of States

In quantum mechanics, the time evolution of any closed system $|\psi(t)\rangle$ is described by the *Schrödinger equation*:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle$$

where $\hbar$ is a constant, and $\hat{H}$ is a hermitian operator called the Hamiltonian. While the exact details of the solution are not relevant for our purposes, there is one important consequence: the time evolution of a state is *unitary*.

Specifically, if $|\psi(t)\rangle$ describes the state of a system at time t, and $t_1 < t_2$, then there is a *unitary* operator U such that $|\psi(t_2)\rangle = U|\psi(t_1)\rangle$. The fact that the time evolution is unitary has several key consequences. Quantum computing is a transformation on a quantum state, and so this property forces that any part of how our computation is performed must be unitary. Unitary transformations have several key properties. One is that they are *linear*. That is, if a quantum system $|\psi\rangle$ is in a superposition of states so that $|\psi\rangle = \alpha|\psi_1\rangle + \beta|\psi_2\rangle$

then

$$U|\psi\rangle = U(\alpha|\psi_1\rangle + \beta|\psi_2\rangle) = \alpha U|\psi_1\rangle + \beta U|\psi_2\rangle.$$

The other key property is that transformations on quantum states are, in principle, invertible. Unitaries satisfy the property that $UU^* = U^*U = I$, (where U^* is the adjoint matrix) and so any transformation we make on our quantum states must be invertible.

At first, the invertibility of unitary transformations seems like a problem. Basic functions such as computing the AND of two bits are not invertible. So there can be no physical implementation of a quantum system that performs the mapping in a way that is capable of respecting quantum effects like superposition. But non-invertible functions like AND *can* be inverted as long as we do not 'throw away' the input. Let m and n be some integers. Let $f : \{0,1\}^m \to \{0,1\}^n$ be any function, invertible or otherwise. Then we can define another function, $\mathbf{f} : \{0,1\}^{m+n} \to \{0,1\}^{m+n}$ by

$$\mathbf{f}(x, y) = (x, y \oplus f(x)).$$

Note that two applications of $\mathbf{f}$ always map to the original input. So $\mathbf{f}$ is its own inverse.

Because any classical function can be made invertible in this way, any efficient classical algorithm or function can also be efficiently implemented on a quantum computer. A unitary U_f can be created that maps

$$U_f : |x\rangle|y\rangle \to |x\rangle|y \oplus f(x)\rangle. \tag{1.1}$$

To consider how this operator acts on a superposition of states, we simply extend this mapping to obey the linearity property. Let $|\psi\rangle$ be an arbitrary state of $m + n$ qubits. Then U_f acts on $|\psi\rangle$ as

$$\begin{aligned} U_f|\psi\rangle =& U_f \left(\sum_{(x,y)\in\{0,1\}^{m+n}} \alpha_{x,y}|x\rangle|y\rangle \right) \\ =& \sum_{(x,y)\in\{0,1\}^{n+m}} \alpha_{x,y} U_f|x\rangle|y\rangle \\ =& \sum_{(x,y)\in\{0,1\}^{n+m}} \alpha_{x,y} \left(|x\rangle|y \oplus f(x)\rangle\right). \end{aligned}$$

1.1.4 Measurement

We now know how to encode some information and perform transformations on it, but we need to have a method to extract an answer after any transformations. Given an arbitrary state $|\psi\rangle$ over n qubits, we can write it as

$$\sum_{x\in\{0,1\}^n} \alpha_x |x\rangle.$$

To be able to completely identify the state would be to be able to learn α_x for all x (or some suitable approximation of each α_x). However, this is not possible in general. The rules that specify how quantum information can be converted to classical information prevent this.

To convert a quantum state into classical information, we perform a *measurement* according to some basis. Let $\{|\phi_i\rangle\}$ be a set of basis states for n qubits (note that this set has size 2^n). Then any n-qubit state $|\psi\rangle$ can be written in terms of this basis, as $\sum_{i=1}^{2^n} \beta_i |\phi_i\rangle$. Then a measurement according to this basis will leave the system in the state $|\phi_i\rangle$, which is measurable with certainty, with probability $|\beta_i|^2$.

A measurement with respect to the basis $\{|x\rangle\}_{x\in\{0,1\}^n}$ is called a measurement in the computational basis, and any measurement of the previous type can be performed by an application of a change-of-basis unitary and a measurement in the computational basis.

1.2 The Random-Oracle Model

Many (if not most) cryptographic protocols rely on the use of a hash function. A hash function maps bitstrings of arbitrary length to bitstrings of some fixed length, such as 256 bits. While hash functions have very specific security properties that are expected and relied on, these properties are meant to make hash functions resemble random oracles.

A random oracle $\mathcal{O}$ from a domain $\mathcal{D}$ to a codomain $\mathcal{R}$ is a uniformly random function taken from the universe of all functions from $\mathcal{D}$ to $\mathcal{R}$. To properly describe an entire random oracle takes $|\mathcal{D}| \cdot \log_2(|\mathcal{R}|)$ bits, by listing what each element in $\mathcal{D}$ maps to.

As $\mathcal{D}$ generally has fully exponential size, and often is infinite in size, it is impossible to describe or even generate a complete random oracle when $\mathcal{D}$ is large. This is why hash functions are used instead, as a sort of efficiently computable and manageable random oracle that anyone can implement or use.

While random oracles are not useful in the real world, they can still be incredibly useful in the context of security reductions. Whereas protocols in the real world always use hash functions, we can imagine someone using a random oracle, or at least using some construction equivalent to a random oracle. Often we can prove the desired security relations when a random oracle is used, but do not know how, or even cannot come up with a proof when a hash function is used.

Models of security in which a real hash function is used are called *standard models*. Models of security in which a random oracle is used in place of a hash function are called *random-oracle models*.

An important discussion item is validity of using the random-oracle model in a security reduction for a protocol. In the real world, such protocols are deployed with a hash function, so to prove real world security, we want to prove a statement of the form **if** *an underlying problem is hard and a hash function is used* **then** *the protocol is secure*. But in the random-oracle model, we prove the statement **if** *an underlying problem is hard and a random oracle is used* **then** *the protocol is secure*. As these are different theorems, some cryptographers are concerned that reductions in the random-oracle model give no guarantee of security in the real world.

However, the random-oracle model has yet to result in any significant lack in security in practice. Specifically, there has yet to be a protocol that was designed to be secure in the standard model, and proved to have a reduction in the random-oracle model, that was later shown to be insecure in the standard model.

There have, however, been protocols that were intended to be *insecure* in the standard model that can be shown to have a reduction in the random-oracle model. While this certainly shows that the random-oracle model provides no unconditional guarantee of security, these schemes are structured to cause the scheme to interact with the hash function in ways to purposely introduce security flaws [15].

The random oracle therefore continues to be immensely popular in establishing security reductions. It enables security reductions that allow for simpler analysis of the generic security of protocols.

1.3 Common Mathematical Definitions and Results

Definition 1.3.1 (Signature Scheme)**.** *A signature scheme is a triple of algorithms,* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *such that:*

- KeyGen *takes in a security parameter* 1^n *and returns a public key / private key pair,* (pk, sk).
- Sign *takes in a message* msg, *and a secret key* sk, *and returns a signature* σ.
- Vrfy *takes in a message* msg, *a signature* σ, *and a public key* pk, *and outputs either* accept *or* reject.

Signature schemes also satisfy the correctness property, *which is that if* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$, *then for any* msg *we have that*

$$\mathsf{Vrfy}(\mathsf{msg}, \mathsf{Sign}(\mathsf{msg}, sk), pk) \rightarrow \mathsf{accept}.$$

Game 1.3.1 (Existential Unforgeability)**.**

1. $\mathcal{C}$ runs Keygen and sends pk to $\mathcal{A}$.
2. $\mathcal{A}$ queries a message M_1 to $\mathcal{C}$.
3. $\mathcal{C}$ returns $\sigma_1 \leftarrow \mathsf{Sign}(M_1, sk)$ to $\mathcal{A}$.
4. $\mathcal{A}$ and $\mathcal{C}$ repeat steps 2-3 for $M_2, M_3, \ldots, M_{q_S}$.
5. $\mathcal{A}$ outputs (M^*, σ^*) such that $M^* \neq M_i$ for any $i \in \{1, \ldots, q_S\}$.

Definition 1.3.2 (Existential Unforgeability under Chosen Message Attack)**.** *A signature scheme* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is said to be* Existential Unforgeable under Chosen Message Attack *if the probability that any polynomially-bounded adversary* $\mathcal{A}$ *playing game 1.3.1 is able to produce a pair such that* $\mathsf{Vrfy}(M^*, \sigma^*, pk) \rightarrow \mathsf{accept}$ *is negligible in the security parameter.*

1.4 ROM Example — Full Domain Hash

The initial use of the random-oracle model was to establish a security reduction for the RSA full domain hash. The reduction was shown by Bellare and Rogaway, in [6]. While Bellare and Rogaway specifically reduced to the RSA problem, we will present a reduction to a generic trapdoored permutation.

Definition 1.4.1 (Trapdoored Permutation)**.** *A trapdoored permutation on a set* $\mathcal{R}$ *is a pair* (f, f^{-1}) *where:*

- *f is some information (of poly-log size in $|\mathcal{R}|$) that allows anyone to evaluate the permutation $f : \mathcal{R} \to \mathcal{R}$.*
- *f^{-1} is some information (also of size poly-log in $|\mathcal{R}|$) that enables the ability to invert f.*

We can also define a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where $\mathcal{A}$ is trying to invert f without f^{-1}.

Game 1.4.1 (Trapdoor Permutation Inversion)**.**

1. $\mathcal{C}$ chooses a uniformly random $r \in \mathcal{R}$ and sends f, r to $\mathcal{A}$.
2. $\mathcal{A}$ performs some computations, and then outputs an $x \in R$.

$\mathcal{A}$ is said to have won the game if $f(x) = r$.

Using a hash function $H : \mathcal{D} \to \mathcal{R}$ and the trapdoor permutation, we can construct a signature scheme as follows:

Signature Scheme 1.4.1. Full Domain Hash Signature Scheme

Algorithm 1 FDHKeyGen

Input: Security Parameter 1^n
Output: Public key f, secret key f^{-1}

1: Generate and output f and f^{-1}.

Algorithm 2 FDHSign

Input: Message $\mathsf{msg} \in \mathcal{D}$, secret key f^{-1}
Output: Signature $\sigma \in \mathcal{R}$

1: Compute $\sigma \leftarrow f^{-1}(H(\mathsf{msg}))$
2: Return σ

Algorithm 3 FDHVrfy

Input: Message $\mathsf{msg} \in \mathcal{D}$, public key f, signature $\sigma \in \mathcal{R}$
Output: accept or reject

1: Check if $H(\mathsf{msg}) = f(\sigma)$. If so, output accept
2: Otherwise, output reject

We now show our first security reduction, originally established in [6].

Theorem 1.4.1 (Existential Unforgeability of Generic FDH in the ROM)**.** *Let $\mathcal{A}$ be an adversary that wins game 1.3.1 with signature scheme 1.4.1 in time t with probability ϵ having made q_H queries to the random oracle and q_S queries to the signing oracle. Then there is an algorithm $\mathcal{B}$ that wins game 1.4.1 in time $\approx t$ with probability ϵ/q_H.*

Proof. We construct the reduction algorithm $\mathcal{B}$ to play game 1.3.1 with the adversary $\mathcal{A}$ as the challenger, and taking the role of the adversary in game 1.4.1. Without loss of generality, we assume that $\mathcal{A}$ runs $\mathsf{FDHVrfy}$ on their attempted forgery $(\mathsf{msg}^*, \sigma^*)$ and that they never repeat a signature or random oracle query.

To begin, $\mathcal{C}$ sends (f, r) to $\mathcal{B}$, who in turn chooses a random index $i^* \xleftarrow{\$} \{1, \ldots, q\}$ and sends the public key $pk = f$ to $\mathcal{A}$.

Whenever $\mathcal{A}$ submits a query $d_i \in \mathcal{D}$, $i \neq i^*$ to the random oracle, $\mathcal{B}$ chooses a uniformly random $y \xleftarrow{\$} \mathcal{R}$, and returns $f(y)$, recording $(d_i, y, f(y))$ in a table to answer future queries consistently.

When $\mathcal{A}$ submits d_{i^*} to the random oracle, $\mathcal{B}$ simply responds with r.

When $\mathcal{A}$ submits msg_i, their ith message query to the signing oracle, $\mathcal{B}$ checks the random oracle table to see if msg_i was ever submitted as a query to the random oracle. If it was not, $\mathcal{B}$ queries it, adding it to the table in the usual way. Then, if $\mathsf{msg}_i = d_{i^*}$, abort, and stop the reduction. Otherwise, return the y such that $f(y) = H(\mathsf{msg}_i)$, found in the table.

When $\mathcal{A}$ submits their forgery $(\mathsf{msg}^*, \sigma^*)$, hope that $\mathsf{msg}^* = d_{i^*}$. If not, abort. If so, then if this is a valid forgery and $f(\sigma^*) = H(\mathsf{msg}^*) = H(d_{i^*}) = r$. So σ^* is a preimage of r under f, and if we get to this point, then $\mathcal{B}$ has won game 1.4.1.

To analyse this reduction, note that we have assumed that $\mathcal{A}$ submits their forgery message msg^* as a hash query at some point. So when $\mathcal{B}$ chooses a random index for d_{i^*}, there is at least a $1/q_H$ chance of being correct. Then if $\mathcal{B}$ did guess correctly, they will be able to respond to all of the hash and sign queries correctly. Furthermore, since f is a permutation on $\mathcal{R}$, the method used to answer random oracle queries has the exact same distribution as a truly random oracle. So if $\mathcal{B}$ guesses the correct forgery query with probability $1/q_H$, then $\mathcal{A}$ succeeds with probability ϵ, and the chance of $\mathcal{B}$ winning game 1.4.1 is ϵ/q_H, as expected. □

1.5 Quantum Security Models

With the development of quantum information processing, and researchers becoming ever closer to building a universal quantum computer, many previous assumptions made by cryptographers have been shown to be invalid. The obvious false assumptions are of course the computational difficulty of problems such as integer factorization and the discrete logarithm problem. However, other assumptions are challenged in more subtle ways. For example, in security reductions, it is common to assume that the adversary interacts with the reduction algorithm in some classical way, for example by sending oracle queries as classical data that the reduction is allowed to both record and act upon. As we consider the possibility of quantum adversaries, there has been a large amount of research into the security of various schemes under various quantum assumptions. However, there has been relatively little discussion on the motivation behind these assumptions, and more importantly, the justification and necessity of various assumptions for practical post-quantum cryptography.

To begin with, we need to consider the threat model. The main threat model of post-quantum cryptography is the possibility of an adversary with a quantum computer attacking the classical systems and resources we use today, or in the near future. We need classical computers to resist quantum attacks, and the ability to use classical (but post-quantum) cryptography to defend against quantum attacks. It is certainly possible that when quantum computers and quantum communication become widely available, quantum resources will need to be protected and the threat model will need to be modified accordingly. However, the bulk of post-quantum cryptography is concerned with the much more imminent threat of quantum computers being used to attack classical resources.

When modelling an adversary, there are three main considerations:

- What the adversary's goal is.
- What computational resources the adversary has access to.
- What information the adversary has access to, and how they can access it.

The adversary's goal is specified by the security definition. For existential unforgeability, the adversary's goal is to find a valid message-signature pair that they have not seen before. Usually (but not always) the goal does not change when considering a quantum model of security. When we model security, the goal corresponds to what we want to *not* happen, like decrypting a ciphertext or forging a signature. Because the threat model is

concerned with protecting classical resources, the adversary's goal can generally be left unchanged when considering an adversary with a quantum computer.

The computational resources of the adversary naturally becomes the biggest change when considering a quantum adversary. Usually in security reductions, no assumptions are made about the adversary's computational resources, except that they run in polynomial time. If this is the case, then when moving to a quantum threat model, we properly take into consideration the adversary's resources by considering the difficulty of the underlying problem with respect to an adversary with a quantum computer. However this is not always the case. One technique sometimes used in security reduction is that of *rewinding*. In this technique, the internal state of the adversary is saved and copied, and the adversary is started twice from two possible initial points. But if the adversary's internal state is quantum, rather than classical, then by the no-cloning theorem, its internal state cannot be copied. This emphasizes the importance of being careful when using non-standard techniques in reductions, and what assumptions these techniques make on the adversary.

The information the adversary has access to and how it is accessed is one of the more subtle difficulties for quantum models. Consider the proof of the full domain hash signature scheme. The adversary has access to three types of information:

1. The public key, f.
2. Signing queries from the signing oracle.
3. Hash queries from the random oracle.

The adversary is granted this information for simple reasons. They are provided the public key because it is public, and therefore natural for an adversary to possess. They are granted access to a signing oracle because there are situations where an adversary can influence messages being signed. Since it is difficult to quantify how much influence an adversary might have, we allow the adversary to request signatures on arbitrary messages. Lastly, the adversary has access to evaluations of a hash function by way of a random oracle. They are provided this because in the real world the adversary is able to evaluate the hash function themselves.

Note that in the proof of theorem 1.4.1, all three of these information sources were provided in totally classical ways. A classical public key is provided, classical signing queries are returned, and access to the random oracle is provided by submitting a classical query and getting back a classical response. Each of these types of information, as well as how the information is accessed needs to be reevaluated to consider modelling a quantum adversary.

How the public key is provided does not need to be changed. As we are still considering the threat model of a quantum adversary attacking classical resources, the public key does not need to be changed or reevaluated. It is possible that there are cases when an adversary would want to prepare a superposition of public keys — for example if we are working in the multi-target model. However even in this model they would presumably be provided with the public keys classically, and then left to prepare such a state themselves.

Handling the signing queries classically is similarly motivated. A signing oracle is provided to the adversary to model their potential ability to coerce the legitimate signer to sign certain messages. But as long as the signing algorithm is implemented on a classical computer, there is no reasonable way to expect an adversary to be able to coerce a target into signing a superposition of messages. Authors have considered the possibility of security under this attack model, and shown that it is achievable [11]. However, as long as the threat model is a quantum adversary attacking classical resources, this is not necessary.

Queries to the random oracle is an entirely different matter. This behaviour is supposed to model the adversary's ability to implement and evaluate a hash function. If the adversary has a quantum computer then they can create a quantum circuit that allows for evaluation of the hash oracle in superposition. In our unitary model of quantum computing this means that they can create a unitary U_H that performs the mapping

$$U_H : |x\rangle|y\rangle \mapsto |x\rangle|y \oplus H(x)\rangle \tag{1.2}$$

on the basis states, and is extended linearly for superpositions of states. In our proof of full domain hash, however, the adversary could not possibly recreate the ability to implement such a mapping with how we have provided access to the random oracle in a purely classical way, without making an exponential number of queries to the random oracle.

Imagine that there is a quantum attack on the FDH scheme, and that this attack relies on the ability of the adversary to prepare the state $\frac{1}{2^{n/2}} \sum_x |x\rangle|H(x)\rangle$. In the real world a quantum adversary could prepare such a state, but when we modelled the adversary in our proof of FDH, we only allowed for classical queries, and so we have robbed the adversary of an ability we should reasonably expect them to have in the real world. Therefore when considering a quantum adversary, the random-oracle model is lacking in providing the adversary with all capabilities they may reasonably be expected to have. This is why the quantum random-oracle model is necessary to consider security against quantum adversaries.

To model a quantum adversary and determine which aspects of the interaction should be classical and which should be quantum, it is best to consider what these interactions

correspond to in the real world threat model. If they correspond to the adversary's communication with some classical external component of the protocol, they can be left classical. But if they correspond to the adversary's personal computational abilities, they must be made quantum.

1.6 QROM Basics and Challenges

As noted in the previous section, to model the abilities of a quantum adversary implementing a hash function on a quantum computer, we have to provide access to the random oracle H via access to the unitary mapping

$$U_H : |x\rangle|y\rangle \mapsto |x\rangle|y \oplus H(x)\rangle$$

and allow quantum access to this mapping. The problem of the classical random-oracle model when considering quantum adversaries was first noted by Boneh et al. [10], who also defined this way of generalizing the random-oracle model. However, the management and use of the quantum random oracle immediately results in several issues, originally outlined in the same paper. We repeat these here, as they provide an excellent list of challenges that are faced when considering the quantum random-oracle model. The first five items in the list were all described in [10], while the last item is a challenge not directly considered in [10].

1.6.1 Oracle Simulation

Classically, the simulation of the random oracle by a reduction algorithm is handled in a straightforward way. As queries to the random oracle are received, a table is built up of queries and responses. When a query is submitted that isn't in the table, a random output is generated as a response, and the query and the output are recorded in the table. By doing this, the simulation of the random oracle is entirely indistinguishable from a truly random oracle, and the reduction algorithm only needs to maintain a table that has size at most q.

In the quantum random-oracle model however, managing such a table is infeasible. The adversary can submit, as their first query, a superposition of all inputs, $\sum_x |x\rangle|0\rangle$, which requires the oracle to be defined for all possible inputs when the first query is made. A table is not suitable for this, as its length would be the size of the domain, requiring an exponential amount of work on the part of the reduction algorithm.

When this problem was noted in [10], the authors also proposed a solution in the form of quantum-accessible pseudorandom functions. Classically, a pseudorandom function (PRF) is a function f that can be implemented in an efficient way. However any distinguishing algorithm $\mathcal{D}$, when given black-box access either to f or a truly random oracle O (where the choice of providinf f or O was made uniformly at random), cannot guess which they were provided with probability non-negligibly higher than $1/2$.

A quantum-accessible PRF is simply a PRF where the distinguisher is instead given quantum access to O or f by way of the unitary mapping U_O or U_f. If $\mathcal{D}$'s chances of success are still negligibly smaller than $1/2$, then f can securely be used as a replacement for a quantum random oracle.

While quantum-accessible PRFs were suggested in [10], no PRFs were suggested as being secure against a quantum distinguisher. In a later work [47], Zhandry showed that some classes of PRFs known to be classically secure, such as those in [24, 33, 5] are also secure with respect to a quantum adversary, as long as a quantumized version of the underlying assumptions holds.

Quantum-accessible PRFs are an efficient and flexible replacement for a quantum random oracle. However they do introduce an additional computational assumption. Having to consider this additional assumption is sometimes undesirable. The classical random-oracle model needs no such assumption, as queries can be answered as they are made in a uniform and independent way. This allows a reduction algorithm to act in a way that is completely indistinguishable from a random oracle (even though the entire random oracle is not constructed).

In another paper [46], Zhandry proposes a different way to simulate a quantum random oracle, using k-wise independent functions.

Definition 1.6.1. *A family of k-wise independent functions is a set $\mathcal{F}$ of functions $f : \mathcal{D} \to \mathcal{R}$ such that if $d_1, \ldots, d_k$ are any k different elements of $\mathcal{D}$ and $r_1, \ldots, r_k$ are any k elements of $\mathcal{R}$ (possible with repeats), then*

$$Pr_{f \xleftarrow{\$} \mathcal{F}}[f(d_1) = r_1 \wedge f(d_2) = r_2 \wedge \cdots \wedge f(d_k) = r_k] = \frac{1}{|\mathcal{R}|^k}. \tag{1.3}$$

Intuitively, a k-wise independent function is a function that appears perfectly uniform and independent if you look at no more than k input/output pairs. The following result establishes how these functions may be used to replace a quantum random oracle.

Theorem 1.6.1 ([46])**.** *Let $\mathcal{A}$ be a quantum algorithm outputting some classical state z, that makes q quantum queries to a random oracle $\mathcal{O} : \mathcal{D} \to \mathcal{R}$, drawn uniformly from the*

set of all such functions. If $\mathcal{F}$ is a family of $2q$-wise independent functions $f : \mathcal{D} \to \mathcal{R}$, then

$$Pr[\mathcal{A}^{\mathcal{O}} \to z] = Pr_{f \xleftarrow{\$} \mathcal{F}}[\mathcal{A}^{f} \to z]. \tag{1.4}$$

That is, the output distribution of any quantum algorithm that makes q queries is identical when given a $2q$-wise independent function instead of a true random oracle. This powerful result allows us to devise proofs in the quantum random-oracle model without additional assumptions about PRFs. However, it is somewhat less flexible than quantum-accessible PRFs, as it requires a priori knowledge (or at least an upper bound) on the number of queries to the random oracle made by the adversary.

1.6.2 Query Lookup

When a classical adversary makes a query to a random oracle, the reduction algorithm is able to see what the query is. Having knowledge of what queries the adversary is interested in can often allow a reduction algorithm to succeed. But in the quantum random-oracle model, the queries come in the form of superpositions. To extract (classical) information about a superposition, a measurement needs to be performed. But a measurement on an *arbitrary* superposition can never give complete information about that superpositioned query. Furthermore, a measurement necessarily destroys some information, meaning that if the reduction algorithm wants some information about a query being made, complications could arise in both getting information about the query and responding with a consistent output for the oracle.

It seems unlikely that any satisfying method to gain complete knowledge of an adversary's queries is possible, as the no-cloning theorem implies that it is impossible to create a copy of an arbitrary query to the oracle. As complete information about a query would allow for the creation of a copy, the no-cloning theorem suggests that this is impossible.

However there are other ways to get around this impossibility. For example, this problem arises when trying to establish a quantum random-oracle proof of the Fujisaki-Okamoto transform [22], where a fundamental step in the reduction algorithm is to look up the queries the adversary makes in order to be able to respond to decryption queries.

Recall that the Fujisaki-Okamoto transform is a hybrid encryption scheme that uses a public key system and a symmetric key system. It uses two random oracles, H_1 and H_2, which are entirely independent. Then the transformation, with security parameter n, works as follows:

1. Choose a uniformly random bitstring δ of length n.
2. Encrypt the message with the symmetric key scheme, with the key being $H_1(\delta)$, to obtain ciphertext c_1.
3. Encrypt δ under the public key scheme, with the randomness used by the scheme being $H_2(\delta||c_1)$ to obtain ciphertext c_2.
4. Send (c_1, c_2).

In [40], the authors proposed a modification to the transformation in order to establish a quantum random-oracle model reduction. In their modification, before outputting the ciphertext (c_1, c_2), a third (independent) random oracle H_3 is used to hash the value δ, and the outputted ciphertext is $(c_1, c_2, H_3(\delta))$.

The reduction algorithm needs to have a way to obtain δ from the ciphertext. Classically, this is done by looking at the queries made to H_2, and looking for any one of the form $(\delta'||c_1)$, for some δ'. In the quantum random-oracle model, by appending $H_3(\delta)$ to the ciphertext, the reduction algorithm can obtain δ by using an invertible function for H_3. In our discussion of implementing a quantum random oracle, we noted that a $2q$-wise independent function is indistinguishable from a random oracle. There are constructions of k-wise independent functions that can be inverted in polynomial-time [7]. By using such a construction, the reduction algorithm is able to recover δ and complete the reduction.

1.6.3 Rewinding

The concept of rewinding was discussed in section 1.5. It is a technique that is not unique to the random-oracle model, but often appears in random-oracle model proofs. In it, an adversary's internal state is saved, and then restarted from that point with some modification to the interaction that occurs between the reduction algorithm and the adversary. The no-cloning theorem (which follows directly from the fact that closed-system transformations must be unitary) states that arbitrary quantum states can never be copied. Rewinding a quantum adversary after extracting some information is therefore impossible to do in a generic way.

This problem is in some sense independent from the random-oracle model, in that if a proof requires rewinding a quantum adversary, this may be impossible whether using the random-oracle model or not. However because this technique is often used in the random-oracle model, it is often associated with it.

Rewinding is used in the proof of the Fiat-Shamir transform, which is commonly used to construct signature schemes. In [19], the authors established a negative result for the security of the Fiat-Shamir transform in the quantum random-oracle model. While their results did not completely rule out security of the transform in all contexts, it did suggest that its security will be difficult to establish. However, there have also been results in other contexts that were able to show how a quantum adversary could be rewound [45].

1.6.4 Reprogramming

As discussed, a necessary consequence of allowing superpositions of queries to the random oracle is that the oracle must be defined for all possible inputs when the first query occurs. Thus after the first query, it is possible for the adversary to have information about any one of the inputs to the oracle.

A common technique in the random oracle is that of oracle reprogramming. In this technique, after some amount of interaction with the adversary, the reduction algorithm chooses an input to the random oracle, say a domain point x. Their choice of this x may be arbitrary, but due to being able to perform query lookups, the reduction algorithm can ensure that the adversary has not queried x before, and thus has no information about $H(x)$. Then the reduction algorithm can choose $H(x)$ to be whatever they would like, as long as it has a distribution indistinguishable from uniform.

Due to similar reasons as query lookups, it is difficult to justify this technique in the quantum random-oracle model. It is no longer clear which queries the adversary has information on, and which they don't, or how one can quantify how much information the adversary has on a query.

One of the first papers on quantum computing and its limitations included a result that would form a basis for discussing reprogramming in the quantum random oracle [8]. The authors first defined the concept of query magnitude.

Definition 1.6.2 (Query Magnitude)**.** *Let $|\psi\rangle$ be a query to a quantum random oracle. Write $|\psi\rangle$ as*

$$|\psi\rangle = \sum_{x,y} \alpha_{x,y}|x\rangle|y\rangle. \tag{1.5}$$

Then the query magnitude of x' in $|\psi\rangle$ is

$$q_{x'}(|\psi\rangle) = \sum_{y} |\alpha_{x',y}|^2. \tag{1.6}$$

Note that according to our rules of measurement, this value corresponds to the probability of obtaining x' when measuring a query $|\psi\rangle$ in the computational basis.

Theorem 1.6.2 (Theorem 3.3 in [8]). *Let $\mathcal{A}$ be a quantum algorithm that makes queries $|\psi_1\rangle, \ldots, |\psi_q\rangle$ to a random oracle H, resulting in a final state $|\psi\rangle$. Let $\epsilon > 0$. Define a new random oracle H', which is the same as H except on some subset $D \subseteq \mathcal{D}$. Let $|\psi'\rangle$ be the state that results from running $\mathcal{A}$ on the same input but with H' rather than H. Then if*

$$\sum_{i=1}^{q} \sum_{d \in D} q_d(|\psi_i\rangle) \leq \frac{\epsilon^2}{q}, \tag{1.7}$$

we have that

$$||\psi\rangle - |\psi'\rangle| \leq \epsilon. \tag{1.8}$$

Theorem 1.6.2 will form the basis of our discussion of reprogramming in various contexts in chapters 3 and 4.

Results related to reprogramming have also appeared in the context of revocable timed-release encryption and position verification [43] [42]. These results similarly bound the distance between the output distribution of a quantum algorithm when an oracle is modified partway through interaction with a reduction algorithm.

1.6.5 Challenge Injection

Many reductions succeed in the random-oracle model by injecting a challenge into one of the responses to the random oracle. This was seen in the proof of FDH. A random query was selected, and rather than responding in the usual way, the reduction algorithm responded with the element r that was provided by the challenger.

It is not immediately clear how to apply this technique to the quantum random-oracle model. Certainly a random query can't simply be responded to by returning the classical element r. If a superposition query had already been made, a classical response would be noticed as not acting in accordance to a random oracle.

Most obvious approaches to overcome this challenge would not work. For example, we might try and embed the challenge y into a single point in the random oracle, that is choose a domain point $d \in \mathcal{D}$ and set $H(d) = y$. But this offers no guarantee that the adversary will choose d to be their forgery, and so the probability of the reduction algorithm being able to invert the function is exponentially small, even if the adversary's chances of success are quite large.

A second approach might be to try something similar to how the classical case is handled. That is, to choose a random index and simply reply with y. But in this case we are not acting according to a 'proper' quantum random oracle. An adversary making a superposition query could reasonably be expected to notice that the unitary applied to their query wasn't very random, and so their chances of success could be altered, and we have no way to guarantee that their forgery would correspond to y.

One possible solution is to choose a random subset D of the domain $\mathcal{D}$ and define the oracle H so that for any $d \in D$, $H(d) = y$, the challenge point. The question then is if it is possible to choose D in such a way that it is large enough so that we can reasonably hope for the forgery to be associated with y, but not so large that the adversary notices that our oracle isn't a true random oracle.

In [46], Mark Zhandry showed that this was possible, defining a construction called *semi-constant distributions*.

Definition 1.6.3 (Semi-constant distribution, from [46])**.** *The semi-constant distribution $\mathsf{SC}_{\lambda,y}$ is a distribution on mappings from a domain $\mathcal{D}$ to a range $\mathcal{R}$. It is parameterized by a value $\lambda \in [0,1]$ and an element $y \in \mathcal{R}$. The distribution is defined by how it is sampled. For each $d \in \mathcal{D}$, with probability λ set $H(d) = y$. Otherwise set it to a uniformly random element of $\mathcal{R}$.*

Then Zhandry proved the following theorem:

Theorem 1.6.3 (Corollary 4.3 in [46])**.** *If y is a uniformly random element of $\mathcal{R}$, then the distribution of any quantum algorithm that makes q queries to a random oracle has distance at most $\frac{8}{3}q^4\lambda^2$ from the distribution generated when $\mathsf{SC}_{\lambda,y}$ is used instead.*

In section 1.7, we will use this theorem to show that FDH is secure in the quantum random-oracle model, assuming the existence of a quantum-secure trapdoored permutation.

1.6.6 Query Categorization

Query categorization is a technique related to, but distinct from, the concept of query lookup. Often in the specification of a scheme, multiple variables are hashed together, usually by concatenation. For example, the scheme may specify the hashing of two 128-bit strings, where each string represents something different. Classically, we can then categorize the queries made by the adversary by grouping together queries that are the same on one of the registers. This can often be helpful in analysis, as the probability of

certain events may be conditioned on the number of queries made with respect to one register being constant.

In the quantum random-oracle model, this categorization cannot be so easily done. Each query is potentially a superposition of all possible classical queries, and so we may no longer be able to cleanly categorize the queries the adversary makes. This can significantly complicate the analysis required to determine the probability of certain events.

1.7 FDH in the QROM

We will now show how the generic FDH scheme using trapdoored permutations has a security reduction in the quantum random-oracle model. Note that such a security reduction is only meaningful if there is a trapdoored permutation that cannot be inverted by a quantum computer. Trapdoored permutations based on problems such as the RSA problem are of course useless against a quantum computer. To date, there are no known trapdoored permutations secure against a quantum computer. Nonetheless, this proof at least shows that the basic idea of the scheme is still sound. As well, there are trapdoored functions that are not permutations [23], but can be used to construct essentially the same scheme.

Theorem 1.7.1 ([46] Existential Unforgeability of Generic FDH in the QROM)**.** *Let $\mathcal{A}$ be a quantum adversary that wins game 1.3.1 with signature scheme 1.4.1 in time t with probability ϵ, having made q_H quantum queries to the random oracle and q_S classical queries to the signing oracle. Then there is an algorithm $\mathcal{B}$ that wins game 1.4.1 in time $\approx t$ with probability $poly(\epsilon)/poly(q_H, q_S)$.*

Proof. We will define an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine in order to try and win game 1.4.1. First, the challenger $\mathcal{C}$ gives $\mathcal{B}$ a uniformly random $y \in \mathcal{R}$. $\mathcal{B}$ chooses a parameter $\lambda \in [0, 1]$ (how it is chosen will be discussed later).

First, $\mathcal{B}$ constructs a quantum random oracle $\mathcal{O}_1 : \mathcal{D} \to \{0, 1\}$ such that for each $d \in \mathcal{D}$, $Pr[\mathcal{O}_1(d) = 1] = \lambda$, and these probabilities are independent. They also construct a quantum random oracle $\mathcal{O}_2 : \mathcal{D} \to \mathcal{R}$. These oracles can be instantiated with either quantum-accessible PRFs or $2q$-wise independent functions. Then, $\mathcal{B}$ can define the random oracle $H : \mathcal{D} \to \mathcal{R}$ that $\mathcal{A}$ will be given access to as

$$H(d) := \begin{cases} y & \text{if } \mathcal{O}_1(d) = 1 \\ f(\mathcal{O}_2(d)) & \text{if } \mathcal{O}_1(d) = 0 \end{cases}. \tag{1.9}$$

Note that since f is a permutation, and O_2 is a random oracle mapping onto $\mathcal{R}$, the distribution of $f \circ O_2$ is the same as that of O_2 (taken over the randomness of O_2). Then H has distribution $\mathsf{SC}_{\lambda,y}$, and so by Theorem 1.6.3, the distance between the adversary's output distribution is at most $\frac{8}{3}q_H^4\lambda^2$.

Then when $\mathcal{A}$ submits a signing query, $d_i \in \mathcal{D}$, $\mathcal{B}$ does the following:

1. Check if $O_1(d_i) = 0$. If not, abort.
2. Output $O_2(d_i)$ as a signature for d_i.

When the adversary submits a forgery (d^*, σ^*), hope that $O_1(d^*) = 1$. If so, then $H(d^*) = y$ and so if the forgery is valid then $f(\sigma^*) = y$.

We now analyse the probability with which $\mathcal{B}$ successfully finds the preimage of y. Note that assuming that the distribution of $\mathcal{A}$'s signing queries are independent from O_1, the probability that all of their signing queries satisfy $O_1(d_i) = 0$ is at least $1 - q_s\lambda$. Then assuming that the distribution of the forgery is independent from O_1, the probability that the forgery satisfies $O_1(d^*) = 1$ is λ. So ignoring that H doesn't behave like a normal hash function, we have that the probability of success is at least $\lambda(1 - q_s\lambda)$.

While we have an upper bound on the change in the adversary's output distribution, this is complicated by the fact that the adversary's interactions with the random oracle and signing oracle are affected by the altered oracle. Nonetheless, as $\mathcal{A}$ has only q_h and q_s signing queries, the success probability of $\mathcal{B}$ can be shown to be

$$\frac{3\epsilon^2}{4(2q_h + 3q_s + 2)^4}. \tag{1.10}$$

For simplicity, we omit how this bound is fully established. The full proof can be found in appendix E of [46]. □

1.8 Thesis Roadmap

In chapter 2 we will study LMS, a hash-based signature scheme. Jonathan Katz's classical security proof of LMS [28] upper bounded the success probability of any classical adversary by categorizing the queries made to the oracle and considering the probability of these events happening. As we can no longer categorize queries being made in such a way, we need to reformulate the events and find different ways to upper bound the probabilities.

In chapter 3 we examine a lattice-based scheme, TESLA. A security reduction for this scheme reduces to the LWE problem. However, the security reduction relies on the ability to reprogram the quantum random oracle. We develop some theory and an approach that will allow us to justify reprogramming the random oracle in this particular context.

In chapter 4 we look at a transformation on signature schemes that can generically upgrade its security from existential unforgeability to something called strong unforgeability. To establish a security reduction for the scheme, we need to again reprogram the random oracle. However the results from chapter 3 do not apply to this context, so we have to establish the soundness of reprogramming in this context.

Chapter 2

LMS in the QROM

In a recent paper [28], Jonathan Katz analyzed the security of the Leighton-Micali hash-based signature scheme, which we will refer to as LMS. LMS has been proposed for standardization in [31].

Katz's analysis used the random oracle to establish the security of LMS. However, as the random-oracle model is insufficient for establishing the security of a protocol against an adversary with access to a quantum computer, we must move to the quantum random oracle, as discussed in section 1.5.

In this section, we reformulate and update Katz's random-oracle model proof of security for LMS to the quantum random-oracle model. As LMS is a hash-based scheme, this is particularly important as it is a strong candidate for post-quantum standardization. We also discuss some of the difficulties that need to be overcome in order to establish this proof in the QROM.

2.1 Scheme Description

2.1.1 One-Time Scheme

The basic component of the full scheme is the one-time (OT) signature scheme, also known as the Winternitz signature scheme. This scheme consists of OT key generation, signing, and verifying algorithms. It uses, as a basic component, a hash function $H : \{0,1\}^* \to \{0,1\}^n$, where n is the security parameter.

The parameters are:

- n, the security parameter.
- w, the Winternitz parameter, which is a small divisor of n (generally less than or equal to eight).

These parameters define the following values:

- $E = 2^w - 1$
- $u_1 = n/w$
- $u_2 = \lceil \lfloor \log_2 (u_1 \cdot E) + 1 \rfloor / w \rceil$
- $p = u_1 + u_2$.

For our purposes, string concatenation is denoted by $||$.

We can parse a string of n bits as the concatenation of u_1 strings, each w bits long and representing an integer from 0 to E. This allows us to define the $\mathsf{checksum} : (\{0,1\}^w)^{u_1} \to (\{0,1\}^w)^{u_2}$ function as

$$\mathsf{checksum}(h_1, \ldots, h_{u_1}) = \sum_{i=1}^{u_1} (E - h_i). \tag{2.1}$$

We can then see that u_2 was chosen so that $w \cdot u_2$ is the maximum bit length of the result of the $\mathsf{checksum}$ function.

We define the function F as a repeated application of H, with each application also adding some additional information, such as the number of times H has been applied. We also include $s = I||Q||i$, a string consisting of an identifying string I for the owner of the public key, a string Q indicating which instance of the scheme is being used, and a number i indicating which chain of hashes we are referring to. This information is used in the multi-user and multi-instance analysis of the scheme. For $0 \leq b \leq f \leq E$, define

$$F_s(x; b, f) = \begin{cases} x & \text{if } b = f \\ F_s(H(x||s||b||00); b+1, f) & \text{if } b < f. \end{cases}$$

Signature Scheme 2.1.1. OT-LMS

Algorithm 4 OTLMSKeyGen

Input: Security Parameter 1^n, Winternitz parameter w.
Output: Public key pk, secret key sk.

1: Choose p values $x_1^0, x_2^0, \ldots, x_p^0 \in \{0,1\}^n$, uniformly at random.
2: For $i = 1$ to p, let $s = I||Q||i$ and compute $x_i^E = F_s(x_i^0; 0, E)$.
3: Let $pk = H(x_1^E||x_2^E||\ldots||x_p^E||01)$.
4: The one-time public key is pk, and the secret key is $sk = (x_1^0, \ldots, x_p^0)$.

Algorithm 5 OTLMSSign

Input: Message $M \in \{0,1\}^*$, secret key sk.
Output: Signature σ.

1: Choose a uniformly random $r \in \{0,1\}^n$.
2: Compute $h = H(M||r||I||Q||02)$ and $c = \mathsf{checksum}(h)$. Set $v := h||c$ and parse v as p w-bit integers in $\{0, \ldots, E\}$, $v = (v_1, v_2, \ldots, v_p)$.
3: For $i = 1$ to p, let $s = I||Q||i$ and compute $\sigma_i = F_s(x_i^0; 0, v_i)$.
4: Output signature $\sigma = (r, \sigma_1, \ldots, \sigma_p)$.

Algorithm 6 OTLMSVrfy

Input: Message $M \in \{0,1\}^*$, public key pk (if being used as a standalone scheme), signature $\sigma = (r, \sigma_1, \ldots, \sigma_p)$.
Output: accept or reject if being used as a standalone signature scheme, value pk' if being used as part of the full LMS scheme.

1: Compute $h' = H(M||r||I||Q||02)$ and $c' = \mathsf{checksum}(h)$. Set $v' = h'||c'$, and parse v' as p w-bit integers in $\{0, \ldots, E\}$, $v' = (v'_1, v'_2 \ldots, v'_p)$.
2: For $i = 1$ to p, let $s = I||Q||i$ and compute ${x'}_i^E = F_s(\sigma_i; v'_i, E)$.
3: Let $pk' = H({x'}_1^E||{x'}_2^E||\ldots {x'}_p^E||01)$. If the scheme is used as part of the full scheme, output pk'. If it is being used as a standalone signature scheme, output 'accept' if and only if $pk' = pk$.

The correctness property can verified by inspection.

2.1.2 Full Scheme

In the full scheme, we combine the one-time scheme as a subroutine with a Merkle tree construction in order to have a full (stateful) signature scheme.

In addition to the parameters for the one-time scheme, we have the parameter H. We will create 2^H separate instances of the one-time scheme.

Signature Scheme 2.1.2. LMS Signature Scheme

Algorithm 7 LMSKeyGen

Input: Security Parameter 1^n, Winternitz parameter w, Merkle tree height 1^H
Output: Public key pk, secret key sk

1: For $i = 1$ to 2^H, obtain $(pk_i, sk_i) \leftarrow \mathsf{OTLMSKeyGen}(1^n, w)$.
2: For $i = 1$ to 2^H, compute $y_i^0 := H(pk_i||I||i||03)$.
3: For $j = 1$ to H:
    1. For $k = 1$ to 2^{H-j}, compute $y_k^j := H(y_{2k-1}^{j-1}||y_{2k}^{j-1}||k||j||04)$.
4: Output $pk = y_1^H$ as the public key, and $sk = (sk_1, \ldots, sk_{2^H})$ as the secret key.
5: Initialize $Q = 0$.

Algorithm 8 LMSSign

Input: Message $M \in \{0,1\}^*$, secret key sk
Output: Signature σ

1: Increment Q by 1. If $Q = 2^H + 1$, **STOP**; all signatures have been used.
2: Obtain $\sigma' \leftarrow \mathsf{OTLMSSign}(M, sk_Q)$.
3: Let $c \leftarrow Q$. Update $\sigma \leftarrow \sigma'||Q$.
4: For $j = 0$ to $H - 1$:
    1. If c is even, let $\sigma \leftarrow \sigma||y_{c-1}^j$ and $c \leftarrow c/2$.
    2. If c is odd, let $\sigma \leftarrow \sigma||y_{c+1}^j$ and $c \leftarrow (c+1)/2$.
5: Output σ.

Algorithm 9 LMSVrfy

Input: Message $M \in \{0,1\}^*$, public key pk, signature $\sigma = \sigma'||Q||y^0||y^1||\dots||y^{H-1}$
Output: accept or reject

1: Obtain $pk' \leftarrow \mathsf{OTLMSVrfy}(M, \sigma')$.
2: Compute $y = H(pk'||I||Q||03)$.
3: Let $c \leftarrow Q$.
4: For $j = 0$ to $H - 1$:

1. If c is even, let $y \leftarrow H(y^j||y||c/2||j+1||04)$ and $c \leftarrow c/2$.
2. If c is odd, let $y \leftarrow H(y||y^j||(c+1)/2||j+1||04)$ and $c \leftarrow (c+1)/2$.

5: Output accept if and only if $y = pk$. Output reject otherwise.

2.2 LMS in the (Quantum) Random-Oracle Model

Katz's classical proof of the security of LM-Winternitz takes place in the random-oracle model. In this proof, we are running an experiment with an adversary $\mathcal{A}$, who is attacking the existential unforgeability of the scheme. Whenever this adversary wishes to evaluate the hash function H on a point x, they must instead query us for the evaluation, and we respond in a way that is (computationally) indistinguishable from truly random. Katz shows that for any adversary that makes q queries, the probability that $\mathcal{A}$ can win a game of existential-unforgeability is at most $3q/2^n$. He establishes this by showing that for the adversary to win a game, one of a series of events must occur. Then by upper bounding the probability of these events happening, the upper bound follows.

In the quantum random-oracle model however, new issues arise. Katz's events are defined by considering the queries that the adversary makes and the responses they receive. However in the quantum random-oracle model, the queries the adversary makes no longer need be classical, and so the definition of these events no longer quite makes sense. Instead the events must be defined by considering what classical information the adversary is able to *find*, rather than just what they *query*. Classically, the information the adversary has about an oracle is entirely specified by the queries being made. Quantumly, the information an adversary has about an oracle is much more challenging to classify.

Note that there is a slight issue in that the classical oracle is defined from $\{0,1\}^*$ to $\{0,1\}^n$, that is, takes in variable length input. In our circuit model of computation, we must specify the length of input to the oracle in advance. This issue is resolved by considering an upper bound on the length of binary strings that the adversary needs to

query (such an upper bound must exist, and as long as the adversary runs in polynomial time, this length is poly-log in the security parameter n).

2.2.1 Second-preimage Resistance with Non-uniform First-preimage

The concept of second-preimage resistance in the random-oracle model is an essential one to the analysis of LMS. The usual game of second-preimage resistance (in the random-oracle model) between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is defined as:

Game 2.2.1 (Uniform Second-preimage Resistance)**.**

1. $\mathcal{C}$ chooses a random function $H : \{0,1\}^* \to \{0,1\}^n$ from all possible mappings, and provides oracle access to H to $\mathcal{A}$.
2. $\mathcal{A}$ makes q_1 queries to H.
3. $\mathcal{C}$ selects an element $x \in \{0,1\}^*$ uniformly at random, and sends x to $\mathcal{A}$.
4. $\mathcal{A}$ makes q_2 queries to H, and then submits an $x' \in \{0,1\}^*$, $x' \neq x$.

We define $q = q_1 + q_2$. If $H(x') = H(x)$, we say that the adversary has won. Classically, upper bounding the success probability of $\mathcal{A}$ is trivial. As H is a random oracle, the adversay's only choice is to submit q different $x' \neq x$ and hope that one of them maps to $H(x)$. This happens with probability $\leq \frac{q}{2^n}$.

If $\mathcal{A}$ has quantum access to H, they gain an advantage, as they are now able to perform Grover's search to try and find such an x'. However by results on upper bounding quantum search probability, such as those in [13], we have that the success probability is at most $2q/\sqrt{2^n}$.

However for our purposes, we need a somewhat different game. In particular, we need to consider what happens when the first-preimage, x, is chosen from a non-uniform distribution.

Game 2.2.2 (Non-uniform Second-Preimage Resistance)**.**

1. $\mathcal{C}$ chooses a random function $H : \{0,1\}^* \to \{0,1\}^n$ from all possible mappings, and provides $\mathcal{A}$ with oracle access to H.
2. $\mathcal{A}$ makes q_1 queries to H.

3. $\mathcal{C}$ selects an element $x \in \{0,1\}^*$ according to some distribution D on $\{0,1\}^*$, and sends x to $\mathcal{A}$.

4. $\mathcal{A}$ makes q_2 queries to H, and then submits an $x' \in \{0,1\}^*$, $x' \neq x$.

As before, $\mathcal{A}$ wins this game if $H(x) = H(x')$.

We would like this game to be as difficult as the uniform second-preimage game, but this is not true for all possible distributions D. In particular, D can depend on H. We can imagine a challenger $\mathcal{C}$ who chooses the first-preimage x by querying H until they find an x such that $H(x) = H(0)$ and giving this x as the first-preimage. Then $\mathcal{A}$ can easily win the game by simply returning $x' = 0$. Although in this case D is exponentially (in n) difficult to actually sample from, this does establish that D can matter in general. However D was constructed to explicitly depend on H. Intuitively, it should be the case that if D is independent of H, it should not matter if D were uniform or not.

We now establish that this intuition is perfectly sensible by a reduction from one game to another. This reduction is valid whether $\mathcal{A}$ has classical or quantum access to H. However this reduction is explicitly in the random-oracle model and does not carry over to the case where H is a real hash function.

Theorem 2.2.1. *Let $\mathcal{A}$ be an adversary capable of winning the game of non-uniform second-preimage resistance with probability p in q queries to the random oracle, with the distribution being D, which is independent of the random oracle. Then there is an algorithm $\mathcal{B}$, which samples from D once, and is capable of winning the game of uniform second-preimage resistance with probability p in q queries.*

Proof. We construct $\mathcal{B}$, using $\mathcal{A}$ as a blackbox to solve the problem. $\mathcal{B}$ is attempting to win the game of uniform second-preimage resistance, so they are provided with a uniform x and access to a random oracle H. Then $\mathcal{B}$ first samples a $d \leftarrow D$, $d \in \{0,1\}^*$. Then $\mathcal{B}$ constructs a random oracle, H', as follows:

$$H'(m) = \begin{cases} H(x) & \text{if } m = d \\ H(d) & \text{if } m = x \\ H(m) & \text{if } m \neq x, d. \end{cases} \tag{2.2}$$

Intuitively, H' is constructed from H by swapping the outputs of x and d. Then note that since x was sampled uniformly, and d was sampled from a distribution entirely independent from H, H' still has the distribution of a random oracle. Also note that for

each query $\mathcal{A}$ makes to H', $\mathcal{B}$ can answer it by making precisely one query to H, and this is true for classical or quantum queries.

$\mathcal{B}$ presents d to $\mathcal{A}$ and provides access to H'. Eventually, $\mathcal{A}$ outputs a $d' \neq d$. With probability p, $H'(d') = H'(d) = H(x)$. On all inputs other than x and d, $H'(m) = H(m)$, and so as long as $d' \neq x$, we have that d' is a second-preimage to x. If $d' = x$ then d is a second preimage to x. □

2.2.2 Second-preimage Resistance with Adversary Prefixes

Also important to the analysis of LMS is a slight modification of game 2.2.2, where the adversary is able to specify a prefix of the element whose second preimage they are then searching for. We again define this in terms of a game.

Game 2.2.3 (Second Pre-image Resistance with Adversary Prefixes)**.**

1. $\mathcal{C}$ chooses a random function $H : \{0,1\}^* \to \{0,1\}^n$ from all possible mappings, and provides access to H to $\mathcal{A}$.
2. $\mathcal{A}$ makes q_1 queries to H, and then submits a prefix M'.
3. $\mathcal{C}$ selects some suffix $r' \leftarrow \{0,1\}^n$ uniformly at random, and sends r' to $\mathcal{A}$.
4. $\mathcal{A}$ makes q_2 queries to H, and then submits an $M^*, r^* \in \{0,1\}^* \times \{0,1\}^n$, with $M' \neq M^*$.

Again, we say that the adversary has won if $H(M^*||r^*) = H(M'||r')$. Note that this game is conceptually very similar to the game of non-uniform second-preimage resistance, as the adversary may choose the distribution of the first portion of the first preimage according to any distribution. However, the adversary is allowed to choose this prefix in an arbitrary way, which can depend on the random oracle, so our result on the distribution not being affected does not apply, and it is possible that some advantage can be gained by the adversary over second-preimage resistance.

Classically, it is not difficult to show that an adversary does not obtain much of an advantage. In Katz's paper [28], he tackles this issue through the use of random oracle reprogramming. Specifically, he considers the challenger that, when the adversary submits their prefix M', modifies H to H' so that $H(M'||r') = h'$, where r' and h' are uniformly random n bit strings that were chosen in step 1 of the game. The adversary will only notice

that $\mathcal{C}$ isn't playing by the 'real' rules of the game if they had previously queried $M'||r'$, and since r' is not disclosed to the adversary in advance, this happens with probability $\leq \frac{q_1}{2^n}$. Then the probability that an adversary queries a different $M^*||r^*$ such that $H(M^*||r^*) = h'$ is simply $q/2^n$. So we upper bound the probability that the adversary wins this game by $q_1/2^n + q/2^n \leq 2q/2^n$.

This proof is much more difficult in the quantum setting however. In Katz's proof, an essential step was to reprogram the oracle to reduce to something that more closely resembled second-preimage resistance. Since the adversary has a limited number of queries, they don't have any information about what is reprogrammed with high probability. In the quantum case however, this is much more challenging. Since the adversary can make a quantum superposition of queries, an adversary can make a query giving them some information about the entire oracle. While there are existing results [21, 43, 42] discussing reprogramming in the quantum random-oracle model, we instead avoid completely the issue of reprogramming and show a reduction without it.

Without relying on reprogramming, it is not clear how to proceed with a proof. We would like to establish that the difficulty of this game is roughly the same as that of second-preimage resistance. But as the adversary is able to control the message being chosen, we need to be able to quantify the adversary's ability in finding oracles $H(M||\cdot)$ that can make their job easier in some sense. As a quantum computer is capable of seeing the 'forest rather than the trees' in a certain sense, it is difficult to guarantee that any quantum adversary cannot force an oracle to have certain properties.

However, any type of second-preimage always has the property the it forms a collision in the oracle H. So we can always upper bound an adversary's ability to win game 2.2.3 by their ability to find a collision. Classically, this is somewhat unsatisfying. The probability a classical adversary finds a second-preimage in q queries is $q/2^n$, while their ability to find a collision is roughly $q/2^{n/2}$. For a quantum adversary, the ability of a quantum adversary to find a second-preimage is roughly $q/2^{n/2}$, and the ability to find a collision is roughly $q/2^{n/3}$. While it would still be preferable to have the stronger bound, the gap between the two is much smaller for quantum adversaries. It has even been argued that this gap is very minor. In fact in [9], Bernstein argues that because the best known quantum collision-finding algorithms are not parallelizable, the bounds provided by a classical adversary may better reflect the cost of an optimal attack.

This tells us that

$$\Pr[\mathcal{A} \text{ wins game } 2.2.3] \leq \Pr[\mathcal{A} \text{ finds a collision in } H] \leq q/2^{n/3}. \tag{2.3}$$

2.2.3 Random Oracle Composition

In the description of LMS, and occasionally in other constructions, a function is defined by a composition of independent random oracles. It would be convenient for this function to itself be a random oracle, or at least have certain properties of a random oracle, from the perspective of both classical and quantum adversaries. However, this is not quite the case.

Let $\mathcal{O}_1, \ldots, \mathcal{O}_E$ be independent random oracles mapping n-bit strings to n-bit strings. Consider the oracle $\mathcal{O} = \mathcal{O}_E \circ \mathcal{O}_{E-1} \circ \cdots \circ \mathcal{O}_1$, $\mathcal{O} : \{0,1\}^n \to \{0,1\}^n$. We want to consider properties of the combined oracle $\mathcal{O}$ with respect to standard properties like preimage resistance.

Lemma 2.2.1. *Let O be a random mapping from a domain $\mathcal{D}$ of size N to a codomain $\mathcal{R}$ of size M. Then the expected size of the image of $\mathcal{D}$ under O is*

$$M\left(1 - \left(1 - \frac{1}{M}\right)^N\right).$$

Proof. Let $\mathcal{R} = \{1, \ldots, M\}$. For each $1 \leq i \leq M$, let X_i be a binary random variable where X_i is 1 if there is an $x \in \mathcal{D}$ such that $O(x) = i$, and 0 otherwise. It is not hard to see that $E[X_i] = 1 - (\frac{M-1}{M})^N$. Then the expected number of elements in the codomain that are hit is $E[X_1 + X_2 + \cdots + X_M] = E[X_1] + E[X_2] + \cdots + E[X_M]$, from which the result follows. □

Writing $N = \alpha \cdot M$, for sufficiently large N and M, Lemma 2.2.1 tells us that the fraction of the codomain that is hit is roughly

$$\left(1 - \frac{1}{e^\alpha}\right)$$

where $e \approx 2.71828$ is Euler's constant. So when k oracles, each of which maps to a codomain of size 2^n, are composed, the overall oracle maps to an image that has size roughly

$$2^n \cdot \left(1 - \left(\frac{1}{e}\right)^{1-(1/e)^{1-(1/e)^{\cdots}}} k\right). \tag{2.4}$$

For example, for $k = 256$, this tells us that after 256 applications of independent random oracles, the final range will be roughly 2^{-7} the size of the original domain.

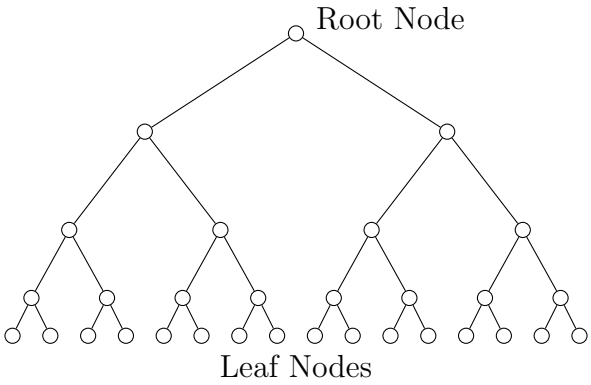

Figure 2.1: A Merkle Tree

2.2.4 Merkle Trees in the Random-Oracle Model

In this section we discuss how similar arguments from section 2.2.3 apply to Merkle trees.

Recall that Merkle trees are used to generically transform a one-time signature scheme into a many-time signature scheme that uses multiple instances of the one-time scheme. It takes in the parameter H which defines the height of the binary tree. Then there are 2^H one-time instances. By labelling the leaf nodes from 1 to 2^H, we define the values of the next level of the tree by hashing the values of adjacent leaf nodes (i.e., leaf nodes 1 and 2 are hashed together, 3 and 4 are hashed together, etc.), along with a string that identifies the location of the nodes in the overall tree.

We recursively build up the next level of the tree in this way until we have a root node, as in figure 2.1.

This definition allows us to consider the overall mapping from the set of leaf nodes (here n-bit strings) to the root node:

$$T : (\{0,1\}^n)^{2^H} \to \{0,1\}^n. \tag{2.5}$$

In section 2.2.3, we noted that by recursively applying independent oracles, the overall mapping had a somewhat smaller range than might be expected. Using similar analysis, we can show that this is *unlikely* to happen for the Merkle tree mapping.

Consider the first oracle, which takes in two leaf nodes, which are n-bit strings, and outputs the hash of their concatenation. This is a random mapping from $\{0,1\}^{2n}$ to $\{0,1\}^n$.

From our analysis in section 2.2.3, we know that the expected size of the range is

$$2^n\left(1-\left(1-\frac{1}{2^n}\right)^{2^{2n}}\right) \approx 2^n\left(1-\frac{1}{e^{2^n}}\right) = 2^n - 2^{n-\log_2(e)\cdot 2^n}. \tag{2.6}$$

So if we let M represent a random variable indicating the number of elements in the codomain that are missed by the mapping. Then by Markov's inequality,

$$Pr[M \geq 1] \leq \mathrm{Ex}[M] = 2^{n-\log_2(e)\cdot 2^n}. \tag{2.7}$$

For any reasonable value of n (such as 128 or 256), this probability is so incredibly small that it essentially negates the entire possibility of a single point in the codomain being missed. Considering that there are a total of $2^H - 1$ oracles, and H generally takes values on the order of 20 or 40, this implies that with overwhelming probability, not a single point in the codomain of the mapping T is missed.

2.3 Security Proof for OTLMS in the QROM

We define security in terms of the standard notion of existential unforgeability under chosen-message attack. This standard notion of security is defined in terms of the following interaction between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Game 2.3.1 (One-time existential-unforgeability under chosen-message attack (OTeucma)).

1. $\mathcal{C}$ chooses a random function $H : \{0,1\}^* \to \{0,1\}^n$ from all possible mappings (considering that there is in principle an upper bound on the length of binary strings $\mathcal{A}$ will ask for evaluation on). $\mathcal{C}$ then creates a quantum random oracle that provides quantum access to H as in equation 1.2.
2. $\mathcal{C}$ runs $\mathsf{OTKeyGen}$, obtaining (pk, sk), and sends pk to $\mathcal{A}$.
3. $\mathcal{A}$ makes q_1 queries to the quantum random oracle and then submits a message M' for signing.
4. $\mathcal{C}$ runs $\mathsf{OTSign}(M', sk)$ and sends the resulting signature, σ' to $\mathcal{A}$.
5. $\mathcal{A}$ makes q_2 queries to the quantum random oracle.
6. $\mathcal{A}$ submits a message-signature pair, (M^*, σ^*), such that $M^* \neq M'$.

We say that $\mathcal{A}$ has 'won' the $OTeucma$ game if $\mathsf{OTVrfy}(M^*, \sigma^*, pk) \rightarrow \mathsf{accept}$.

Theorem 2.3.1. *For any adversary $\mathcal{A}$, the probability that they win the $OTeucma$ game in the random-oracle model is $\leq (2^{n/6} + 38)q/\sqrt{2^n}$.*

Proof. We write a query $|\phi_i\rangle$ to the quantum random oracle as

$$|\phi_i\rangle = \sum_{x,y} \alpha_{i,x,y}|x\rangle|y\rangle.$$

As per our description of OTeucma, q_1 is the number of queries prior to the signing query, and q_2 is the number of queries after the signing query. Then $q_H = q_1 + q_2$ is the total number of queries to the quantum random oracle.

To upper bound $\mathcal{A}$'s chances of winning $OTeucma$, we define a few subsets of the domain of H.

- $S_{0,i,j} := \{x \in \{0,1\}^* : x = x'||I||Q||i||j||00,\ F_{I||Q||i}(x; j, E) = x_i^E\}$
- $S_1 := \{x \in \{0,1\}^* : x = {x'}_1^E|| \dots ||{x'}_p^E||01,\ H(x) = pk,\ ({x'}_1^E|| \dots ||{x'}_p^E) \neq (x_1^E|| \dots ||x_p^E)\}$
- $S_2 := \{x \in \{0,1\}^* : x = M||r||I||Q||02,\ H(x) = h', M \neq M'\}$.

Then we define the following three events that may occur over the course of the game $OTeucma$.

- $E0$ is the event that $\mathcal{A}$ has complete knowledge of some $x \in S_{0,i,j}$ for some i and j where $v_i' > j$.
- $E1$ is the event that $\mathcal{A}$ has complete knowledge of some $x \in S_1$.
- $E2$ is the event that $\mathcal{A}$ has complete knowledge of some $x \in S_2$.

Classically, it is simple to define what we mean by "complete knowledge" — it is simply whether or not the adversary has queried such an x in one of the defined sets. In our quantum setting, this is less meaningful, as a superposition of queries contains some amount of information about such subsets. Instead, we define this as meaning that the adversary has some classical information that allows the derivation of such an x with probability 1.

We will establish that if (M^*, σ^*) is a valid forgery, at least one of the three events has occurred. We do this by establishing that if a forgery has occurred, and events $E1$ and $E2$ did not occur, then $E0$ must have happened.

We are assuming that $\mathcal{A}$ has succeeded in submitting a forgery and that events $E1$ and $E2$ have not occurred. We will examine the properties of (M^*, σ^*) and show that $E0$ must have occurred.

When the adversary submits a forgery (M^*, σ^*), we can run the verification algorithm on this pair. Then the following values are derived in the process of running the verification algorithm.

- $M^*||r^*||I||Q||02$
- $x_1^{*E}||\ldots||x_p^{*E}||01$
- $\sigma_i^*||I||Q||i||v_i^*||00$, for $i = 1$ to p.

Now $E1$ did not occur, and if the verification algorithm accepts (M^*, σ^*), then we must have that $H(x_1^{*E}||\ldots||x_1^{*E}||01) = pk$, so we must have that $x_1^{*E}||\ldots||x_p^{*E}||01 \notin S_1$, and so $x_1^{*E}||\ldots||x_p^{*E} = x_1^E||\ldots||x_p^E$.

Similarly, $E2$ did not occur, and since $M^* \neq M'$, it must be the case that $H(M^*||r^*||I||Q||02) \neq h'$.

So we know that $h^* \neq h'$, and that $x_1^{*E}||\ldots||x_p^{*E} = x_1^E||\ldots||x_p^E$. Note that by the construction of the checksum, when we compare v^* and v', there must be an index i for which $v_i^* < v_i'$. But then since we have that $x_1^{*E} = x_1^E$, we can see that this means that $\sigma_i^*||I||q||i||v_i^*||00 \in S_{0,i,v_i^*}$ and $E0$ has occurred.

Now all we need to do is provide an upper bound on the probability of any of the events occurring. To upper bound these, we consider the adversary that attempts to maximize the probability of event Ei occurring, for $i = 0$, 1, 2.

Event $E0$ For event $E0$, we want to upper bound the adversary's ability on finding any new x, i, and j, with $j < v_i'$ and $x \in S_{0,i,j}$. Note that finding an $x \in S_{0,i,j}$ implies complete knowledge of some $x' \in S_{0,i,k}$, for $j \leq k < E$. In particular, it implies complete knowledge of some $x \in S_{0,i,v_i'-1}$. So we need to upper bound the adversaries ability to find such an x.

From the signing query, the adversary knows precisely one element of $S_{0,i,v_i'}$. However, we can imagine an adversary who knows this set entirely. We will show that finding an element of $S_{0,i,v_i'-1}$ is still difficult. From section 2.2.3, we know that when considering the function F as a composition of random oracles, we can expect the overall compression

from the domain to the codomain to be by a factor of roughly 2^{-7}. One consequence is that $S_{0,i,v_i'}$ will have size at most roughly 2^{-7}. Similarly, we can expect that the size of the range of the function $F_{I||Q||i}(\cdot; 0, v_i' - 1)$ is going to be larger than $2^{-7}2^n$. Let $\mathcal{R}_i$ be this range.

So we can imagine an adversary that for each i, knows entirely $\mathcal{R}_i$ and the set $S_{0,i,v_i'}$. In particular they can search over $\mathcal{R}_i$ and test membership in the set without any queries to an oracle (in practice of course, this should not be possible). The adversary still needs to find something in $\mathcal{R}_i$ that is mapped to a point in $S_{0,i,v_i'}$ by the oracle $H(\cdot||I||Q||i||v_i' - 1||00)$.

In the worst case, we can imagine that $S_{0,i,v_i'}$ is a strict subset of the range of $H(\cdot||I||Q||i||v_i' - 1||00)$ acting on $\mathcal{R}_i$. In this case, the adversary has to find one marked item out of a set of size $2^{-7}2^n$ where each item is marked with uniform and independent probability 2^{-7}. Therefore the adversary's ability to find such an item is determined by the Grover's lower bound of $2q/\sqrt{2^8/2^n} = 2^5 q\sqrt{2^n}$.

Event $E1$ Event $E1$ is simply the adversary's ability to find some distinct $x \neq x_1^e||\ldots||x_p^e$ that maps to pk under $H(\cdot||01)$, when the adversary is already given such an element. This is a game of second-preimage resistance and the quantum adversaries success probability is bounded by $2q/\sqrt{2^n}$.

Event $E2$ Event $E2$ refers to the adversary's ability to find a distinct M^* and any r^* such that $H(M^*||r^*||I||Q||02) = H(M'||r'||I||Q||02)$, where M' is chosen by the adversary and r' is chosen uniformly at random. But this is precisely the game of second-preimage resistance with adversary prefixes with respect to the random oracle $H(\cdot||\cdot||I||Q||02)$. So, the adversary's probability of success is at most $2^{n/6}q/\sqrt{2^n} \leq 2^2 q/\sqrt{2^n}$.

Altogether, we have that

$$\begin{aligned} Pr(\mathcal{A} \text{ wins } OTeucma) \leq & Pr(E0 \text{ or } E1 \text{ or } E2) \\ \leq & Pr(E0) + Pr(E1) + Pr(E2) \\ \leq & (2^5 + 2 + 2^{n/6})q/\sqrt{2^n} \\ = & (2^{n/6} + 34)q/\sqrt{2^n}. \end{aligned}$$

□

2.4 Security Proof for LMS in the QROM

For the full version of LMS, we proceed in a similar fashion, defining a full game of existential-unforgeability under chosen-message attack, and upper bounding the success

probability of any adversary in this game with respect to a random oracle.

Game 2.4.1 (LMS Existential-unforgeability under chosen message attack (eucma))**.**

1. $\mathcal{C}$ chooses a random function $H : \{0,1\}^* \to \{0,1\}^n$ from all possible mappings. $\mathcal{C}$ then creates a quantum random oracle that provides quantum access to H as in equation 1.2.
2. $\mathcal{C}$ runs KeyGen, obtaining (pk, sk), and sends pk to $\mathcal{A}$.
3. $\mathcal{A}$ makes quantum random oracle queries $|\phi_1\rangle, \ldots,$ and then submits a message M'_1 for signing.
4. $\mathcal{C}$ runs $\mathsf{Sign}(M'_1, sk)$ and sends the resulting signature, σ'_1 to $\mathcal{A}$.
5. $\mathcal{A}$ continues to make random oracle queries, as well as signing queries $M'_2, M'_3, \ldots, M'_{q_S}$.
6. $\mathcal{A}$ submits a message-signature pair, (M^*, σ^*), such that $M^* \neq M'_1, M'_2, \ldots, M'_{q_S}$.

As before, $\mathcal{A}$ is said to win if and only if $\mathsf{Vrfy}(M^*, \sigma^*, pk) \to$ 'accept'.

Theorem 2.4.1. *For any adversary $\mathcal{A}$, the probability that they wins game 2.4.1 in the random-oracle model is $\leq (2^{n/6} + 38)q/\sqrt{2^n}$.*

Proof. To upper bound $\mathcal{A}$'s probability of winning gamw 2.4.1, we again characterize several subsets and events based on these subsets, and characterize what a forgery may look like with respect to these subsets.

Parse the forged signature as $\sigma^* = \sigma'^*, Q^*, y^{*0}, \ldots, y^{*H-1}$. σ'^* is the signature for the one-time verification algorithm, Q^* is the placement identifier, and $y^{*0}, \ldots, y^{*H-1}$ form the Merkle tree verification path. Let $pk'^* \leftarrow \mathsf{OTVrfy}(M^*, \sigma'^*)$

- EOT is the event that $pk'^* = pk_{Q^*}$.
- $E3$ is the event that EOT did not occur and $H(pk'^*||I||Q^*||03) = y^0_{Q^*}$, where $y^0_{Q^*}$ is the value as defined in the KeyGen algorithm.
- $E4$ is the event that EOT and $E3$ did not occur and the adversary has complete knowledge of values $x', x_0, \ldots, x_{H-1}$, as well as an index i such that $x' \neq y^0_i$ and by computing the Merkle tree verification path with respect to these values and this index, we get the value pk.

It is straightforward to establish, as we did for the one-time signature scheme, that for a forgery to occur, one of these events must happen. We establish it in the same way, that is, if EOT and $E3$ do not occur for a forgery, then $E4$ must occur.

$E3$ and EOT not occurring means that $H(pk'^*||I||Q^*||03) \neq y^0_{Q^*}$, and so setting this to x', $i = Q^*$ and considering the $y^{*0}, \ldots, y^{*H-1}$, we know that since the signature verified, this verification path must lead to pk, and $E4$ has occurred.

Then we just need to upper bound the probability of any of these events occurring.

Event EOT This event corresponds to the possibility of an adversary breaking the one-time signature scheme for one of the Q^* instances of the one-time signature scheme. Note that the adversary's advantage in breaking any one of the one-time signature schemes is at most 2^H times their advantage in breaking one scheme, implying that their advantage is certainly at most

$$2^H \frac{(2^{n/6} + 34)q}{\sqrt{2^n}}.$$

In fact their advantage should be much less than this. As each one-time instance is fully independent due to the identifying information passed into each hash oracle, the adversary's advantage should simply be the same as it is for the one-time signature scheme. We note that each event that determines EOT is reduced exactly to either collision resistance or some second-preimage resistance problem. From a result in [27], no advantage is gained in having mutliple targets for the second-preimage or collision resistance, as long as the search spaces are disjoint. In this case, they are disjoint because of the identifier Q.

We can therefore bound the value EOT by

$$\frac{2^{n/6} + 34}{\sqrt{2^n}}. \tag{2.8}$$

Event $E3$ This is the event that the adversary finds a second preimage of one of the leaf nodes of the Merkle tree. As this is directly a second-preimage problem, the probability of this event happening is

$$\frac{2q}{\sqrt{2^n}}.$$

Event $E4$ This is the event that the adversary obtains knowledge of a 'false' verification path for the Merkle tree that leads to pk. Note that finding a verification path for an $x \neq y^0_i$ implies finding a second preimage to one of the values of the Merkle tree. The first preimage is the concatenation of the two values below that point on the tree. Note

that each point in the Merkle tree is generated by a random oracle that is independent from the oracles in the rest of the tree. Therefore by our discussions in section 2.2.4, the probability of event $E4$ happening is bounded by an adversary's ability to break one of the second-preimage instances of the Merkle tree, of which there are $2^H - 1$.

By the analysis in [27] we know that the probability of an adversary finding a second preimage for any of these $2^H - 1$ instances is the same as finding them for one, namely $\leq 2q/\sqrt{2^n}$.

Altogether, we have

$$\begin{aligned} Pr(\mathcal{A} \text{ wins } eucma) \leq & Pr(EOT \text{ or } E3 \text{ or } E4) \\ \leq & Pr(EOT) + Pr(E3) + Pr(E4) \\ \leq & (2^{n/6} + 34 + 2 + 2)q/\sqrt{2^n} \\ = & (2^{n/6} + 38)q/\sqrt{2^n}. \end{aligned}$$

□

Chapter 3

TESLA

When attempting to evaluate the security of a signature scheme, the security gap of the reduction is an important consideration. This quantity measures how much extra work a reduction must perform in order to convert an adversary capable of breaking the scheme into solving an underlying problem.

To determine the security gap, a reduction that explicitly relates the costs of breaking the scheme and carrying out the reduction is necessary. In contrast, many reductions establish only asymptotic security, resulting in a mere heuristic argument of a scheme's security. Moreover, as pointed out by Chatterjee et al. [17], tight reductions, i.e., with a security gap of approximately 1, provide a much higher degree of trust in the scheme's security. A large security gap could be explained by a yet to be discovered attack against the scheme, and therefore a conservative approach to selecting parameters must take the security gap into account.

To fully address the challenges in a quantum world, concrete instantiations of schemes should be chosen and their security should be analyzed with respect to adversaries that have access to large-scale quantum computers.

In the realm of lattice-based signature schemes, the only lattice-based signature scheme that is proven secure in the QROM and accounts for its security gap (which is close to 1) is the GPV-scheme by Gentry, Peikert, and Vaikuntanathan [23]. Lattice-based schemes that are significantly more efficient with respect to run time and space [20, 4, 18, 25] have not been proven secure in the QROM and/or do not account for their proven security gap.

In this chapter we consider a candidate signature scheme for standardization in a quantum world: TESLA. We prove this signature scheme to be tightly secure in the QROM.

The scheme TESLA is a variant of Bai and Galbraith's earlier construction [4]. Bai and Galbraith give a (non-tight) security reduction from the Learning with Errors problem (LWE) [37] and the Short Integer Solution (SIS) problem in the random-oracle model. This scheme was then reconsidered (and called TESLA) in [2]. However the proof contained in that document contained flaws, as pointed out independently by Gus Gutoski and Chris Peikert. Additionally, the analysis in the quantum random-oracle model was somewhat dissatisfying, as it required additional computational assumptions.

3.1 Chapter Notation

Integer scalars are denoted using Roman letters and if not stated otherwise, q is a prime integer. For any positive integer n the set $\mathbb{Z}_n$ of integers modulo n is represented by $\{-\lfloor (n-1)/2 \rfloor, \ldots, \lfloor n/2 \rfloor\}$. Fix a positive integer d and define the functions $[\cdot], [\cdot]_L : \mathbb{Z} \to \mathbb{Z}$ as follows. For any integer x let $[x]_L$ denote the representative of x in $\mathbb{Z}_{2^d}$, i.e., $[x]_L = x (\text{mod } 2^d)$, and let $[x] = (x - [x]_L)/2^d$. Informally, $[x]$ is viewed as the *most significant bits* of x and $[x]_L$ is viewed as the *least significant bits* of x. The definitions are easily extended to vectors by applying the operators for each component.

Vectors with entries in $\mathbb{Z}_q$ are viewed as column vectors and denoted with lowercase Roman letters in sans-serif font, e.g., $\mathsf{y}, \mathsf{z}, \mathsf{w}$. Matrices with entries in $\mathbb{Z}_q$ are denoted with uppercase Roman letters in sans-serif font, e.g., $\mathsf{A}, \mathsf{S}, \mathsf{E}$. The transpose of a vector or a matrix is denoted by v^T or M^T, respectively. We denote by $\|\mathsf{v}\|$ the Euclidean norm of a vector v, and by $\|\mathsf{v}\|_\infty$ its infinity norm. An integer vector y is *B-short* if each entry is at most B in absolute value. All logarithms are base 2. A function is called negligible in the security parameter n, denoted by $negl(n)$, if it decreases faster than the inverse of every polynomial in n, for sufficiently large n.

The centered discrete Gaussian distribution for $x \in \mathbb{Z}$ with standard deviation σ is defined to be $\mathcal{D}_\sigma = \rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$, where $\sigma > 0$, $\rho_\sigma(x) = \exp(\frac{-x^2}{2\sigma^2})$, and $\rho_\sigma(\mathbb{Z}) = 1 + 2\sum_{x=1}^{\infty} \rho_\sigma(x)$.

For a finite set S, we denote sampling the element s uniformly from S by $s \xleftarrow{\$} \mathcal{U}(S)$ or simply $s \xleftarrow{\$} S$. Let χ be a distribution over $\mathbb{Z}$, then we write $x \leftarrow \chi$ if x is sampled according to χ. Moreover, we denote sampling each coordinate of a matrix $\mathsf{A} \in \mathbb{Z}^{m \times n}$ with distribution χ by $\mathsf{A} \leftarrow \chi^{m \times n}$ with $m, n \in \mathbb{Z}_{>0}$. For an algorithm $\mathcal{A}$, the value $y \leftarrow \mathcal{A}(x)$ denotes the output of $\mathcal{A}$ on input x; if $\mathcal{A}$ uses randomness then $\mathcal{A}(x)$ is a random variable. $\mathcal{A}^\chi$ denotes that $\mathcal{A}$ can request samples from the distribution χ.

3.2 The Learning with Errors Problem

In the following we recall the decisional learning with errors problem (LWE) and define its matrix vararian (M-LWE) [1].

Definition 3.2.1 (Learning with Errors Problem)**.** *Let $n, m, q > 0$ be integers and χ be a distribution over $\mathbb{Z}$. For $\mathsf{s} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n)$ define $\mathcal{A}_{\mathsf{s},\chi}$ to be the distribution that samples $\mathsf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $\mathsf{e} \leftarrow \chi$ and then returns $(\mathsf{a}, \langle \mathsf{a}, \mathsf{s} \rangle + \mathsf{e}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. The decisional learning with errors problem $LWE_{n,m,q,\chi}$ is (t, ε)-hard if for any algorithm $\mathcal{D}$, running in time t and making at most m queries to the distribution $\mathcal{A}_{\mathsf{s},\chi}$, it holds that*

$$\left| \Pr\left[\mathcal{D}^{\mathcal{A}_{\mathsf{s},\chi}}(\cdot) = 1\right] - \Pr\left[\mathcal{D}^{\mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)}(\cdot) = 1\right] \right| \leq \varepsilon \ .$$

We can also write m LWE instances to a secret $\mathsf{s} \in \mathbb{Z}_q^n$ as $(\mathsf{A}, \mathsf{As} + \mathsf{e} \mod q)$ with $\mathsf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ and $\mathsf{e} \leftarrow \chi^m$.

The security of the signature scheme covered in this paper is based on the matrix version of LWE (M-LWE) defined in the following.

Definition 3.2.2 (Matrix Learning with Errors Problem)**.** *Let $n, n', m, q > 0$ be integers and χ be a distribution over $\mathbb{Z}$. Define $\mathcal{A}_{\mathsf{S},\chi}$ to be the distribution that, for $\mathsf{S} = (\mathsf{s}_1, ..., \mathsf{s}_{n'})$ with $\mathsf{s}_1, ..., \mathsf{s}_{n'} \xleftarrow{\$} \mathcal{U}\left(\mathbb{Z}_q^n\right)$, samples $\mathsf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $e_1, ..., e_{n'} \leftarrow \chi$ and then returns $\left(\mathsf{a}^T, \mathsf{a}^T\mathsf{S} + (e_1, ..., e_{n'})\right) \in \mathbb{Z}_q^{1 \times n} \times \mathbb{Z}_q^{1 \times n'}$. The matrix decisional learning with errors problem $M\text{-}LWE_{n,n',m,q,\chi}$ is (t, ε)-hard if for any algorithm $\mathcal{D}$, running in time t and making at most m queries to the distribution $\mathcal{A}_{\mathsf{S},\chi}$, it holds that*

$$\left| \Pr\left[\mathcal{D}^{\mathcal{A}_{\mathsf{S},\chi}}(\cdot) = 1\right] - \Pr\left[\mathcal{D}^{\mathcal{U}\left(\mathbb{Z}_q^n \times \mathbb{Z}_q^{n'}\right)}(\cdot) = 1\right] \right| \leq \varepsilon \ .$$

As before, m M-LWE samples to the secret matrix $\mathsf{S} = (\mathsf{s}_1, ..., \mathsf{s}_{n'}) \in \mathbb{Z}_q^{n \times n'}$ can be written as $(\mathsf{A}, \mathsf{AS} + \mathsf{E}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times n'}$ with $\mathsf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ and $\mathsf{E} \leftarrow \chi^{m \times n'}$.

We call $(\mathsf{A}, \mathsf{T}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times n'}$ a **yes-instance** if there exists an $\mathsf{S} = (\mathsf{s}_1, ..., \mathsf{s}_{n'})$ with $\mathsf{s}_1, ..., \mathsf{s}_{n'} \in \mathbb{Z}_q^n$ and (A, T) are m M-LWE samples from the distribution $\mathcal{A}_{\mathsf{S},\chi}$. Otherwise, i.e.when $(\mathsf{A}, \mathsf{T}) \xleftarrow{\$} \mathcal{U}\left(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times n'}\right)$, we call (A, T) a **no-instance**.

[1] Note that, in contrast with common terminology in papers about module lattices, we do not mean *Module LWE* by the abbreviation M-LWE, but *Matrix LWE*.

Theorem 3.2.1. *If $LWE_{n,m,q,\chi}$ is $(\varepsilon/n', t)$-hard then M-$LWE_{n,n',m,q,\chi}$ is (ε, t)-hard.*

Intuitively, the reduction loss exists since an adversary that can solve LWE has n' possibilities to solve M-LWE (see also [12, 36, 4]). The proof follows similar arguments as given in [36].

For the remainder of this chapter, whenever we refer to LWE, we mean the matrix version, M-LWE.

We note that the hardness of LWE (resp., M-LWE) is retained even if all coordinates of the secret vector s are sampled according to the error distribution χ, known as the "normal form" [32, 3]. We use the notation $\text{LWE}_{n,m,q,\sigma}$ if χ is distributed according to $\mathcal{D}_\sigma$. The LWE assumption comes with a worst-to-average-case reduction [37, 35, 14]; breaking certain average instances of LWE allows one to break all instances of certain standard lattice problems (namely GapSVP and SIVP).

3.3 The Signature Scheme TESLA

TESLA's key generation, sign, and verify algorithms are listed in Algorithms 10, 11, and 12. For more details, rationale, comparisons with other schemes, and implementation results, see [1].

Parameters and Notation. TESLA is parameterized by positive integers q, m, n, n', h, d, B, L, L_S, U, a positive real σ, a hash oracle $H(\cdot)$, and the publicly available matrix $\mathsf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$. Let $\mathbb{H}$ denote the set of vectors $c \in \{-1, 0, 1\}^{n'}$ with exactly h nonzero entries. For simplicity we assume that the hash oracle $H(\cdot)$ has range $\mathbb{H}$, i.e., we ignore how a standard hash function like SHA-256 is converted into one which has range $\mathbb{H}$. An integer vector w is *well-rounded* if w is $(\lfloor q/2 \rfloor - L)$-short and $\lceil \mathsf{w} \rfloor$ is $(2^d - L)$-short.

The parameters selected must satisfy the bounds in table 3.1.

Deterministic signatures. It is recommended that any implementation of TESLA employ standard techniques to achieve deterministic signatures. In particular, the key generation algorithm should produce a random seed as part of the secret key. The sign algorithm should use this seed, along with the input message msg, to derive a deterministic sequence of pseudorandom data that dictate the "random" choices of Algorithm 11.

σ	$> 2\sqrt{n}$
L_S	$14\sigma h$
h	$2^h\binom{n'}{h} \geq 2^{5\lambda}$
B	$\geq 14n\sqrt{h}\sigma$
U	$\lceil 14\sqrt{h}\sigma \rceil$
d	$(1 - 2L/2^d)^m \geq 0.3$

Table 3.1: TESLA parameter bounds

Fixed-weight hash outputs. Any implementation of the hash oracle $H(\cdot)$ will require an encoding function that embeds the output of a concrete hash function such as SHA-256 into the set $\mathbb{H}$ (see [25] for more information on embedding functions of this type). Naturally, the output length of the underlying hash function should be large enough so as to preclude collision attacks.

Additional checks in KeyGen and Sign. In contrast to earlier proposals [4, 18], we add an additional check during the signature generation. Namely, to ensure correctness of the scheme, we check that the absolute value of each coordinate of $\mathsf{Ay} - \mathsf{Ec}$ is less than or equal to $\lfloor q/2 \rfloor - L$. To achieve better concrete bounds during the security reduction, we add another check to ensure that no coefficient of the matrix S is too large. The parameter L_S is chosen such that the probability of rejecting S is smaller than 2^{-n}.

Algorithm 10 KeyGen

Input: A.

Output: Public key T, secret key (S, E).

1: Choose entries of $\mathsf{S} \in \mathbb{Z}_q^{n \times n'}$ and $\mathsf{E} \in \mathbb{Z}_q^{m \times n'}$ from the centered discrete Gaussian distribution with standard deviation σ.
2: If E has a row whose h largest entries sum to L or more then retry at step 1.
3: If S has a row whose h largest entries sum to L_S or more then retry at step 1.
4: $\mathsf{T} \leftarrow \mathsf{AS} + \mathsf{E}$.
5: Return public key (A, T) and secret key (S, E).

Algorithm 11 Sign

Input: Message msg, secret key (S, E).
Output: Signature (z, c).

1: Choose y uniformly at random among B-short vectors from $\mathbb{Z}_q^n$.
2: $\mathsf{c} \leftarrow H([\mathsf{Ay}], \mathsf{msg})$.
3: $\mathsf{z} \leftarrow \mathsf{y} + \mathsf{Sc}$.
4: If z is not $(B-U)$-short then retry at step 1.
5: If $\mathsf{Ay} - \mathsf{Ec}$ is not well-rounded then retry at step 1.
6: Return signature (z, c).

Algorithm 12 Verify

Input: Message msg, public key (A, T), purported signature (z, c).
Output: "Accept" or "reject".

1: If z is not $(B-U)$-short then reject.
2: If $H([\mathsf{Az} - \mathsf{Tc}], \mathsf{msg}) \neq \mathsf{c}$ then reject.
3: Accept.

3.4 Brief Sketch of Security Proof for TESLA

Our main theorem on the security of TESLA is as follows.

Theorem 3.4.1 (Security of TESLA)**.** *Let q, m, n, n', h, d, B, L, L_S, U, σ, λ, κ be TESLA parameters that are convenient (according to Definition 3.8.1 in Section 3.8) and that satisfy the bounds in table 3.1.*[2] *If M-LWE is (t, ϵ)-hard then TESLA is existentially $(t', \epsilon', q_h, q_s)$-unforgeable against adaptively chosen message attacks with $t' \approx t$ in (i) the* quantum *random-oracle model with*

$$\epsilon' < \epsilon + \frac{3}{2^\lambda} + \frac{2^{m(d+1)+3\lambda+1}}{q^m}(q_h + q_s)^2 q_s^3 + 2(q_h+1)\sqrt{\frac{1}{2^h\binom{n'}{h}}}, \tag{3.1}$$

and in (ii) the classical *random-oracle model with*

$$\epsilon' < \epsilon + \frac{3}{2^\lambda} + \frac{2^{m(d+1)+3\lambda+1}}{q^m}(q_h + q_s)^2 q_s^3 + q_h\frac{1}{2^h\binom{n'}{h}}. \tag{3.2}$$

[2] It is not necessary that TESLA parameters be convenient in order to derive negligibly small upper bounds on ϵ'; the definition of convenience merely facilitates a simplified statement of those bounds.

The proof of Theorem 3.4.1 is given in Sections 3.5, 3.6, 3.7, 3.8. In this section we present a sketch of this proof and a selection of some intermediate results we feel are the most significant technical contributions of the proof.

Let $\mathcal{F}$ be a forger that forges TESLA signatures with probability $\Pr[\text{forge}(\mathsf{A}, \mathsf{T})]$. We build an LWE-solver $\mathcal{S}$ whose run time is close to that of $\mathcal{F}$ and whose success bias is close to $\Pr[\text{forge}(\mathsf{A}, \mathsf{T})]$. It then follows from the presumed hardness of LWE that $\Pr[\text{forge}(\mathsf{A}, \mathsf{T})]$ must be small.

Given an LWE input (A, T), the LWE-solver $\mathcal{S}$ treats (A, T) as a TESLA public key; $\mathcal{S}$ runs $\mathcal{F}$ on input (A, T) and outputs "yes" if and only if $\mathcal{F}$ succeeds in forging a TESLA signature.

In order to run $\mathcal{F}$, the LWE-solver $\mathcal{S}$ must respond in some way to $\mathcal{F}$'s quantum queries to the hash oracle and to $\mathcal{F}$'s classical queries to the sign oracle. Our description of $\mathcal{S}$ includes a procedure for responding to these queries.

That $\mathcal{S}$ solves LWE with success bias close to $\Pr[\text{forge}(\mathsf{A}, \mathsf{T})]$ is a consequence of the following facts:

1. For yes-instances of LWE, the probability with which $\mathcal{S}$ outputs "yes" is close to $\Pr[\text{forge}(\mathsf{A}, \mathsf{T})]$.
2. For no-instances of LWE, $\mathcal{F}$ successfully forges (and hence $\mathcal{S}$ outputs "yes") with only negligible probability.

3.4.1 Yes-Instances of LWE

We argue that $\mathcal{S}$'s responses to $\mathcal{F}$'s oracle queries are indistinguishable from the responses $\mathcal{F}$ would receive from real oracles, from which it follows that $\mathcal{S}$ reports "yes" with probability close to $\Pr[\text{forge}(\mathsf{A}, \mathsf{T})]$.

Each time $\mathcal{S}$ simulates a call to the sign oracle, it must "re-program" its simulated hash oracle on one input. Because $\mathcal{F}$ is permitted to make quantum queries to the hash oracle, we must show that $\mathcal{F}$ is unlikely to notice when a quantum random oracle has been re-programmed.

To this end, let $\mathbb{Y}$ denote the set of vectors $\mathsf{y} \in \mathbb{Z}_q^n$ such that y is B-short and define the following quantities for each choice of TESLA keys $(\mathsf{A}, \mathsf{T}), (\mathsf{S}, \mathsf{E})$:

$\text{nwr}(\mathsf{A}, \mathsf{E})$: The probability over $(\mathsf{y}, \mathsf{c}) \in \mathbb{Y} \times \mathbb{H}$ that $\mathsf{Ay} - \mathsf{Ec}$ is not well-rounded.

$\text{coll}(\mathsf{A}, \mathsf{E})$: The maximum over all $\mathsf{w} \in \left\{[\mathsf{x}] : \mathsf{x} \in \mathbb{Z}_q^m\right\}$ of the probability over $(\mathsf{y}, \mathsf{c}) \in \mathbb{Y} \times \mathbb{H}$ that $[\mathsf{Ay} - \mathsf{Ec}] = \mathsf{w}$.

We prove the following (cf. Proposition 3.6.3 in Section 3.6.6).

Proposition 3.4.1 (Re-Programming in TESLA, Informal Statement)**.** *The following holds for each choice of TESLA keys* $(\mathsf{A},\mathsf{T}),(\mathsf{S},\mathsf{E})$*, each hash oracle* $H(\cdot)$*, and each* $\gamma > 0$*.*

Suppose the quantum state ρ_H *was prepared by some party* $\mathcal{D}$ *using* t *quantum queries to* $H(\cdot)$*. Let* $H'(\cdot)$ *be a hash oracle that agrees with* $H(\cdot)$ *except on a small number of randomly chosen inputs* $(\cdot,\mathsf{msg})$ *for each possible message* msg*. Let* $\rho_{H'}$ *be the state prepared when* $\mathcal{D}$ *uses hash oracle* $H'(\cdot)$ *instead of* $H(\cdot)$*. Then* $\|\rho_{H'}-\rho_H\|_{\mathrm{Tr}} < \gamma$ *except with probability at most*

$$\frac{t^2}{\gamma^2}\frac{\mathrm{coll}(\mathsf{A},\mathsf{E})}{1-\mathrm{nwr}(\mathsf{A},\mathsf{E})} \tag{3.3}$$

over the choice of inputs upon which $H(\cdot)$ *and* $H'(\cdot)$ *differ.*

We also prove bounds on $\mathrm{nwr}(\mathsf{A},\mathsf{E})$ and $\mathrm{coll}(\mathsf{A},\mathsf{E})$ that hold with high probability over the choice of TESLA keys $(\mathsf{A},\mathsf{T}),(\mathsf{S},\mathsf{E})$.

3.4.2 No-Instances of LWE

We argue that, except with negligibly small probability over the choice of hash oracle $H(\cdot)$ and LWE no-instance (A,T), a TESLA forger cannot forge a signature for (A,T) without making an intractably large number of queries to the hash oracle.

To forge a signature for message msg a forger must find a hash input $(\mathsf{w},\mathsf{msg})$ whose output $\mathsf{c} = H(\mathsf{w},\mathsf{msg})$ has the property that there exists a $(B-U)$-short $\mathsf{z}\in\mathbb{Z}_q^n$ for which $[\mathsf{Az}-\mathsf{Tc}] = \mathsf{w}$. Let $\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T})\subset\mathbb{H}$ denote the set of all such c. A hash input $(\mathsf{w},\mathsf{msg})$ is called *good for* $H(\cdot),\mathsf{A},\mathsf{T}$ if $H(\mathsf{w},\mathsf{msg})\in\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T})$. (Once a good hash input has been found, the forger must then somehow *find* the vector z witnessing this fact. For our purpose, we assume that the forger gets it for free.)

For each LWE no-instance (A,T), a given hash input $(\mathsf{w},\mathsf{msg})$ is good for $H(\cdot),\mathsf{A},\mathsf{T}$ with probability

$$\frac{\#\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T})}{\#\mathbb{H}} \tag{3.4}$$

over the choice of hash oracle $H(\cdot)$. In Section 3.7 we argue that, except with negligibly small probability over the choice of $H(\cdot),(\mathsf{A},\mathsf{T})$, the fraction of hash inputs that are good is at most the expectation over LWE no-instances (A,T) of the ratio (3.4), maximized over all $\mathsf{w}\in\{[x]:x\in\mathbb{Z}_q^m\}$. We then prove the following (*cf.* Proposition 3.7.2 in Section 3.7).

Proposition 3.4.2 (Good Hash Inputs are Rare)**.** *If the TESLA parameters are convenient (according to Definition 3.8.1 in Section 3.8) then*

$$\mathop{\mathrm{Ex}}_{(\mathsf{A},\mathsf{T})}\left[\max_{\mathsf{w}}\left\{\frac{\#\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T})}{\#\mathbb{H}}\right\}\right] \leq \frac{1}{\#\mathbb{H}}. \tag{3.5}$$

Thus, the fraction of good hash inputs is at most $1/\#\mathbb{H}$ except with vanishingly small probability over the choice of hash oracle $H(\cdot)$ and LWE no-instance (A,T).

Since each hash input is good with a fixed probability independent of other hash inputs, the set of all good hash inputs is a randomly selected set. Thus, the only way to discover a good input is via search through an unstructured space. It then follows from known lower bounds for quantum search over an unstructured space that the forger cannot find a good hash input—and thus a TESLA forgery—using only q_h quantum queries to the hash oracle.

3.5 Overview of Security Proof

As in Section 3.4, let $\mathcal{F}$ be a forging algorithm that, on input a TESLA public key (A,T), makes no more than q_h quantum queries to a hash oracle $H(\cdot)$ and no more than q_s classical queries to a TESLA sign oracle for public key (A,T). Let $\Pr[\text{forge}(\mathsf{A},\mathsf{T})]$ denote the probability that $\mathcal{F}$ produces a valid TESLA forgery. We build an LWE-solver $\mathcal{S}$ whose run time is close to that of $\mathcal{F}$ and who solves LWE with success bias close to $\Pr[\text{forge}(\mathsf{A},\mathsf{T})]$. It then follows from the presumed hardness of LWE that $\Pr[\text{forge}(\mathsf{A},\mathsf{T})]$ must be small.

The LWE-solver $\mathcal{S}$ is described in Algorithm 13. Classical queries made by $\mathcal{F}$ to the sign oracle are simulated by $\mathcal{S}$ as specified in Simulated sign (Algorithm 14). Quantum queries made by $\mathcal{F}$ to the hash oracle are simulated by $\mathcal{S}$ according to the construction of Zhandry based on $2q_h$-wise independent functions [46]. (Alternately, if performance of the simulator is a concern then one could instead use a quantum-resistant pseudorandom function.)

That $\mathcal{S}$ solves LWE with success bias close to $\Pr[\text{forge}(\mathsf{A},\mathsf{T})]$ is a consequence of the following facts, which are proven in subsequent sections:

Subsection 3.6: For yes-instances of LWE, the probability with which $\mathcal{S}$ outputs "yes" is close to $\Pr[\text{forge}(\mathsf{A},\mathsf{T})]$.

Subsection 3.7: For no-instances of LWE, $\mathcal{F}$ successfully forges (and hence $\mathcal{S}$ outputs "yes") with only negligible probability.

Algorithm 13 LWE-solver $\mathcal{S}$ using a forger $\mathcal{F}$.

Input: A LWE instance (A, T).
Output: "Yes" or "no".

1: Invoke the forger $\mathcal{F}$ with public key (A, T).
Whenever $\mathcal{F}$ makes a hash or sign query, simulate that query as follows:

Classical sign queries. Execute Simulated sign (Algorithm 14).

Quantum hash queries. Apply a quantum circuit that implements a random but fixed $2q_h$-wise independent function, except on inputs that have been re-programmed by Simulated sign.

2: Eventually, $\mathcal{F}$ produces a purported forgery.
If that forgery is legitimate then output "yes", otherwise output "no".

Algorithm 14 Simulated sign

Input: Message msg, public key (A, T).
Output: Signature (z, c).

1: Choose z uniformly at random among $(B-U)$-short vectors from $\mathbb{Z}_q^n$.
2: Choose $\mathsf{c} \in \mathbb{H}$ uniformly at random.
3: If $\mathsf{Az} - \mathsf{Tc}$ is not well-rounded then retry at step 1.
4: Re-program the hash oracle $H(\cdot)$ so that $H([\mathsf{Az} - \mathsf{Tc}], \mathsf{msg}) = \mathsf{c}$.
5: Return (z, c).

3.5.1 Notation and Sizes for Various Sets of Vectors

Discussion in sections 3.6, 3.7, 3.8 refers to the following sets of integer vectors:

$\mathbb{Y}$:	The set of vectors $\mathsf{y} \in \mathbb{Z}_q^n$ such that y is B-short.
$\Delta\mathbb{Y}$:	$\{\mathsf{y} - \mathsf{y}' : \mathsf{y}, \mathsf{y}' \in \mathbb{Y}\}$.
$\mathbb{S}$:	The set of vectors $\mathsf{z} \in \mathbb{Z}_q^n$ such that z is $(B-U)$-short.
$\Delta\mathbb{S}$:	$\{\mathsf{z} - \mathsf{z}' : \mathsf{z}, \mathsf{z}' \in \mathbb{S}\}$.
$\mathbb{H}$:	The set of vectors $c \in \{-1, 0, 1\}^{n'}$ with exactly h nonzero entries.
$\Delta\mathbb{H}$:	$\{\mathsf{c} - \mathsf{c}' : \mathsf{c}, \mathsf{c}' \in \mathbb{H}\}$.
$\mathbb{W}$:	The set $\{[\mathsf{w}] : \mathsf{w} \in \mathbb{Z}_q^m\}$ of integer vectors obtained from the high bits of a vector in $\mathbb{Z}_q^m$.
$\Delta\mathbb{L}$:	$\{\mathsf{x} - \mathsf{x}' : \mathsf{x}, \mathsf{x}' \in \mathbb{Z}_q^m \text{ and } [\mathsf{x}] = [\mathsf{x}']\} = [-(2^d - 1), 2^d - 1]^m$.

The sizes of some of these sets are listed below:

$$\#\mathbb{Y} = (2B-1)^n \qquad \#\Delta\mathbb{Y} = (4B-1)^n \tag{3.6}$$

$$\#\mathbb{S} = (2(B-U)-1)^n \qquad \#\Delta\mathbb{S} = (4(B-U)-1)^n \tag{3.7}$$

$$\#\mathbb{H} = \binom{n'}{h} 2^h \tag{3.8}$$

$$\#\Delta\mathbb{L} = (2^{d+1}-1)^m \tag{3.9}$$

The size of $\Delta\mathbb{H}$ is computed as follows.

Lemma 3.5.1 (Size of $\Delta\mathbb{H}$)**.** *We have*

$$\#\Delta\mathbb{H} = \sum_{k=0}^{h} \sum_{i=0}^{h-k} \binom{n'}{2i} 2^{2i} \binom{n'-2i}{k} 2^k. \tag{3.10}$$

Proof. For each $k = 0, \ldots, h$ let $\Delta\mathbb{H}_k \subset \Delta\mathbb{H}$ denote the set of vectors in $\Delta\mathbb{H}$ with exactly k entries in $\{-2, 2\}$ and exactly $n' - k$ entries in $\{-1, 0, 1\}$. Observe that $\#\Delta\mathbb{H} = \sum_{k=0}^{h} \#\Delta\mathbb{H}_k$.

Fix k and for each $i = 0, \ldots, h-k$ let $\Delta\mathbb{H}_{k,i} \subset \Delta\mathbb{H}_k$ denote the set of vectors in $\Delta\mathbb{H}_k$ with exactly $2i$ entries in $\{-1, 1\}$. Observe that $\#\Delta\mathbb{H}_k = \sum_{i=0}^{h-k} \#\Delta\mathbb{H}_{k,i}$.

One can count the number of elements in each $\Delta\mathbb{H}_{k,i}$ as

$$\#\Delta\mathbb{H}_{k,i} = \binom{n'}{2i} 2^{2i} \binom{n'-2i}{k} 2^k, \tag{3.11}$$

from which the lemma follows. □

3.6 Yes-Instances of LWE

In this section we establish a lower bound on the probability

$$\Pr\left[\mathcal{S} \text{ output "yes"} \mid (\mathsf{A}, \mathsf{T}) \text{ yes-instance of LWE}\right] \tag{3.12}$$

in terms of $\Pr\left[\text{forge}(\mathsf{A}, \mathsf{T})\right]$. This is accomplished by proving that the simulated oracles are indistinguishable from the real oracles.

To this end we consider an arbitrary distinguisher $\mathcal{D}$ who, like the forger $\mathcal{F}$, makes at most q_h quantum queries to the hash oracle and at most q_s classical queries to the sign oracle. Unlike $\mathcal{F}$, however, $\mathcal{D}$'s goal is merely to distinguish the real oracles from the simulated oracles.

3.6.1 Adaptively Chosen Queries

An arbitrary distinguisher could adaptively and arbitrarily interleave its hash and sign queries. To facilitate our analysis we wish to model the distinguisher in such a way that sign queries occur at fixed, predictable points throughout the protocol. This goal is accomplished by a continuous accounting method for hash queries that we describe presently.

Standard formalism specifies that a quantum oracle for $H : X \to Y$ be implemented by a unitary channel $|x\rangle|y\rangle \mapsto |x\rangle|y + H(x)\rangle$. We modify this formalism so that the unitary channel for $H(\cdot)$ is a *controlled* unitary channel. Specifically, the channel acts on an additional qubit, the state of which dictates whether the unitary channel is applied:

$$|\text{off}\rangle|x\rangle|y\rangle \mapsto |\text{off}\rangle|x\rangle|y\rangle \tag{3.13}$$

$$|\text{on}\rangle|x\rangle|y\rangle \mapsto |\text{on}\rangle|x\rangle|y + H(x)\rangle. \tag{3.14}$$

Consider a new type of distinguisher with the following properties:

1. The distinguisher makes $q_h q_s$ hash queries instead of just q_h hash queries.
2. Exactly one sign query occurs after every q_h hash queries.
3. For each choice of hash oracle $H(\cdot)$, the distinguisher's total "query magnitude" (in the BBBV sense — see Section 3.6.6) on query states with the control qubit set to $|\text{on}\rangle$ over all $q_h q_s$ hash queries does not exceed q_h.

A distinguisher of this form is called a *live-switch distinguisher*. For later convenience, we refer to the query magnitude on states with the control qubit set to $|\text{on}\rangle$ as the *query magnitude on the live-switch*.

Intuition: a live-switch distinguisher can make "partial" queries to the hash oracle. If its query state has only α amplitude on the live-switch then the distinguisher is "charged" for only a a $|\alpha|^2$-fraction of a query.

It is clear that any ordinary distinguisher who makes q_h hash queries and q_s sign queries, adaptively chosen and interleaved, could be simulated by a live-switch distinguisher. Thus, live-switch distinguishers are at least as powerful as ordinary distinguishers, and possibly more powerful. We will prove indistinguishability against a live-switch distinguisher, which is more security than is strictly necessary.

The benefit of the live-switch distinguisher is that sign queries occur at fixed points throughout the protocol — one sign query after every q_h hash queries. This property allows us to partition the interaction into q_s *blocks*. Each block consists of q_h quantum queries to the hash oracle, followed by a single classical query to the sign oracle. We prove security of each block and then claim security of the entire interaction inductively.

3.6.2 The Distinguisher's State, a First Look

To begin, consider the state of $\mathcal{D}$'s system immediately prior to the sign oracle query in the first block. At this point in the interaction the real and simulated oracles are perfectly identical — both respond to the first q_h hash queries in accordance with some fixed choice of hash oracle $H(\cdot)$. Let ρ_H denote the state of $\mathcal{D}$'s system at this point in the interaction, conditioned on $H(\cdot)$. The sign oracle (both real and simulated) acts as follows on $\mathcal{D}$'s system:

1. Measure the message register, resulting in outcome msg.
2. Select a signature (z, c) for message msg.
3. Prepare an output register in the classical basis state $|\mathsf{msg}\rangle|(\mathsf{z}, \mathsf{c})\rangle$.

These actions can be viewed as a quantum channel. If the sign oracle is a real sign oracle then the signature (z, c) is a function of private randomness and the hash oracle $H(\cdot)$. In this case, the channel is denoted $\Psi_{\text{real},H}$. If the sign oracle is a simulated sign oracle then the signature (z, c) is a function only of private randomness. In this case, the channel is denoted Ψ_{simsign}.

Thus, the state of $\mathcal{D}$'s system at the end of the first block, conditioned on the choice of $H(\cdot)$, is either $\Psi_{\text{simsign}}(\rho_H)$ or $\Psi_{\text{real},H}(\rho_H)$. We will argue that the state $\Psi_{\text{simsign}}(\rho_H)$ is δ-close to a probabilistic mixture over re-programmed hash oracles $H'(\cdot)$ of states of the form $\Psi_{\text{real},H'}(\rho_{H'})$.

This δ-closeness is preserved by the hash queries in the second block of the interaction, since both the real and simulated hash oracles remain consistent with $H'(\cdot)$ in this block. Let $\rho_{2,H'}$ denote the state of $\mathcal{D}$'s system immediately prior to the sign oracle query in the second block. As above, we have that $\Psi_{\text{simsign}}(\rho_{2,H'})$ is δ-close to a mixture of states of the form $\Psi_{\text{real},H''}(\rho_{2,H''})$.

Continuing inductively, we see that the state of $\mathcal{D}$'s system at the end of an interaction with simulated oracles is $q_s\delta$-close to a probabilistic mixture over hash oracles of states of $\mathcal{D}$'s system at the end of an interaction with real oracles. Averaging over the choice of initial hash oracle $H(\cdot)$, we then see that the simulated oracles are indistinguishable from the real oracles.

We formalize these arguments in subsequent sections.

3.6.3 Mid-Sign

Consider the sign oracle Mid-sign of Algorithm 15. Mid-sign should be viewed as a hybrid

Algorithm 15 Mid-sign

Input: Message msg, public key (A, T), secret key (S, E).
Output: Signature (z, c).

1: Choose $(\mathsf{y}, \mathsf{c}) \in \mathbb{Y} \times \mathbb{H}$ uniformly at random.
2: $\mathsf{z} \leftarrow \mathsf{y} + \mathsf{Sc}$.
3: If $\mathsf{z} \notin \mathbb{S}$ then retry at step 1.
4: If $\mathsf{Ay} - \mathsf{Ec}$ is not well-rounded then retry at step 1.
5: Re-program the hash oracle $H(\cdot)$ so that $H([\mathsf{Ay}], \mathsf{msg}) = \mathsf{c}$.
6: Return (z, c).

of Simulated sign (Algorithm 14) and the real sign oracle Sign (Algorithm 11).

In this section we prove that Mid-sign (Algorithm 15) and Simulated sign (Algorithm 14) are identical. This fact can be stated in terms of quantum channels as follows. Let Ψ_{midsign} denote the channel described by Algorithm 15. The claim of this section is that $\Psi_{\text{midsign}} = \Psi_{\text{simsign}}$.

To this end, define the following sets for each choice of $\mathsf{c} \in \mathbb{H}$:

$$\text{good}_{\text{simsign}}(\mathsf{c}) := \{\mathsf{z} \in \mathbb{S} : \mathsf{Az} - \mathsf{Tc} \text{ is well-rounded}\} \tag{3.15}$$

$$\text{good}_{\text{midsign}}(\mathsf{c}) := \{\mathsf{y} \in \mathbb{Y} : \mathsf{y} + \mathsf{Sc} \in \mathbb{S} \text{ and } \mathsf{Az} - \mathsf{Tc} \text{ is well-rounded}\}\,. \tag{3.16}$$

We begin with a simple observation on these sets.

Lemma 3.6.1. *The mapping* $f : \mathsf{y} \mapsto \mathsf{y}+\mathsf{Sc}$ *is a bijection from* $\text{good}_{midsign}(\mathsf{c})$ *to* $\text{good}_{simsign}(\mathsf{c})$ *with inverse* $f^{-1} : \mathsf{z} \mapsto \mathsf{z} - \mathsf{Sc}$.

Proof. It is clear that $f^{-1}f$ is the identity function on $\text{good}_{\text{midsign}}(\mathsf{c})$. It remains to prove the following:

1. For each $\mathsf{y} \in \text{good}_{\text{midsign}}(\mathsf{c})$ it holds that $f(\mathsf{y}) \in \text{good}_{\text{simsign}}(\mathsf{c})$.
2. For each $\mathsf{z} \in \text{good}_{\text{simsign}}(\mathsf{c})$ it holds that $f^{-1}(\mathsf{z}) \in \text{good}_{\text{midsign}}(\mathsf{c})$.

To prove item 1 we must show (i) $f(\mathsf{y}) \in \mathbb{S}$, and (ii) $\mathsf{A}f(\mathsf{y}) - \mathsf{Tc}$ is well-rounded. Both items are immediate from the definitions of $\text{good}_{\text{midsign}}(\mathsf{c})$ and f.

To prove item 2 we must show (i) $f^{-1}(\mathsf{z}) \in \mathbb{Y}$, and (ii) $\mathsf{A}f^{-1}(\mathsf{z}) - \mathsf{Ec}$ is well-rounded. Item (i) follows from the fact that z is $(B-U)$-short and Sc is U-short. Item (ii) is immediate from the definitions of $\text{good}_{\text{simsign}}(\mathsf{c})$ and f^{-1}. □

We now prove this section's claim.

Proposition 3.6.1 (Equivalence of Mid-sign and Simulated sign)**.** *The observable behaviour of Mid-sign (Algorithm 15) is statistically identical to that of Simulated sign (Algorithm 14). In terms of quantum channels, we have* $\Psi_{midsign} = \Psi_{simsign}$.

Proof. The observable effects of both the Simulated sign and Mid-sign algorithms can be summarized as follows. Given a message msg as input, the algorithm selects (i) a signature $(\mathsf{z}_{\mathsf{msg}}, \mathsf{c}_{\mathsf{msg}})$ as output, and (ii) a vector $\mathsf{w}_{\mathsf{msg}}$ inducing a hash input $(\mathsf{w}_{\mathsf{msg}}, \mathsf{msg})$ upon which the hash oracle is re-programmed. Thus, to establish statistical equivalence betwen Simulated sign and Mid-sign it suffices to prove that, for each choice of message msg, the joint distribution over $(\mathsf{z}_{\mathsf{msg}}, \mathsf{c}_{\mathsf{msg}}, \mathsf{w}_{\mathsf{msg}})$ induced by each algorithm is identical.

Fix an arbitrary message msg and let $(Z_{\text{simsign}}, C_{\text{simsign}}, W_{\text{simsign}})$, $(Z_{\text{midsign}}, C_{\text{midsign}}, W_{\text{midsign}})$ denote the joint random variables representing the observable behaviour of Simulated sign

and Mid-sign, respectively, on input message msg. We argue that the joint random variables $(Z_{\text{simsign}}, C_{\text{simsign}})$, $(Z_{\text{midsign}}, C_{\text{midsign}})$ are identical. The proposition will then follow from the observation that the hash input w to be re-programmed is specified in both algorithms by the same deterministic function of (z, c). Specifically, in Simulated sign we have $\mathsf{w} = [\mathsf{Az} - \mathsf{Tc}]$, whereas in Mid-sign we have $\mathsf{w} = [\mathsf{Ay}]$. Since $\mathsf{Ay} - \mathsf{Ec}$ is well-rounded, we have

$$\mathsf{w} = [\mathsf{Ay}] = [\mathsf{Ay} - \mathsf{Ec}] = [\mathsf{A}(\mathsf{y} + \mathsf{Sc}) - \mathsf{Tc}] = [\mathsf{Az} - \mathsf{Tc}] \tag{3.17}$$

as desired.

To begin, we argue that

$$\Pr\left[Z_{\text{midsign}} = \mathsf{z} \mid C_{\text{midsign}} = \mathsf{c}\right] = \Pr\left[Z_{\text{simsign}} = \mathsf{z} \mid C_{\text{simsign}} = \mathsf{c}\right] \tag{3.18}$$

for each choice of $\mathsf{c} \in \mathbb{H}$. In Simulated sign, conditioned on a choice of c, the vector z is chosen uniformly among those $\mathsf{z} \in \text{good}_{\text{simsign}}(\mathsf{c})$. In Mid-sign, conditioned on a choice of c, the vector y is chosen uniformly among those $\mathsf{y} \in \text{good}_{\text{midsign}}(\mathsf{c})$ and the vector z is computed as $\mathsf{z} \leftarrow \mathsf{y} + \mathsf{Sc}$. It follows from Lemma 3.6.1 that z is uniform on $\text{good}_{\text{simsign}}(\mathsf{c})$, as desired.

Next, we argue that

$$\Pr\left[C_{\text{midsign}} = \mathsf{c}\right] = \Pr\left[C_{\text{simsign}} = \mathsf{c}\right] \tag{3.19}$$

for each choice of $\mathsf{c} \in \mathbb{H}$, from which it follows that the joint random variables $(Z_{\text{simsign}}, C_{\text{simsign}})$, $(Z_{\text{midsign}}, C_{\text{midsign}})$ are identical. It follows from Lemma 3.6.1 that $\#\,\text{good}_{\text{midsign}}(\mathsf{c}) = \#\,\text{good}_{\text{simsign}}(\mathsf{c})$ for each $c \in \mathbb{H}$. Thus,

$$\Pr\left[C_{\text{midsign}} = \mathsf{c}\right] = \frac{\#\,\text{good}_{\text{midsign}}(\mathsf{c})}{\sum_{\mathsf{c}'} \#\,\text{good}_{\text{midsign}}(\mathsf{c}')} = \frac{\#\,\text{good}_{\text{simsign}}(\mathsf{c})}{\sum_{\mathsf{c}'} \#\,\text{good}_{\text{simsign}}(\mathsf{c}')} = \Pr\left[C_{\text{simsign}} = \mathsf{c}\right] \tag{3.20}$$

as desired. □

3.6.4 Consistent-Mid-Sign

Broadly speaking, Mid-sign (Algorithm 15) behaves like Sign (Algorithm 11) except that c is selected freshly at random instead of according to some hash oracle $H(\cdot)$. It is tempting to claim that the only difference between Mid-sign and Sign is that repeated invocations of Sign always use the same hash oracle $H(\cdot)$, whereas each invocation of Mid-sign switches

to another hash oracle $H'(\cdot)$ that differs from $H(\cdot)$ on a small number of randomly selected inputs.

However, there is a small probability that the random choices in a given execution of Mid-sign are not consistent with *any* hash oracle. To understand how such an inconsistency can occur, observe that each candidate (y, c) selected by Mid-sign induces an associated claim about the underlying hash oracle: that $H([\mathsf{Ay}], \mathsf{msg}) = \mathsf{c}$. Suppose Mid-sign rejects one candidate pair (y, c) because $\mathsf{Ay} - \mathsf{Ec}$ is not well-rounded before finally accepting another candidate pair $(\mathsf{y}', \mathsf{c}')$. If $[\mathsf{Ay}] = [\mathsf{Ay}']$ but $c \neq c'$ then these two candidates represent conflicting claims about the underlying hash oracle.

To address this problem we present a new sign oracle Consistent-mid-sign in Algorithm 16 and argue that its observable behaviour is negligibly close to that of Mid-sign (Algorithm 15). This fact can be stated in terms of quantum channels as follows. Let Ψ_{cmidsign} denote the channel described by Algorithm 16. The claim of this section is that $\Psi_{\text{cmidsign}} \approx \Psi_{\text{midsign}}$, meaning that $\Psi_{\text{cmidsign}}(\rho) \approx \Psi_{\text{midsign}}(\rho)$ for all input states ρ.

Algorithm 16 Consistent-mid-sign

Input: Message msg, public key (A, T), secret key (S, E).
Output: Signature (z, c).

1: Initialize the dictionary $\mathcal{A} \subset (\mathbb{W} \mapsto \mathbb{H})$ to the empty dictionary $\mathcal{A} = \emptyset$.
2: Choose $\mathsf{y} \in \mathbb{Y}$ uniformly at random.
3: **if** $[\mathsf{Ay}] \in \mathcal{A}$ **then**
4: $\quad \mathsf{c} \leftarrow \mathcal{A}[[\mathsf{Ay}]]$
5: **else**
6: $\quad$ choose $\mathsf{c} \in \mathbb{H}$ uniformly at random
7: $\quad$ add $\mathcal{A}[[\mathsf{Ay}]] \leftarrow \mathsf{c}$ to the dictionary $\mathcal{A}$.
8: **end if**
9: $\mathsf{z} \leftarrow \mathsf{y} + \mathsf{Sc}$.
10: If $\mathsf{z} \notin \mathbb{S}$ then retry at step 2.
11: If $\mathsf{Ay} - \mathsf{Ec}$ is not well-rounded then retry at step 2.
12: Re-program the hash oracle $H(\cdot)$ so that $H([\mathsf{Ay}], \mathsf{msg}) = \mathsf{c}$.
13: Return (z, c).

The only difference between Consistent-mid-sign and Mid-sign is that each invocation of Consistent-mid-sign remembers the random candidate pairs it selected throughout the invocation and alters them as needed so as to maintain consistency with a hash oracle. Thus, in order to prove $\Psi_{\text{cmidsign}} \approx \Psi_{\text{midsign}}$ it suffices to prove that only a negligibly small

fraction of the random choices made by Mid-sign lead to an inconsistency that is corrected in Consistent-mid-sign.

A sequence $r = \{(\mathsf{y}_i, \mathsf{c}_i)\}_{i=1}^{\infty}$ of random choices made by Mid-sign leads to an inconsistently derived signature only if there exists $k \geq 2$ such that the following conditions hold:

1. $\mathsf{Ay}_1 - \mathsf{Ec}_1, \ldots, \mathsf{Ay}_{k-1} - \mathsf{Ec}_{k-1}$ are not well-rounded.
2. $\mathsf{Ay}_k - \mathsf{Ec}_k$ is well-rounded.
3. $[\mathsf{Ay}_k] \in \{[\mathsf{Ay}_1], \ldots, [\mathsf{Ay}_{k-1}]\}$.

Consider the event that a random sequence r meets conditions 1–3 for some choice of $k \geq 2$, and let inconsistent(r) denote the infinite disjunction of these events over all $k \geq 2$.[3] We seek an upper bound on the probability of event inconsistent(r) over the choice of r. To this end, define the following quantities for each choice of TESLA keys (A, T), (S, E):

$\mathrm{nwr}(\mathsf{A}, \mathsf{E})$:	The probability over $(\mathsf{y}, \mathsf{c}) \in \mathbb{Y} \times \mathbb{H}$ that $\mathsf{Ay} - \mathsf{Ec}$ is not well-rounded.
$\mathrm{coll}(\mathsf{A}, \mathsf{E})$:	The maximum over all $\mathsf{w} \in \mathbb{W}$ of the probability over $(\mathsf{y}, \mathsf{c}) \in \mathbb{Y} \times \mathbb{H}$ that $[\mathsf{Ay} - \mathsf{Ec}] = \mathsf{w}$.

In symbols, these quantities are written

$$\mathrm{nwr}(\mathsf{A}, \mathsf{E}) := \Pr_{(\mathsf{y},\mathsf{c}) \in \mathbb{Y} \times \mathbb{H}} [\mathsf{Ay} - \mathsf{Ec} \text{ not well-rounded}] \tag{3.21}$$

$$\mathrm{coll}(\mathsf{A}, \mathsf{E}) := \max_{\mathsf{w} \in \mathbb{W}} \left\{ \Pr_{(\mathsf{y},\mathsf{c}) \in \mathbb{Y} \times \mathbb{H}} [[\mathsf{Ay} - \mathsf{Ec}] = \mathsf{w}] \right\} \tag{3.22}$$

In Sections 3.6.8 and 3.6.9 we prove bounds on these quantities that hold with high probability over the choice of TESLA keys (A, T), (S, E).

Broadly speaking, $\mathrm{nwr}(\mathsf{A}, \mathsf{E})$ should be viewed as a constant that's noticeably smaller than 1 — for example, $\mathrm{nwr}(\mathsf{A}, \mathsf{E}) = 1/2$. By contrast $\mathrm{coll}(\mathsf{A}, \mathsf{E})$ is negligibly small. We prove the following.

[3] In item 3 it suffices to look for a collision only between $[\mathsf{Ay}_k]$ and any previous $[\mathsf{Ay}_i]$; we don't need to look for a collision among arbitrary $[\mathsf{Ay}_i] = [\mathsf{Ay}_j]$. This is because such a collision among the bad entries is statistically identical to as if $\mathsf{c}_i = \mathsf{c}_j$. Namely, $[\mathsf{Ay}_i]$ is rejected regardless of whether $\mathsf{c}_i = \mathsf{c}_j$. If however, c_j changes $[\mathsf{Ay}_i]$ from bad to good then that difference will be detected at $k = j$. Thus, there's no need to check for this when $k > j$.

Proposition 3.6.2 (Probability of inconsistency)**.** *For each choice of TESLA keys* (A, T), (S, E) *it holds that*

$$\Pr_r\left[\mathit{inconsistent}(r)\right] \leq \mathrm{coll}(\mathsf{A},\mathsf{E})\frac{\mathrm{nwr}(\mathsf{A},\mathsf{E})}{\left(1-\mathrm{nwr}(\mathsf{A},\mathsf{E})\right)^2}. \tag{3.23}$$

Proof. For each $k \geq 2$ the probability with which events 1 and 2 hold is

$$\mathrm{nwr}(\mathsf{A},\mathsf{E})^{k-1}(1-\mathrm{nwr}(\mathsf{A},\mathsf{E})). \tag{3.24}$$

Conditioned on those events, the probability of event 3 is

$$\Pr_{\substack{(\mathsf{y}_1,\mathsf{c}_1),\ldots,(\mathsf{y}_k,\mathsf{c}_k)\notin \mathrm{WR}(\mathsf{A},\mathsf{E})\\(\mathsf{y}_k,\mathsf{c}_k)\in\mathrm{WR}(\mathsf{A},\mathsf{E})}}\left[\bigvee_{i=1}^{k-1}[\mathsf{A}\mathsf{y}_k]=[\mathsf{A}\mathsf{y}_i]\right] \tag{3.25}$$

$$\leq (k-1)\max_{\mathsf{w}\in\mathbb{W}}\left\{\Pr_{(\mathsf{y},\mathsf{c})\in\mathrm{WR}(\mathsf{A},\mathsf{E})}\left[[\mathsf{A}\mathsf{y}]=\mathsf{w}\right]\right\} \tag{3.26}$$

$$\leq (k-1)\frac{\max_{\mathsf{w}\in\mathbb{W}}\left\{\Pr_{(\mathsf{y},\mathsf{c})\in\mathbb{Y}\times\mathbb{H}}\left[[\mathsf{A}\mathsf{y}-\mathsf{E}\mathsf{c}]=\mathsf{w}\right]\right\}}{\Pr_{(\mathsf{y},\mathsf{c})\in\mathbb{Y}\times\mathbb{H}}\left[\mathsf{A}\mathsf{y}-\mathsf{E}\mathsf{c}\text{ is well-rounded}\right]} \tag{3.27}$$

$$= (k-1)\frac{\mathrm{coll}(\mathsf{A},\mathsf{E})}{1-\mathrm{nwr}(\mathsf{A},\mathsf{E})}. \tag{3.28}$$

(Here we have used the notation $(\mathsf{y},\mathsf{c}) \in \mathrm{WR}(\mathsf{A},\mathsf{E})$ to mean that $\mathsf{A}\mathsf{y}-\mathsf{E}\mathsf{c}$ is well-rounded.) Thus,

$$\Pr_r\left[\text{inconsistent}(r)\right] \leq \sum_{k=2}^{\infty}\mathrm{nwr}(\mathsf{A},\mathsf{E})^{k-1}(1-\mathrm{nwr}(\mathsf{A},\mathsf{E}))(k-1)\frac{\mathrm{coll}(\mathsf{A},\mathsf{E})}{1-\mathrm{nwr}(\mathsf{A},\mathsf{E})} \tag{3.29}$$

$$= \mathrm{coll}(\mathsf{A},\mathsf{E})\sum_{k=2}^{\infty}(k-1)\,\mathrm{nwr}(\mathsf{A},\mathsf{E})^{k-1}. \tag{3.30}$$

The proposition then follows from the formula for the derivative of a geometric progression. □

An immediate corollary of Proposition 3.6.2 is that

$$\left\|\Psi_{\mathrm{cmidsign}}(\rho)-\Psi_{\mathrm{midsign}}(\rho)\right\|_{\mathrm{Tr}} < 2\Pr_r\left[\text{inconsistent}(r)\right] \tag{3.31}$$

for all states ρ.

3.6.5 Consistent-Mid-Sign is a Mixture of Real Sign Oracles

In the previous section we introduced the sign oracle Consistent-mid-sign (Algorithm 16) and claimed that it behaves exactly like Sign (Algorithm 11) with the following exception: repeated invocations of Sign always use the same hash oracle $H(\cdot)$, whereas each invocation of Consistent-mid-sign switches to another hash oracle $H'(\cdot)$ that differs from $H(\cdot)$ on a small fraction of randomly selected inputs.

Let us formalize this claim. Unfortunately, we must introduce some cumbersome notation. For each message msg define the symbols

$\mathsf{y}_{\mathsf{msg}}$: A sequence $\{\mathsf{y}_{\mathsf{msg},i}\}_{i=1}^{\infty}$ of elements drawn randomly from $\mathbb{Y}$.
$\mathsf{c}_{\mathsf{msg}}(\mathsf{y}_{\mathsf{msg}})$: A sequence $\{\mathsf{c}_{\mathsf{msg},i}\}_{i=1}^{\infty}$ of elements drawn randomly from $\mathbb{H}$ subject to the constraint that if $[\mathsf{Ay}_{\mathsf{msg},i}] = [\mathsf{Ay}_{\mathsf{msg},j}]$ then $\mathsf{c}_{\mathsf{msg},i} = \mathsf{c}_{\mathsf{msg},j}$.

The output of Consistent-mid-sign on input msg is a deterministic function of the random data $\mathsf{y}_{\mathsf{msg}}, \mathsf{c}_{\mathsf{msg}}(\mathsf{y}_{\mathsf{msg}})$. Specifically, let $k(\mathsf{msg})$ denote the minimum index for which $\mathsf{Ay}_{\mathsf{msg},k(\mathsf{msg})} - \mathsf{Ec}_{\mathsf{msg},k(\mathsf{msg})}$ is well-rounded. Then Consistent-mid-sign outputs the signature $(\mathsf{y}_{\mathsf{msg},k(\mathsf{msg})} + \mathsf{Sc}_{\mathsf{msg},k(\mathsf{msg})}, \mathsf{c}_{\mathsf{msg},k(\mathsf{msg})})$. For shorthand, write $\tau_{\mathsf{msg}} = (\mathsf{y}_{\mathsf{msg}}, \mathsf{c}_{\mathsf{msg}}(\mathsf{y}_{\mathsf{msg}}))$.

Let $\mathsf{y} = \{\mathsf{y}_{\mathsf{msg}}\}_{\mathsf{msg}}$ and $\mathsf{c}(\mathsf{y}) = \{\mathsf{c}_{\mathsf{msg}}(\mathsf{y}_{\mathsf{msg}})\}_{\mathsf{msg}}$ denote selections of random data for each possible message msg. For shorthand, write $\tau = (\mathsf{y}, \mathsf{c}(\mathsf{y}))$ so that the behaviour of Consistent-mid-sign on all inputs is completely specified by τ.

For each choice of hash oracle $H(\cdot)$ and random data τ consider the hash oracle $H_\tau(\cdot)$ that agrees with $H(\cdot)$ everywhere except that $H_\tau([\mathsf{Ay}_{\mathsf{msg},i}], \mathsf{msg}) = \mathsf{c}_{\mathsf{msg},i}$ for each message msg and each $i = 1, \ldots, k(\mathsf{msg})$. In other words,

$$H_\tau(\mathsf{w}, \mathsf{msg}) = \begin{cases} \mathsf{c}_{\mathsf{msg},i} & \text{if } \mathsf{w} = [\mathsf{Ay}_{\mathsf{msg},i}] \text{ for some } i \in \{1, \ldots, k(\mathsf{msg})\} \\ H(\mathsf{w}, \mathsf{msg}) & \text{otherwise} \end{cases} . \tag{3.32}$$

The behaviour of Sign (Algorithm 11) with hash oracle $H_\tau(\cdot)$ on all inputs is completely specified by y and $H_\tau(\cdot)$. Moreover, the behaviour of Sign with hash oracle $H_\tau(\cdot)$ and random data y is *identical* to the behaviour of Consistent-mid-sign with random data τ.

For each choice of random data τ define the following quantum channels:

$\Psi_{\mathrm{cmidsign},\tau}$: The quantum channel representing the actions of Consistent-mid-sign (Algorithm 16) with randomness τ.
$\Psi_{\mathrm{real},H_\tau,\mathsf{y}}$: The quantum channel representing the actions of Sign (Algorithm 11) with hash oracle $H_\tau(\cdot)$ and randomness y.

The previous observations establish

$$\Psi_{\text{cmidsign},\tau} = \Psi_{\text{real},H_\tau,\mathsf{y}} \tag{3.33}$$

for each choice of τ. Because Ψ_{cmidsign} is simply a uniform mixture of channels $\Psi_{\text{cmidsign},\tau}$, it follows that[4]

$$\Psi_{\text{cmidsign}} = \sum_{\tau} \Pr[\tau]\, \Psi_{\text{real},H_\tau,\mathsf{y}}. \tag{3.34}$$

3.6.6 Re-Programming of Hash Oracles is Hard to Detect

Thus far we have proved that Consistent-mid-sign behaves like a mixture of real sign oracles *when viewed in isolation*. That is, for all states ρ we have

$$\Psi_{\text{cmidsign}}(\rho) = \sum_{\tau} \Pr[\tau]\, \Psi_{\text{real},H_\tau,\mathsf{y}}(\rho). \tag{3.35}$$

But we must extend this proof so that it holds even in the presence of independent information on the underlying hash oracle. In particular, for any hash oracle $H(\cdot)$ and any state ρ_H prepared using only a tractable number of queries to $H(\cdot)$ we must show that

$$\Psi_{\text{cmidsign}}(\rho_H) \approx \sum_{\tau} \Pr[\tau]\, \Psi_{\text{real},H_\tau,\mathsf{y}}(\rho_{H_\tau}). \tag{3.36}$$

To establish this claim it suffices to show that $\rho_H \approx \rho_{H_\tau}$ with high probability over the choice of τ.

This claim is proven by an application of the BBBV Theorem [8, Theorem 3.3], so let us introduce the formalism necessary to state this theorem. Suppose ρ_H was prepared by some party $\mathcal{R}$ using t queries to some hash oracle $H : X \to Y$. For each $i = 1, \ldots, t$ let ρ_i denote the state of $\mathcal{R}$'s system immediately prior to the ith query to $H(\cdot)$. For each hash input $x \in X$ let

$$\mathcal{Q}_{\mathcal{R}(H)}(x) := \sum_{i=1}^{t} \text{Tr}\left(|x\rangle\langle x|\rho_i\right) \tag{3.37}$$

denote the *query magnitude* on input x for $\mathcal{R}$'s interaction with hash oracle $H(\cdot)$. The BBBV Theorem (or rather, a consequence of it) is as follows.

[4] Strictly speaking, $\Pr[\tau]$ is zero because it represents the uniform distribution over a countably infinite set. We should switch to a probability measure on τ and use an integral instead of a summation over τ.

Theorem 3.6.1 ([8, Theorem 3.3])**.** *The following holds for each $\epsilon > 0$. Suppose ρ_H was prepared by some party $\mathcal{R}$ using t queries to some hash oracle $H : X \to Y$. Let $H'(\cdot)$ be a hash oracle that agrees with $H(\cdot)$ except on a subset $X' \subset X$ of inputs with the property that*

$$\sum_{x \in X'} \mathcal{Q}_{\mathcal{R}(H)}(x) \leq \frac{\epsilon^2}{t}. \tag{3.38}$$

Let $\rho_{H'}$ be the state prepared when $\mathcal{R}$ uses hash oracle $H'(\cdot)$ instead of $H(\cdot)$. It holds that $\|\rho_{H'} - \rho_H\|_{\mathrm{Tr}} \leq \epsilon$.

We are now ready to prove the claim of this section.

Proposition 3.6.3 (Re-Programming in TESLA)**.** *The following holds for each choice of TESLA keys (A, T), (S, E) and each $\delta > 0$.*

Suppose ρ_H was prepared by some party $\mathcal{D}$ using t queries to hash oracle $H(\cdot)$. Let τ be random data and let $H_\tau(\cdot)$ be a hash oracle derived from $H(\cdot)$ and τ as described in Section 3.6.5. Let $\rho_{H'}$ be the state prepared when $\mathcal{D}$ uses hash oracle $H'(\cdot)$ instead of $H(\cdot)$.

Then $\|\rho_{H_\tau} - \rho_H\|_{\mathrm{Tr}} < \delta$ except with probability at most

$$\frac{t^2}{\delta^2} \frac{\mathrm{coll}(\mathsf{A}, \mathsf{E})}{1 - \mathrm{nwr}(\mathsf{A}, \mathsf{E})} \tag{3.39}$$

over the choice of τ.

Proof. By Theorem 3.6.1 it suffices to prove that the quantity

$$\sum_{\mathsf{msg}} \sum_{i=1}^{k(\mathsf{msg})} \mathcal{Q}_{\mathcal{D}(H)}\left([\mathsf{Ay}_{\mathsf{msg},i}], \mathsf{msg}\right) \tag{3.40}$$

is at most δ^2/t with high probability over the choice of τ. To this end, for each message msg let

$$X_{\mathsf{msg}} = \{([\mathsf{Ay}], \mathsf{msg}) : \mathsf{y} \in \mathbb{Y}\} \tag{3.41}$$

denote the set of hash inputs for message msg that are candidates for re-programming, and let

$$t_{\mathsf{msg}} = \sum_{x \in X_{\mathsf{msg}}} \mathcal{Q}_{\mathcal{D}(H)}(x) \tag{3.42}$$

denote the total query magnitude for message msg. Observe that $t = \sum_{\mathsf{msg}} t_{\mathsf{msg}}$.

The quantity (3.40) is maximized if for each message msg all the query magnitude t_{msg} alloted to message msg is placed on the element $\mathsf{w} \in \mathbb{W}$ most likely to collide with $[\mathsf{Ay}]$ when $\mathsf{y} \in \mathbb{Y}$ is chosen uniformly at random. In this case, the quantity (3.40) is at most

$$\sum_{\mathsf{msg}} k(\mathsf{msg}) t_{\mathsf{msg}} \operatorname{coll}(\mathsf{A}, \mathsf{E}). \tag{3.43}$$

By Markov's inequality we have

$$\Pr_{\tau}\left[\sum_{\mathsf{msg}} k(\mathsf{msg}) t_{\mathsf{msg}} \operatorname{coll}(\mathsf{A}, \mathsf{E}) \geq \frac{\delta^2}{t}\right] \leq \frac{t}{\delta^2} \operatorname*{Ex}_{\tau}\left[\sum_{\mathsf{msg}} k(\mathsf{msg}) t_{\mathsf{msg}} \operatorname{coll}(\mathsf{A}, \mathsf{E})\right] \tag{3.44}$$

$$= \frac{t}{\delta^2} \operatorname{coll}(\mathsf{A}, \mathsf{E}) \sum_{\mathsf{msg}} t_{\mathsf{msg}} \operatorname*{Ex}_{\tau_{\mathsf{msg}}}[k(\mathsf{msg})] \tag{3.45}$$

Thus, it suffices to bound the expected number $k(\mathsf{msg})$ of entries one must view from a given list τ_{msg} before encountering an entry (y, c) for which $\mathsf{Ay} - \mathsf{Ec}$ is well-rounded. We have

$$\operatorname*{Ex}_{\tau_{\mathsf{msg}}}[k(\mathsf{msg})] = \sum_{k=1}^{\infty} k \operatorname{nwr}(\mathsf{A}, \mathsf{E})^{k-1}(1 - \operatorname{nwr}(\mathsf{A}, \mathsf{E})) = \frac{1}{1 - \operatorname{nwr}(\mathsf{A}, \mathsf{E})} \tag{3.46}$$

where the final equality follows from the formula for the derivative of a geometric progression. The proposition follows from $\sum_{\mathsf{msg}} t_{\mathsf{msg}} = t$. □

3.6.7 The Distinguisher's State, Revisited

Recall from Section 3.6.2 the state ρ_H, which is the state of $\mathcal{D}$'s system immediately prior to the sign query in the first block. Let $\kappa_1 \leq q_h$ denote the query magnitude on the live-switch for the hash oracle in the first block. We proved the following in previous sections:

$$\Psi_{\text{simsign}}(\rho_H) = \Psi_{\text{midsign}}(\rho_H) \tag{3.47}$$

$$\|\Psi_{\text{midsign}}(\rho_H) - \Psi_{\text{cmidsign}}(\rho_H)\|_{\text{Tr}} < 2 \operatorname{coll}(\mathsf{A}, \mathsf{E}) \frac{\operatorname{nwr}(\mathsf{A}, \mathsf{E})}{(1 - \operatorname{nwr}(\mathsf{A}, \mathsf{E}))^2} \tag{3.48}$$

$$\Psi_{\text{cmidsign}}(\rho_H) = \sum_{\tau} \Pr[\tau] \Psi_{\text{real},H_\tau,\mathsf{y}}(\rho_H) \tag{3.49}$$

$$\Pr_{\tau}[\|\rho_{H_\tau} - \rho_H\|_{\text{Tr}} > \epsilon] < \frac{\kappa_1^2}{\epsilon^2} \frac{\operatorname{coll}(\mathsf{A}, \mathsf{E})}{1 - \operatorname{nwr}(\mathsf{A}, \mathsf{E})} \tag{3.50}$$

We conclude that

$$\left\| \Psi_{\text{simsign}}(\rho_H) - \sum_{\tau} \Pr[\tau]\, \Psi_{\text{real},H_\tau,\mathsf{y}}(\rho_{H_\tau}) \right\|_{\text{Tr}} < \delta(\kappa_1) \tag{3.51}$$

where[5]

$$\delta(\kappa) := 2\,\text{coll}(\mathsf{A},\mathsf{E})\frac{\text{nwr}(\mathsf{A},\mathsf{E})}{(1-\text{nwr}(\mathsf{A},\mathsf{E}))^2} + \epsilon + \frac{\kappa^2}{\epsilon^2}\frac{\text{coll}(\mathsf{A},\mathsf{E})}{1-\text{nwr}(\mathsf{A},\mathsf{E})}. \tag{3.52}$$

That is, the state of $\mathcal{D}$'s system at the end of the first block of an interaction with simulated oracles is $\delta(\kappa_1)$-close to a probabilistic mixture over states of $\mathcal{D}$'s system, each of which could have been obtained from an interaction with real hash oracles.

As suggested in Section 3.6.2, we continue inductively throughout the q_s blocks. As with κ_1, let $\kappa_2, \ldots, \kappa_{q_s}$ denote the query magnitude on the live-switch for blocks two through q_s. Define

$$\delta_{\text{yes}} = \sum_{i=1}^{q_s} \delta(\kappa_i) \tag{3.53}$$

and observe that the state of $\mathcal{D}$'s system at the end of an interaction with simulated oracles is δ_{yes}-close to a probabilistic mixture over states obtained from an interaction with real hash oracles.

Let us compute an upper bound on δ_{yes}. Because each block includes information from hash queries in previous blocks plus one additional hash query learned from the sign oracle, we have

$$\kappa_{i+1} \geq \kappa_i + 1. \tag{3.54}$$

Because $\mathcal{D}$ is permitted at most q_h total query magnitude on the live-switch for its hash queries, we have

$$\kappa_i \leq q_h + i - 1. \tag{3.55}$$

It is clear that δ_{yes} is maximized when $\kappa_1 = q_h$, which corresponds to a distinguisher who makes all q_h hash queries before making any of the q_s sign queries. Taking a loose bound $q_h + q_s$ for each κ_i, we obtain

$$\delta_{\text{yes}} < q_s \left(2\,\text{coll}(\mathsf{A},\mathsf{E})\frac{\text{nwr}(\mathsf{A},\mathsf{E})}{(1-\text{nwr}(\mathsf{A},\mathsf{E}))^2} + \epsilon + \frac{(q_h+q_s)^2}{\epsilon^2}\frac{\text{coll}(\mathsf{A},\mathsf{E})}{1-\text{nwr}(\mathsf{A},\mathsf{E})} \right) \tag{3.56}$$

[5] The final two terms of (3.52) are due to the fact that a $\frac{\kappa^2}{\epsilon^2}\frac{\text{nwr}(\mathsf{A},\mathsf{E})}{(1-\text{nwr}(\mathsf{A},\mathsf{E}))^2}$-fraction of τ lead to a hash oracle $H_\tau(\cdot)$ for which $\|\rho_H - \rho_{H_\tau}\|_{\text{Tr}} > \epsilon$, in which case we assume that ρ_H, ρ_{H_τ} are perfectly distinguishable. For all other τ it holds that ρ_H, ρ_{H_τ} are ϵ-close.

Finally, because the real and simulated oracles are δ_{yes}-close, it follows that

$$\Pr\left[\mathcal{S} \text{ output "yes"} \mid (\mathsf{A},\mathsf{T}) \text{ yes-instance of LWE}\right] > \Pr\left[\text{forge}(\mathsf{A},\mathsf{T})\right] - \delta_{\text{yes}} \tag{3.57}$$

as desired.

3.6.8 Probability of Well-Roundedness

Let ϕ denote the probability that a random vector in $\mathbb{Z}_q^m$ is not well-rounded:

$$\phi := \Pr_{x \in \mathbb{Z}_q^m}\left[x \text{ not well-rounded}\right] \leq m\left(\frac{2L}{2^d} + \frac{2L}{q}\right). \tag{3.58}$$

The quantity ϕ is a function of the Tesla parameters q, m, d, L. It is a constant that's noticeably smaller than 1.

Recall the definition of $\text{nwr}(\mathsf{A},\mathsf{E})$: for TESLA keys (A,T), (S,E) define $\text{nwr}(\mathsf{A},\mathsf{E})$ as the probability over $(\mathsf{y},\mathsf{c}) \in \mathbb{Y} \times \mathbb{H}$ that $\mathsf{Ay} - \mathsf{Ec}$ is not well-rounded:

$$\text{nwr}(\mathsf{A},\mathsf{E}) := \Pr_{(\mathsf{y},\mathsf{c}) \in \mathbb{Y}\times\mathbb{H}}\left[\mathsf{Ay} - \mathsf{Ec} \text{ not well-rounded}\right]. \tag{3.59}$$

We prove the following.

Lemma 3.6.2 (Probability of well-roundedness)**.** *The following holds for all $K > 0$. With probability $1 - 1/K$ over the choice of TESLA keys (A,T), (S,E) it holds that*

$$\text{nwr}(\mathsf{A},\mathsf{E}) \leq \phi + \sqrt{\frac{K(q+1)}{\#\mathbb{Y}}}. \tag{3.60}$$

Proof. Our strategy is to bound the variance of $\text{nwr}(\mathsf{A},\mathsf{E})$ over the choice of TESLA keys (A,T), (S,E) and use Chebyshev's inequality. By definition,

$$\operatorname*{Var}_{(\mathsf{A},\mathsf{E})}\left[\text{nwr}(\mathsf{A},\mathsf{E})\right] = \operatorname*{Ex}_{(\mathsf{A},\mathsf{E})}\left[\text{nwr}(\mathsf{A},\mathsf{E})^2\right] - \operatorname*{Ex}_{(\mathsf{A},\mathsf{E})}\left[\text{nwr}(\mathsf{A},\mathsf{E})\right]^2 \tag{3.61}$$

so it suffices to compute the expectation of $\text{nwr}(\mathsf{A},\mathsf{E})$ and an upper bound on the expectation of $\text{nwr}(\mathsf{A},\mathsf{E})^2$.

We begin by computing the expectation of $\mathrm{nwr}(\mathsf{A},\mathsf{E})$. We have

$$\begin{aligned}
&\operatorname*{Ex}_{(\mathsf{A},\mathsf{E})}\left[\mathrm{nwr}(\mathsf{A},\mathsf{E})\right] && (3.62)\\
&= \sum_{(\mathsf{A},\mathsf{E})} \Pr[(\mathsf{A},\mathsf{E})] \frac{1}{\#\mathbb{Y}\#\mathbb{H}} \sum_{(\mathsf{y},\mathsf{c})} \mathrm{bool}\left[\mathsf{Ay}-\mathsf{Ec} \text{ not well-rounded}\right] && (3.63)\\
&= \frac{1}{\#\mathbb{Y}\#\mathbb{H}} \sum_{(\mathsf{y},\mathsf{c})} \sum_{(\mathsf{A},\mathsf{E})} \Pr[(\mathsf{A},\mathsf{E})]\, \mathrm{bool}\left[\mathsf{Ay}-\mathsf{Ec} \text{ not well-rounded}\right] && (3.64)\\
&= \frac{1}{\#\mathbb{Y}\#\mathbb{H}} \sum_{(\mathsf{y},\mathsf{c})} \Pr_{(\mathsf{A},\mathsf{E})}\left[\mathsf{Ay}-\mathsf{Ec} \text{ not well-rounded}\right]. && (3.65)
\end{aligned}$$

(Here we have used the notation $\mathrm{bool}\,[s]$ for any statement s that can be either true or false to mean that $\mathrm{bool}\,[s]=1$ if the statement is true and $\mathrm{bool}\,[s]=0$ otherwise.) So we need to bound the probability

$$\Pr_{(\mathsf{A},\mathsf{E})}\left[\mathsf{Ay}-\mathsf{Ec} \text{ not well-rounded}\right] \tag{3.66}$$

for each fixed choice of $(\mathsf{y},\mathsf{c}) \in \mathbb{Y}\times\mathbb{H}$. There are two cases:

1. If $\mathsf{y}\neq 0$ then Ay is a uniformly random vector in $\mathbb{Z}_q^m$. So too is $\mathsf{Ay}-\mathsf{Ec}$, since c is fixed and E is independent of A. In this case, the probability (3.66) equals ϕ.
2. If $\mathsf{y}=0$ then the probability (3.66) equals 0, since $-\mathsf{Ec}$ is well-rounded for all E,c.

Case 2 occurs with probability $1/\#\mathbb{Y}$, from which it follows that

$$\operatorname*{Ex}_{(\mathsf{A},\mathsf{E})}\left[\mathrm{nwr}(\mathsf{A},\mathsf{E})\right] = \left(1-\frac{1}{\#\mathbb{Y}}\right)\phi. \tag{3.67}$$

Next, we compute an upper bound on the expectation of $\mathrm{nwr}(\mathsf{A},\mathsf{E})^2$. Similar to the above, we have

$$\operatorname*{Ex}_{(\mathsf{A},\mathsf{E})}\left[\mathrm{nwr}(\mathsf{A},\mathsf{E})^2\right] = \frac{1}{\left(\#\mathbb{Y}\#\mathbb{H}\right)^2} \sum_{(\mathsf{y},\mathsf{c}),(\mathsf{y}',\mathsf{c}')} \Pr_{(\mathsf{A},\mathsf{E})}\left[\mathsf{Ay}-\mathsf{Ec},\mathsf{Ay}'-\mathsf{Ec}' \text{ not well rounded}\right] \tag{3.68}$$

and so we need to bound the probability

$$\Pr_{(\mathsf{A},\mathsf{E})}\left[\mathsf{Ay}-\mathsf{Ec},\mathsf{Ay}'-\mathsf{Ec}' \text{ not well-rounded}\right] \tag{3.69}$$

for each fixed choice of $(\mathsf{y},\mathsf{c}),(\mathsf{y}',\mathsf{c}') \in \mathbb{Y}\times\mathbb{H}$. There are two cases:

1. If y, y' are nonzero and linearly independent then $\mathsf{Ay}, \mathsf{Ay}'$ are uniformly random vectors in $\mathbb{Z}_q^m$; so too are $\mathsf{Ay} - \mathsf{Ec}, \mathsf{Ay}' - \mathsf{Ec}'$. In this case, the probability (3.69) equals ϕ^2.

2. If y, y' are linearly dependent then the probability (3.69) is at most 1.

Case 2 occurs with probability at most $(q+1)/\#\mathbb{Y}$, from which it follows that

$$\underset{(\mathsf{A},\mathsf{E})}{\mathrm{Ex}}\left[\mathrm{nwr}(\mathsf{A},\mathsf{E})^2\right] \leq \left(1 - \frac{q+1}{\#\mathbb{Y}}\right)\phi^2 + \frac{q+1}{\#\mathbb{Y}} \tag{3.70}$$

Combining these bounds on the expectation of $\mathrm{nwr}(\mathsf{A},\mathsf{E})$ and $\mathrm{nwr}(\mathsf{A},\mathsf{E})^2$ (and employing the inequality $1-(q+1)/\#\mathbb{Y} < (1-1/\#\mathbb{Y})^2$), we obtain the inequality

$$\underset{(\mathsf{A},\mathsf{E})}{\mathrm{Var}}\left[\mathrm{nwr}(\mathsf{A},\mathsf{E})\right] \leq \frac{q+1}{\#\mathbb{Y}}. \tag{3.71}$$

By Chebyshev's inequality it holds that

$$\Pr_{(\mathsf{A},\mathsf{E})}\left[\left|\mathrm{nwr}(\mathsf{A},\mathsf{E}) - \underset{(\mathsf{A},\mathsf{E})}{\mathrm{Ex}}\left[\mathrm{nwr}(\mathsf{A},\mathsf{E})\right]\right| \geq \sqrt{\frac{K(q+1)}{\#\mathbb{Y}}}\right] \leq \frac{1}{K}. \tag{3.72}$$

The lemma follows from the expression (3.67) for the expectation of $\mathrm{nwr}(\mathsf{A},\mathsf{E})$. □

3.6.9 Probability of Repetition

Let ψ denote the probability that a random vector $\mathsf{x} \in \mathbb{Z}_q^m$ is in $\Delta\mathbb{L}$:

$$\psi := \Pr_{\mathsf{x}\in\mathbb{Z}_q^m}\left[\mathsf{x} \in \Delta\mathbb{L}\right] \leq \left(\frac{2^{d+1}}{q}\right)^m. \tag{3.73}$$

The quantity ψ is a function of the Tesla parameters q, m, d. It is negligibly small.

Recall the definition of $\mathrm{coll}(\mathsf{A},\mathsf{E})$: for TESLA keys (A,T), (S,E) define $\mathrm{coll}(\mathsf{A},\mathsf{E})$ as the maximum over all $\mathsf{w} \in \mathbb{W}$ of the probability over $(\mathsf{y},\mathsf{c}) \in \mathbb{Y}\times\mathbb{H}$ that $[\mathsf{Ay}-\mathsf{Ec}] = \mathsf{w}$:

$$\mathrm{coll}(\mathsf{A},\mathsf{E}) := \max_{\mathsf{w}\in\mathbb{W}}\left\{\Pr_{(\mathsf{y},\mathsf{c})\in\mathbb{Y}\times\mathbb{H}}\left[[\mathsf{Ay}-\mathsf{Ec}] = \mathsf{w}\right]\right\}. \tag{3.74}$$

We prove the following.

Lemma 3.6.3 (Probability of repetition)**.** *The following holds for all $K > 0$. With probability $1 - 1/K$ over the choice of TESLA keys (A, T), (S, E) it holds that*

$$\mathrm{coll}(\mathsf{A}, \mathsf{E}) \leq K\psi. \tag{3.75}$$

Before proving Lemma 3.6.3 let us introduce some notation. Define the set

$$\mathbb{G}(\mathsf{A}, \mathsf{E}) := \{(\mathsf{y}, \mathsf{c}) \in \Delta\mathbb{Y} \times \Delta\mathbb{H} : \mathsf{Ay} - \mathsf{Ec} \in \Delta\mathbb{L}\}. \tag{3.76}$$

Some basic facts about the set $\mathbb{G}(\mathsf{A}, \mathsf{E})$ are listed below in Lemma 3.6.4.

Proof of Lemma 3.6.3. Let $\mathrm{coll}'(\mathsf{A}, \mathsf{E})$ denote the probability over $(\mathsf{y}, \mathsf{c}) \in \Delta\mathbb{Y} \times \Delta\mathbb{H}$ that $(\mathsf{y}, \mathsf{c}) \in \mathbb{G}(\mathsf{A}, \mathsf{E})$:

$$\mathrm{coll}'(\mathsf{A}, \mathsf{E}) := \Pr_{(\mathsf{y},\mathsf{c}) \in \Delta\mathbb{Y} \times \Delta\mathbb{H}} [(\mathsf{y}, \mathsf{c}) \in \mathbb{G}(\mathsf{A}, \mathsf{E})]. \tag{3.77}$$

It follows from Lemma 3.6.4 that $\mathrm{coll}'(\mathsf{A}, \mathsf{E}) \geq \mathrm{coll}(\mathsf{A}, \mathsf{E})$. Thus, it suffices to prove the lemma with $\mathrm{coll}'(\mathsf{A}, \mathsf{E})$ in place of $\mathrm{coll}(\mathsf{A}, \mathsf{E})$.

Observe that $\mathrm{coll}'(\mathsf{A}, \mathsf{E}) \geq 1/\#\Delta\mathbb{Y}$, which follows from the fact that $\#\mathbb{G}(\mathsf{A}, \mathsf{E}) \geq \#\Delta\mathbb{H}$ (Lemma 3.6.4). Our strategy is to bound the expectation of the positive random variable $\mathrm{coll}'(\mathsf{A}, \mathsf{E}) - 1/\#\Delta\mathbb{Y}$ over the choice of TESLA keys (A, T), (S, E) and use Markov's inequality. To this end let us compute the expectation of $\mathrm{coll}'(\mathsf{A}, \mathsf{E})$:

$$\begin{aligned}
\mathop{\mathrm{Ex}}_{(\mathsf{A},\mathsf{E})} [\mathrm{coll}'(\mathsf{A}, \mathsf{E})] &= \sum_{(\mathsf{A},\mathsf{E})} \Pr[(\mathsf{A}, \mathsf{E})] \frac{1}{\#\Delta\mathbb{S}\#\Delta\mathbb{H}} \sum_{(\mathsf{y},\mathsf{c})} \mathrm{bool}[(\mathsf{y}, \mathsf{c}) \in \mathbb{G}(\mathsf{A}, \mathsf{E})] && (3.78)\\
&= \frac{1}{\#\Delta\mathbb{S}\#\Delta\mathbb{H}} \sum_{(\mathsf{y},\mathsf{c})} \sum_{(\mathsf{A},\mathsf{E})} \Pr[(\mathsf{A}, \mathsf{E})] \mathrm{bool}[(\mathsf{y}, \mathsf{c}) \in \mathbb{G}(\mathsf{A}, \mathsf{E})] && (3.79)\\
&= \frac{1}{\#\Delta\mathbb{S}\#\Delta\mathbb{H}} \sum_{(\mathsf{y},\mathsf{c})} \Pr_{(\mathsf{A},\mathsf{E})} [(\mathsf{y}, \mathsf{c}) \in \mathbb{G}(\mathsf{A}, \mathsf{E})]. && (3.80)
\end{aligned}$$

So we need to bound the probability

$$\Pr_{(\mathsf{A},\mathsf{E})} [(\mathsf{y}, \mathsf{c}) \in \mathbb{G}(\mathsf{A}, \mathsf{E})] \tag{3.81}$$

for each fixed choice of $(\mathsf{y}, \mathsf{c}) \in \Delta\mathbb{Y} \times \Delta\mathbb{H}$. There are two cases:

1. If $\mathsf{y} \neq 0$ then Ay is a uniformly random vector in $\mathbb{Z}_q^m$. So too is $\mathsf{Ay} - \mathsf{Ec}$, since c is fixed and E is independent of A. In this case, the probability (3.81) is exactly ψ.

2. If $\mathsf{y} = 0$ then the probability (3.81) is exactly 1.

Case 2 occurs with probability exactly $1/\#\Delta\mathbb{Y}$ over the choice of (y, c). It follows that

$$\operatorname*{Ex}_{(\mathsf{A},\mathsf{E})}[\mathrm{coll}'(\mathsf{A},\mathsf{E})] = \left(1 - \frac{1}{\#\Delta\mathbb{Y}}\right)\psi + \frac{1}{\#\Delta\mathbb{Y}}. \tag{3.82}$$

Then by Markov's inequality we have

$$\Pr_{(\mathsf{A},\mathsf{E})}\left[\mathrm{coll}'(\mathsf{A},\mathsf{E}) \geq K\left(1 - \frac{1}{\#\Delta\mathbb{Y}}\right)\psi\right] \leq \frac{1}{K}. \tag{3.83}$$

That is, with probability at least $1 - 1/K$ over the choice of TESLA keys (A,T), (S,E) it holds that

$$\mathrm{coll}'(\mathsf{A},\mathsf{E}) \leq K\left(1 - \frac{1}{\#\Delta\mathbb{Y}}\right)\psi \tag{3.84}$$

from which the lemma follows. □

Lemma 3.6.4. *For all TESLA keys* (A,T), (S,E) *and all* $\mathsf{w} \in \mathbb{W}$ *it holds that*

$$\#\mathbb{G}(\mathsf{A},\mathsf{E}) \geq \#\Delta\mathbb{H} \tag{3.85}$$

$$\#\mathbb{G}(\mathsf{A},\mathsf{E}) \geq \#\left\{(\mathsf{y},\mathsf{c}) \in \mathbb{Y} \times \mathbb{H} : [\mathsf{A}\mathsf{y} - \mathsf{E}\mathsf{c}] = \mathsf{w}\right\} \tag{3.86}$$

Proof. The inequality (3.85) is straightforward: For each $\mathsf{c}, \mathsf{c}' \in \mathbb{H}$ we have $[\mathsf{E}\mathsf{c}] = [\mathsf{E}\mathsf{c}'] = [0]$, from which it follows that $(0, \mathsf{c} - \mathsf{c}') \in \mathbb{G}(\mathsf{A},\mathsf{E})$.

It remains to prove the inequality (3.86). Let $(\mathsf{y},\mathsf{c}), (\mathsf{y}',\mathsf{c}')$ be elements of $\mathbb{Y} \times \mathbb{H}$ with $[\mathsf{A}\mathsf{y} - \mathsf{E}\mathsf{c}] = [\mathsf{A}\mathsf{y}' - \mathsf{E}\mathsf{c}'] = \mathsf{w}$. We claim that $(\mathsf{y} - \mathsf{y}', \mathsf{c} - \mathsf{c}')$ is in $\mathbb{G}(\mathsf{A},\mathsf{E})$. It is clear that $\mathsf{y} - \mathsf{y}' \in \Delta\mathbb{Y}$ and $\mathsf{c} - \mathsf{c}' \in \Delta\mathbb{H}$. It remains to verify $\mathsf{A}(\mathsf{y} - \mathsf{y}') - \mathsf{E}(\mathsf{c} - \mathsf{c}') \in \Delta\mathbb{L}$. We have

$$\mathsf{A}(\mathsf{y} - \mathsf{y}') - \mathsf{E}(\mathsf{c} - \mathsf{c}') = \mathsf{A}\mathsf{y} - \mathsf{E}\mathsf{c} - (\mathsf{A}\mathsf{y}' - \mathsf{E}\mathsf{c}'). \tag{3.87}$$

Since $\mathsf{A}\mathsf{y} - \mathsf{E}\mathsf{c}$ and $\mathsf{A}\mathsf{y}' - \mathsf{E}\mathsf{c}'$ have the same high bits, it must be that $\mathsf{A}(\mathsf{y} - \mathsf{y}') - \mathsf{E}(\mathsf{c} - \mathsf{c}')$ is the difference of two vectors from $[-(2^{d-1} - 1), 2^{d-1}]^m$, from which it follows that $\mathsf{A}(\mathsf{y} - \mathsf{y}') - \mathsf{E}(\mathsf{c} - \mathsf{c}') \in \Delta\mathbb{L}$.

If $(\mathsf{y}_1,\mathsf{c}_1), \ldots, (\mathsf{y}_k,\mathsf{c}_k)$ are distinct elements of $\mathbb{Y} \times \mathbb{H}$ with $[\mathsf{A}\mathsf{y}_i - \mathsf{E}\mathsf{c}_i] = \mathsf{w}$ for each $i = 1, \ldots, k$ then $(0,0), (\mathsf{y}_1 - \mathsf{y}_2, \mathsf{c}_1 - \mathsf{c}_2), \ldots, (\mathsf{y}_1 - \mathsf{y}_k, \mathsf{c}_1 - \mathsf{c}_k)$ must be distinct elements of $\mathbb{G}(\mathsf{A},\mathsf{E})$[6]. We have thus listed k distinct elements of $\mathbb{G}(\mathsf{A},\mathsf{E})$, from which the lemma follows. □

[6] By contrast with no-instances, we cannot guarantee that the negations are distinct.

3.7 No-Instances of LWE

In this section we prove that the probability

$$\Pr\left[\mathcal{S} \text{ output "yes"} \mid (\mathsf{A}, \mathsf{T}) \text{ no-instance of LWE}\right] \tag{3.88}$$

is small. Our strategy is to identify a correspondence between valid message-signature pairs and "good" inputs to the hash oracle. We then argue that, with high probability over the choice of LWE no-instance (A, T) and hash oracle $H(\cdot)$, the number of good inputs is a very small fraction of the total number of inputs.

Moreover, whether a given input is good is determined solely by its corresponding output from the hash oracle, implying that the only way to discover good inputs is to perform a search through an unstructured space.

Thus, a computationally bounded forger cannot expect to find good input (and hence a valid forgery), even with quantum access to the random oracle. This argument establishes the claim that the LWE-solver $\mathcal{S}$ outputs "yes" only with small probability, as desired.

3.7.1 Correspondence Between Valid Signatures and Good Hash Inputs

Let $\mathsf{w} \in \mathbb{W}$, and let msg be an arbitrary message. For any fixed choice of random oracle $H(\cdot)$ and LWE no-instance (A, T) the hash input $(\mathsf{w}, \mathsf{msg})$ is called *good for* $H(\cdot), \mathsf{A}, \mathsf{T}$ if there exists $z \in \mathbb{S}$ with

$$[\mathsf{A}z - \mathsf{T}H(\mathsf{w}, \mathsf{msg})] = \mathsf{w}. \tag{3.89}$$

Proposition 3.7.1 (Correspondence between valid signatures and good hash inputs)**.** *If* $(\mathsf{msg}, (\mathsf{z}, \mathsf{c}))$ *is a valid message-signature pair for public key* (A, T) *and hash oracle* $H(\cdot)$ *then* $([\mathsf{A}z - \mathsf{T}c], \mathsf{msg})$ *is good for* $H(\cdot), \mathsf{A}, \mathsf{T}$*.*

Proof. Write $\mathsf{w} = [\mathsf{A}z - \mathsf{T}c]$. Because (z, c) is a valid signature for msg we have

$$H(\mathsf{w}, \mathsf{msg}) = H\left([\mathsf{A}z - \mathsf{T}c], \mathsf{msg}\right) = \mathsf{c}. \tag{3.90}$$

Then

$$[\mathsf{A}z - \mathsf{T}H(\mathsf{w}, \mathsf{msg})] = [\mathsf{A}z - \mathsf{T}c] = \mathsf{w} \tag{3.91}$$

as desired. □

A corollary of Proposition 3.7.1 is that the ability to find a message-signature pair $(\mathsf{msg}, (\mathsf{z}, \mathsf{c}))$ that is valid for public key (A, T) using q_h classical or quantum queries to $H(\cdot)$ implies the ability to find a hash input $(\mathsf{w}, \mathsf{msg})$ that is good for $H(\cdot), \mathsf{A}, \mathsf{T}$ using the same number of classical or quantum queries to $H(\cdot)$.

3.7.2 The Fraction of Hood Hash Inputs

We wish to bound the probability over hash oracles $H(\cdot)$ and LWE no-instances (A, T) that a non-negligible fraction of hash inputs $(\mathsf{w}, \mathsf{msg})$ are good. To this end, define the sets

$$\mathbb{M} := \{(\mathsf{w}, \mathsf{msg}) : \mathsf{w} \in \mathbb{W}, \mathsf{msg} \text{ is a message}\} \tag{3.92}$$

$$\mathbb{M}(H, \mathsf{A}, \mathsf{T}) := \{(\mathsf{w}, \mathsf{msg}) \in \mathbb{M} : (\mathsf{w}, \mathsf{msg}) \text{ is good for } H(\cdot), \mathsf{A}, \mathsf{T}\} \tag{3.93}$$

Discussion is somewhat complicated by the fact that there is an infinite number messages, and hence $\mathbb{M}$ and $\mathbb{M}(H, \mathsf{A}, \mathsf{T})$ are infinite sets. For ease of exposition we presume a fixed, large upper bound such as 2^{2^λ} on the size of $\mathbb{M}$. After all, no computationally bounded forger could possibly query the hash oracle on inputs whose bit length exceeds 2^λ. Under this presumption, $\mathbb{M}$ is a finite set and so $\#\mathbb{M}$ is a positive integer. The ratio

$$\frac{\#\mathbb{M}(H, \mathsf{A}, \mathsf{T})}{\#\mathbb{M}} \tag{3.94}$$

is the fraction of inputs that are good.

Our goal is to show that the ratio (3.94) is negligibly small with high probability over the choice of $H(\cdot), \mathsf{A}, \mathsf{T}$. To this end, for each message $(\mathsf{w}, \mathsf{msg}) \in \mathbb{M}$ define the boolean random variable

$$X_{(\mathsf{w},\mathsf{msg})} = \begin{cases} 1 & \text{if } (\mathsf{w}, \mathsf{msg}) \text{ is good for } H(\cdot), \mathsf{A}, \mathsf{T} \\ 0 & \text{otherwise} \end{cases} \tag{3.95}$$

and observe

$$\frac{\#\mathbb{M}(H, \mathsf{A}, \mathsf{T})}{\#\mathbb{M}} = \frac{1}{\#\mathbb{M}} \sum_{(\mathsf{w},\mathsf{msg}) \in \mathbb{M}} X_{(\mathsf{w},\mathsf{msg})}, \tag{3.96}$$

which is an average over boolean random variables. Moreover, the random variables $X_{(\mathsf{w},\mathsf{msg})}$ are independent and so we may apply Hoeffding bounds to obtain

$$\Pr_{H,(\mathsf{A},\mathsf{T})}\left[\frac{\#\mathbb{M}(H, \mathsf{A}, \mathsf{T})}{\#\mathbb{M}} - \operatorname*{Ex}_{H,(\mathsf{A},\mathsf{T})}\left[\frac{\#\mathbb{M}(H, \mathsf{A}, \mathsf{T})}{\#\mathbb{M}}\right] \geq \delta\right] \leq \exp\left(-2\#\mathbb{M}\delta^2\right). \tag{3.97}$$

Because $\#\mathbb{M}$ is very large relative to other TESLA parameters, we may choose δ so small that it can safely be assumed to equal zero. For example, if $\#\mathbb{M} = 2^{2^\lambda}$ then the probability (3.97) is negligibly small even when δ is as small as $2^{-2^{\lambda-2}}$. Thus, the ratio (3.94) is almost certain to be very close to its expecation

$$\underset{H,(\mathsf{A},\mathsf{T})}{\mathrm{Ex}}\left[\frac{\#\mathbb{M}(H,\mathsf{A},\mathsf{T})}{\#\mathbb{M}}\right]. \tag{3.98}$$

This expectation equals

$$\frac{1}{\#\mathbb{M}}\sum_{(\mathsf{w},\mathsf{msg})\in\mathbb{M}}\underset{H,(\mathsf{A},\mathsf{T})}{\mathrm{Ex}}\left[X_{(\mathsf{w},\mathsf{msg})}\right] \tag{3.99}$$

and by definition,

$$\underset{H,(\mathsf{A},\mathsf{T})}{\mathrm{Ex}}\left[X_{(\mathsf{w},\mathsf{msg})}\right] = \underset{H,(\mathsf{A},\mathsf{T})}{\Pr}\left[(\mathsf{w},\mathsf{msg}) \text{ is good for } H(\cdot),\mathsf{A},\mathsf{T}\right]. \tag{3.100}$$

It remains to bound this probability for each hash input $(\mathsf{w},\mathsf{msg})$.

3.7.3 Good Hash Inputs are Rare

For each choice of $\mathsf{w}\in\mathbb{W}$ and LWE no-instance (A,T) we define the set $\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T})\subset\mathbb{H}$ as

$$\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T}) := \{c\in\mathbb{H} \mid \exists z\in\mathbb{S} : [\mathsf{A}z - \mathsf{T}c] = \mathsf{w}\}. \tag{3.101}$$

Observe that a hash input $(\mathsf{w},\mathsf{msg})$ is good for $H(\cdot),\mathsf{A},\mathsf{T}$ if and only if $H(\mathsf{w},\mathsf{msg})\in\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T})$. Thus,

$$\underset{H,(\mathsf{A},\mathsf{T})}{\Pr}\left[(\mathsf{w},\mathsf{msg}) \text{ is good for } H(\cdot),\mathsf{A},\mathsf{T}\right] = \underset{(\mathsf{A},\mathsf{T})}{\mathrm{Ex}}\left[\frac{\#\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T})}{\#\mathbb{H}}\right]. \tag{3.102}$$

We prove the following.

Proposition 3.7.2 (Good Hash Inputs are Rare)**.** *For all* $\mathsf{w}\in\mathbb{W}$ *it holds that*

$$\underset{(\mathsf{A},\mathsf{T})}{\mathrm{Ex}}\left[\max_{\mathsf{w}\in\mathbb{W}}\left\{\frac{\#\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T})}{\#\mathbb{H}}\right\}\right] \le \frac{1}{2\#\mathbb{H}}\left(1+\frac{\#\Delta\mathbb{H}\#\Delta\mathbb{S}\#\Delta\mathbb{L}}{q^m}\right). \tag{3.103}$$

Proof. Define the set

$$\mathbb{D}(\mathsf{A},\mathsf{T}) := \{\mathsf{b}\in\Delta\mathbb{H} : \exists\mathsf{y}\in\Delta\mathbb{S} \text{ with } \mathsf{A}\mathsf{y} - \mathsf{T}\mathsf{b}\in\Delta\mathbb{L}\}. \tag{3.104}$$

In Lemma 3.7.1 below we prove

$$\#\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T}) \leq \frac{\#\mathbb{D}(\mathsf{A},\mathsf{T})+1}{2} \tag{3.105}$$

for all $\mathsf{w} \in \mathbb{W}$. Thus, it suffices to bound the expectation

$$\mathop{\mathrm{Ex}}_{(\mathsf{A},\mathsf{T})}\left[\frac{\#\mathbb{D}(\mathsf{A},\mathsf{T})+1}{2\#\mathbb{H}}\right] = \frac{1}{2\#\mathbb{H}}\left(1+\mathop{\mathrm{Ex}}_{(\mathsf{A},\mathsf{T})}\left[\#\mathbb{D}(\mathsf{A},\mathsf{T})\right]\right). \tag{3.106}$$

We have

$$\begin{aligned}
\mathop{\mathrm{Ex}}_{(\mathsf{A},\mathsf{T})}\left[\#\mathbb{D}(\mathsf{A},\mathsf{T})\right] &= \frac{1}{\#(\mathsf{A},\mathsf{T})}\sum_{(\mathsf{A},\mathsf{T})} \#\left\{\mathsf{b}\in\Delta\mathbb{H} : \exists \mathsf{y}\in\Delta\mathbb{S} \text{ with } \mathsf{A}\mathsf{y}-\mathsf{T}\mathsf{b}\in\Delta\mathbb{L}\right\} && (3.107)\\
&= \frac{1}{\#(\mathsf{A},\mathsf{T})}\sum_{(\mathsf{A},\mathsf{T})}\sum_{\mathsf{b}\in\Delta\mathbb{H}} \mathrm{bool}\left[\exists \mathsf{y}\in\Delta\mathbb{S} \text{ with } \mathsf{A}\mathsf{y}-\mathsf{T}\mathsf{b}\in\Delta\mathbb{L}\right] && (3.108)\\
&\leq \frac{1}{\#(\mathsf{A},\mathsf{T})}\sum_{(\mathsf{A},\mathsf{T})}\sum_{\mathsf{b}\in\Delta\mathbb{H}}\sum_{\mathsf{y}\in\Delta\mathbb{S}} \mathrm{bool}\left[\mathsf{A}\mathsf{y}-\mathsf{T}\mathsf{b}\in\Delta\mathbb{L}\right] && (3.109)\\
&= \sum_{\mathsf{b}\in\Delta\mathbb{H}}\sum_{\mathsf{y}\in\Delta\mathbb{S}}\frac{1}{\#(\mathsf{A},\mathsf{T})}\sum_{(\mathsf{A},\mathsf{T})} \mathrm{bool}\left[\mathsf{A}\mathsf{y}-\mathsf{T}\mathsf{b}\in\Delta\mathbb{L}\right] && (3.110)\\
&= \sum_{\mathsf{b}\in\Delta\mathbb{H}}\sum_{\mathsf{y}\in\Delta\mathbb{S}}\Pr_{(\mathsf{A},\mathsf{T})}\left[\mathsf{A}\mathsf{y}-\mathsf{T}\mathsf{b}\in\Delta\mathbb{L}\right]. && (3.111)
\end{aligned}$$

For each fixed choice of $\mathsf{y}\in\Delta\mathbb{S}$, $\mathsf{b}\in\Delta\mathbb{H}$, if A,T are uniformly random matrices then $\mathsf{A}\mathsf{y}-\mathsf{T}\mathsf{b}$ is a uniformly random vector from $\mathbb{Z}_q^m$. Thus, the probability $\Pr_{(\mathsf{A},\mathsf{T})}\left[\mathsf{A}\mathsf{y}-\mathsf{T}\mathsf{b}\in\Delta\mathbb{L}\right]$ is simply the probability that a random vector lands in $\Delta\mathbb{L}$. That is,

$$\Pr_{(\mathsf{A},\mathsf{T})}\left[\mathsf{A}\mathsf{y}-\mathsf{T}\mathsf{b}\in\Delta\mathbb{L}\right] = \frac{\#\Delta\mathbb{L}}{q^m}. \tag{3.112}$$

Thus, the expectation becomes

$$\mathop{\mathrm{Ex}}_{(\mathsf{A},\mathsf{T})}\left[\#\mathbb{D}(\mathsf{A},\mathsf{T})\right] \leq \sum_{\mathsf{b}\in\Delta\mathbb{H}}\sum_{\mathsf{y}\in\Delta\mathbb{S}}\frac{\#\Delta\mathbb{L}}{q^m} = \frac{\#\Delta\mathbb{H}\#\Delta\mathbb{S}\#\Delta\mathbb{L}}{q^m} \tag{3.113}$$

as desired. □

Lemma 3.7.1. *Let $\mathbb{D}(\mathsf{A},\mathsf{T})$ be as defined in (3.104). For all LWE no-instances (A,T) and all $\mathsf{w}\in\mathbb{W}$ it holds that*

$$\#\mathbb{H}(\mathsf{w},\mathsf{A},\mathsf{T}) \leq \frac{\#\mathbb{D}(\mathsf{A},\mathsf{T})+1}{2}. \tag{3.114}$$

Proof. Let $\mathsf{c}, \mathsf{c}' \in \mathbb{H}(\mathsf{w}, \mathsf{A}, \mathsf{T})$ as witnessed by $\mathsf{z}, \mathsf{z}' \in \mathbb{S}$, respectively. We claim that $\mathsf{c}-\mathsf{c}'$ is in $\mathbb{D}(\mathsf{A}, \mathsf{T})$. It is clear that $\mathsf{c}-\mathsf{c}' \in \Delta\mathbb{H}$ and $\mathsf{z}-\mathsf{z}' \in \Delta\mathbb{S}$. It remains to verify $\mathsf{A}(\mathsf{z}-\mathsf{z}')-\mathsf{T}(\mathsf{c}-\mathsf{c}') \in \Delta\mathbb{L}$. We have

$$\mathsf{A}(\mathsf{z}-\mathsf{z}') - \mathsf{T}(\mathsf{c}-\mathsf{c}') = \mathsf{A}\mathsf{z} - \mathsf{T}\mathsf{c} - (\mathsf{A}\mathsf{z}' - \mathsf{T}\mathsf{c}'). \tag{3.115}$$

Since $\mathsf{A}\mathsf{z} - \mathsf{T}\mathsf{c}$ and $\mathsf{A}\mathsf{z}' - \mathsf{T}\mathsf{c}'$ have the same high bits, it must be that $\mathsf{A}(\mathsf{z}-\mathsf{z}') - \mathsf{T}(\mathsf{c}-\mathsf{c}')$ is the difference of two vectors from $[-(2^{d-1}-1), 2^{d-1}]^m$, from which it follows that $\mathsf{A}(\mathsf{z}-\mathsf{z}') - \mathsf{T}(\mathsf{c}-\mathsf{c}') \in \Delta\mathbb{L}$. A similar argument proves that the negation $\mathsf{c}'-\mathsf{c} \in \mathbb{D}(\mathsf{A}, \mathsf{T})$.

If $\mathsf{c}_1, \ldots, \mathsf{c}_k$ are distinct elements of $\mathbb{H}(\mathsf{w}, \mathsf{A}, \mathsf{T})$ then $0, \mathsf{c}_1-\mathsf{c}_2, \ldots, \mathsf{c}_1-\mathsf{c}_k$ must be distinct elements of $\mathbb{D}(\mathsf{A}, \mathsf{T})$. Similarly, the negations $\mathsf{c}_2-\mathsf{c}_1, \ldots, \mathsf{c}_k-\mathsf{c}_1$ are also distinct elements of $\mathbb{D}(\mathsf{A}, \mathsf{T})$. To see that $\mathsf{c}_1-\mathsf{c}_2, \ldots, \mathsf{c}_1-\mathsf{c}_k$ are all distinct from their negations, observe that

$$\mathsf{c}_1 - \mathsf{c}_i = -(\mathsf{c}_1 - \mathsf{c}_j) \implies 2\mathsf{c}_1 = \mathsf{c}_i + \mathsf{c}_j \implies \mathsf{c}_i = \mathsf{c}_j \tag{3.116}$$

where the final implication follows from the fact that the entries of $\mathsf{c}_1, \mathsf{c}_i, \mathsf{c}_j$ are all in $\{-1, 0, 1\}$. We have thus listed $2k-1$ distinct elements of $\mathbb{D}(\mathsf{A}, \mathsf{T})$, from which the lemma follows. □

3.7.4 Forgers Cannot Forge on LWE No-Instances

Proposition 3.7.2 provides a bound on the fraction δ_{no} of hash inputs that are good. Moreover, since the goodness of a hash input $(\mathsf{w}, \mathsf{msg})$ depends solely on whether $H(\mathsf{w}, \mathsf{msg})$ is in $\mathbb{H}(\mathsf{w}, \mathsf{A}, \mathsf{T})$, the set of all good hash inputs is a randomly selected set. Thus, the only way to find a good hash input is via search through an unstructured space.

It then follows from lower bounds for quantum search [13] that any algorithm making no more than q_h quantum queries to $H(\cdot)$ finds a good hash input—and thus a valid TESLA forgery—with probability no larger than

$$2(q_h+1)\sqrt{\delta_{\text{no}}}. \tag{3.117}$$

We therefore obtain

$$\Pr\left[\mathcal{S} \text{ output "yes"} \mid (\mathsf{A}, \mathsf{T}) \text{ no-instance of LWE}\right] \leq 2(q_h+1)\sqrt{\delta_{\text{no}}}. \tag{3.118}$$

3.8 Security: Putting it all Together

Assuming that no algorithm with run time comparable to that of $\mathcal{S}$ can solve LWE with success bias exceeding ϵ, we have:

$$\epsilon \geq \Pr\left[\mathcal{S} \text{ output "yes"} \mid (\mathsf{A}, \mathsf{T}) \text{ yes-instance of LWE}\right] \tag{3.119}$$
$$- \Pr\left[\mathcal{S} \text{ output "yes"} \mid (\mathsf{A}, \mathsf{T}) \text{ no-instance of LWE}\right]. \tag{3.120}$$

We know that

$$\Pr\left[\mathcal{S} \text{ output "yes"} \mid (\mathsf{A}, \mathsf{T}) \text{ yes-instance of LWE}\right] \geq \Pr\left[\text{forge}\left(\mathsf{A}, \mathsf{T}\right)\right] - \delta_{\text{yes}}. \tag{3.121}$$

Against a quantum forger, we have that

$$\Pr\left[\mathcal{S} \text{ output "yes"} \mid (\mathsf{A}, \mathsf{T}) \text{ no-instance of LWE}\right] \leq 2(q_h + 1)\sqrt{\delta_{\text{no}}}, \tag{3.122}$$

implying that

$$\Pr\left[\text{forge}(\mathsf{A}, \mathsf{T})\right] \leq \delta_{\text{yes}} + 2(q_h + 1)\sqrt{\delta_{\text{no}}} + \epsilon. \tag{3.123}$$

Against a *classical* forger, we can remove the quadratic speedup on the lower bound query complexity, and the probability becomes

$$\Pr\left[\text{forge}(\mathsf{A}, \mathsf{T})\right] \leq \delta_{\text{yes}} + q_h \cdot \delta_{\text{no}} + \epsilon. \tag{3.124}$$

We now incorporate our bounds on δ_{yes} and δ_{no} in order to derive an explicit upper bound on the forger's success probability. It is convenient to make some simplifying assumptions on the choice of TESLA parameters. These assumptions are not necessary in order to derive a negligibly small upper bound on the forger's success probability—they merely facilitate a simplified statement of the upper bound.

Definition 3.8.1 (Convenient TESLA Parameters)**.** *TESLA parameters are* convenient *if*

1. *With probability* $1 - 2^{-\lambda}$ *over the choice of TESLA keys* (A, T)*,* (S, E) *it holds that* $\text{nwr}(\mathsf{A}, \mathsf{E}) < 1/2$.
2. $\#\Delta\mathbb{H}\#\Delta\mathbb{S}\#\Delta\mathbb{L} < q^m$.

For a given choice of TESLA parameters, condition 1 can be verified via Lemma 3.6.2. (One can check that all proposed TESLA parameter sets meet this condition.)

Using condition 1 of Definition 3.8.1, we can simplify equation 3.56 to

$$\delta_{\text{yes}} \leq q_s\gamma + 4q_s \operatorname{coll}(\mathsf{A},\mathsf{E}) \left(1 + \frac{(q_h+q_s)^2}{2\gamma^2}\right). \tag{3.125}$$

From lemma 3.6.3, we have a bound on $\operatorname{coll}(\mathsf{A},\mathsf{E})$ that holds with probability $1 - 1/K_{\text{coll}}$:

$$\delta_{\text{yes}} \leq q_s\gamma + 4q_s \left(\frac{2^{d+1}}{q}\right)^m K_{\text{coll}} \left(1 + \frac{(q_h+q_s)^2}{2\gamma^2}\right). \tag{3.126}$$

At this point, we note that our result on this bound holds for whatever γ we may choose. As we want the first term to be exponentially small, we will select $\gamma = \frac{1}{2^\lambda q_s}$. Then, using the simplification that $1 + \frac{(q_h+q_s)^2}{2\gamma^2} \approx \frac{(q_h+q_s)^2}{2\gamma^2}$ we get

$$\delta_{\text{yes}} \leq \frac{1}{2^\lambda} + \frac{2^{m(d+1)+2\lambda+1}}{q^m}(q_h+q_s)^2 q_s^3 K_{\text{coll}}. \tag{3.127}$$

From proposition 3.7.2 we have

$$\delta_{\text{no}} \leq \frac{1}{2\#\mathbb{H}} \left(1 + \frac{\#\Delta\mathbb{H}\#\Delta\mathbb{S}\#\Delta\mathbb{L}}{q^m}\right). \tag{3.128}$$

Using condition 2 of Definition 3.8.1, we can simplify this bound on δ_{no} to

$$\delta_{\text{no}} \leq \frac{1}{\#\mathbb{H}}. \tag{3.129}$$

Finally, we substitute this, and our bound for δ_{yes} into (3.123). We also note that $\#\mathbb{H} = 2^h\binom{n'}{h}$. We also must consider the probability with which our bounds do not hold. Doing this, we get that $\Pr[\text{forge}(\mathsf{A},\mathsf{T})]$ is at most

$$\epsilon + \frac{1}{2^\lambda} + \frac{2^{m(d+1)+2\lambda+1}}{q^m}(q_h+q_s)^2 q_s^3 K_{\text{coll}} + 2(q_h+1)\sqrt{\frac{1}{2^h\binom{n'}{h}}} + \frac{1}{K_{\text{nwr}}} + \frac{1}{K_{\text{coll}}}. \tag{3.130}$$

Then by choosing each K value to be 2^λ, we get that this is equal to

$$\epsilon + \frac{3}{2^\lambda} + \frac{2^{m(d+1)+3\lambda+1}}{q^m}(q_h+q_s)^2 q_s^3 + 2(q_h+1)\sqrt{\frac{1}{2^h\binom{n'}{h}}}. \tag{3.131}$$

Classically, we can similarly derive that the adversary's success is bounded by

$$\epsilon + \frac{3}{2^\lambda} + \frac{2^{m(d+1)+3\lambda+1}}{q^m}(q_h+q_s)^2 q_s^3 + q_h\frac{1}{2^h\binom{n'}{h}}. \tag{3.132}$$

Chapter 4

Strong Unforgeability in the QROM

In the previous chapters, we have worked with the concept of existential-unforgeability (definition 1.3.2). This is the standard notion of security for a signature scheme. However, there are situations where existential-unforgeability is not sufficient. In some cases, the notion of strong unforgeability is needed.

Game 4.0.1 (Strong Unforgeability)**.**

1. $\mathcal{C}$ runs Keygen and sends pk to $\mathcal{A}$.
2. $\mathcal{A}$ queries a message M_1 to $\mathcal{C}$.
3. $\mathcal{C}$ returns $\sigma_1 \leftarrow \mathsf{Sign}(M_1, sk)$ to $\mathcal{A}$.
4. $\mathcal{A}$ and $\mathcal{C}$ repeat steps 2-3 for $M_2, M_3, \ldots, M_{q_S}$.
5. $\mathcal{A}$ outputs (M^*, σ^*) such that $(M^*, \sigma^*) \neq (M_i, \sigma_i)$ for all $i \in \{1, \ldots, q_S\}$.

Definition 4.0.1 (Strong Unforgeability under Chosen-Message Attack)**.** *A signature scheme* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is said to be* strongly unforgeable under chosen-message attack *if the probability that any polynomially-bounded adversary* $\mathcal{A}$ *playing game 4.0.1 is able to produce a pair such that* $\mathsf{Vrfy}(M^*, \sigma^*, pk) \rightarrow \mathsf{accept}$ *is negligible in the security parameter.*

Note that this definition is nearly identical to that of existential unforgeability. The only difference is in the adversary's output, (M^*, σ^*). In existential unforgeability, the condition is that M^* cannot be equal to any of the signing queries $M_1, \ldots, M_{q_S}$. In strong

unforgeability, (M^*, σ^*) cannot be equal to any of the signing query and response pairs, $(M_1, \sigma_1), \ldots, (M_{q_S}, \sigma_{q_S})$. However, one part of the pair *can* be equal to one part of a query, as long as both are not. In particular, this means that M^* can be equal to M_i for some i, as long as the forged signature, σ^*, is different from σ_i.

As a result, a forgery is considered valid if $\mathcal{A}$ is able to find a new signature on a message-signature pair they have already seen. As this provides more ways for an adversary to create a forgery, if any polynomially-bounded adversary can *still* not create a forgery in this game, the scheme satisfies a strictly stronger notion of security. However not all schemes that satisfy the definition of existential unforgeability also satisfy strong unforgeability. For these schemes, there exist transformations [39, 26, 41] that modify any existentially unforgeable scheme in a generic way to make it strongly unforgeable.

The transformation in [41] (referred to as TOO hereafter) is particularly interesting because it only needs a mild computational assumption and the overhead it causes to the efficiency is small. However the security reduction for the transformation is in the random-oracle model, and uses techniques such as reprogramming. Therefore its security needs to be reexamined in the quantum random-oracle model.

4.1 Chameleon Hash functions

Chameleon hash functions were introduced by Krawczyk and Rabin [30]. We need a slight generalization proposed in [16]. A family $\mathcal{H}$ of chameleon hash function is a collection of functions h that takes in a message m from a message space $\mathcal{M}$ and some randomness r from a randomness space $\mathcal{R}$, and outputs to a range $\mathcal{Y}$, i.e., $h : \mathcal{M} \times \mathcal{R} \to \mathcal{Y}$. The randomness space is associated with some efficiently sampleable distribution, $D_{\mathcal{R}}$.

In a chameleon hash scheme, there exists a polynomial-time algorithm HG, which upon input of a security parameter 1^n, provides a uniformly random $h \leftarrow \mathcal{H}$ as well as some trapdoor information td. If $(h, td) \leftarrow HG(1^n)$, then for any $m \in \mathcal{M}$ and $y \in \mathcal{Y}$, it is possible to efficiently sample $r \leftarrow h_{td}^{-1}(m, y)$ such that $h(m, r) = y$.

There are three properties we need for a chameleon hash scheme:

- (Chameleon property) If $(h, td) \leftarrow HG$, then for any $m_1, m_2, \ldots, m_{q_S} \in \mathcal{M}$, and $y_1, y_2, \ldots, y_{q_S}$ sampled uniformly and independently from $\mathcal{Y}$, $(h_{td}^{-1}(m_1, y_1), \ldots, h_{td}^{-1}(m_{q_S}, y_{q_S}))$ has distribution computationally indistinguishable from $D_{\mathcal{R}}^{q_S}$.

- (Uniformity) For $h \leftarrow \mathcal{H}$ and $r_1, r_2, \ldots, r_{q_S} \leftarrow D_{\mathcal{R}}$, and any $m_1, \ldots, m_{q_S} \in \mathcal{M}$, $(h, h(M_1, r_1), \ldots, h(M_{q_s}, r_{q_s}))$ is uniform over $(\mathcal{H}, \mathcal{Y}, \ldots, \mathcal{Y})$ up to negligible statistical distance.
- (Collision resistance) For a hash function $h \leftarrow \mathcal{H}$, it is computationally infeasible for an adversary to find $(m, r), (m', r')$, with $(m, r) \neq (m', r')$ such that $h(m, r) = h(m', r')$.

4.2 The TOO Transformation

Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme, H be a hash function, and HG be a chameleon hash generation algorithm. Then we define a new signature scheme, $\Sigma' = (\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Vrfy}')$ as follows.

Signature Scheme 4.2.1. TOO signature scheme Σ'

Algorithm 17 KeyGen'

Input: Security Parameter 1^n
Output: Public key pk', secret key sk'

1: Obtain $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$.
2: Obtain $(h, td) \leftarrow HG(1^n)$.
3: Output $pk' = (pk, h)$, $sk' = (sk, td)$.

Algorithm 18 Sign'

Input: Message $M \in \{0,1\}^*$, secret key $sk' = (sk, td)$
Output: Signature σ'

1: Sample a uniform $C \xleftarrow{\$} \mathcal{Y}$.
2: Obtain $\sigma \leftarrow \mathsf{Sign}(C, sk)$.
3: Compute $m \leftarrow H(M||\sigma)$.
4: Using td, compute $r \leftarrow h_{td}^{-1}(m, C)$.
5: Output $\sigma' = (\sigma, r)$.

Algorithm 19 Vrfy'

Input: Message $M \in \{0,1\}^*$, public key $pk' = (pk, h)$, signature $\sigma' = (\sigma, r)$
Output: accept or reject

1: Compute $m \leftarrow H(M||\sigma)$.
2: Compute $C \leftarrow h(m, r)$.
3: Output $\mathsf{Vrfy}(C, \sigma, pk)$.

The correctness property of the transformed scheme can be verified by inspection, assuming that Σ satisfies the property.

4.3 TOO Classical Proof

Let $\mathcal{A}$ be the forger, $\mathcal{B}$ the reduction, and $\mathcal{C}$ be the challenger. The proof has two cases. In each case, $\mathcal{B}$ and $\mathcal{A}$ will be playing a game of strong unforgeability. Let the probability that $\mathcal{A}$ succeeds be ϵ. In case 1, $\mathcal{C}$ and $\mathcal{B}$ will play a game of existential unforgeability on the signature scheme σ. In case 2, $\mathcal{C}$ and $\mathcal{B}$ will play a game of collision resistance on the chameleon hash function h. We show that if the probability $\mathcal{A}$ succeeds in her forgery is ϵ, then the probability that $\mathcal{B}$ succeeds is $\geq \frac{1}{2}\epsilon - \text{negl}(n)$. At the beginning of the reduction, $\mathcal{B}$ will flip a coin, and guess which case the adversary's forgery will fall under. Clearly, $\mathcal{B}$ will be correct with probability $\frac{1}{2}$.

In our reduction, let the forgery that $\mathcal{A}$ eventually submits be $(M^*, \sigma'^* = (\sigma^*, r^*))$. Let $C^* = h(H(M^*||\sigma^*), r^*)$. Similarly, for each M_i the forger submits to the signing oracle for signing, there is an associated σ'_i and C_i.

Case 1: $C^* \neq C_i$ **for all** i

We show that whenever the forger succeeds in creating a valid forgery of this type, the reduction succeeds in breaking the existential unforgeability of the original scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$.

$\mathcal{C}$ and $\mathcal{B}$ will be playing a game of existential unforgeability, while $\mathcal{B}$ and $\mathcal{A}$ will be playing a game of strong unforgeability. We will show that whenever $\mathcal{A}$ wins their game, $\mathcal{B}$ wins theirs (so long as the forgery is of the type described above).

The games will play out as follows:

First, $\mathcal{B}$ will act as the random oracle for $\mathcal{A}$. In the first case at least (and this will change only slightly case to case), he can do this in the following way. Whenever $\mathcal{A}$ queries the random oracle with a query, $\mathcal{B}$ looks up in a maintained table if that query has been made before. If it has, he responds with the value he responded with before. If it has not, he generates a random number and responds with that.

Now we discuss how the game of strong unforgeability transpires.

$\mathcal{C}$ sends $\mathcal{B}$ a public key pk from the Σ scheme. $\mathcal{B}$ will generate a chameleon hash function h (with corresponding trapdoor td), and send the public key and hash function to $\mathcal{A}$ as $pk' = (pk, h)$.

$\mathcal{A}$ will start submitting messages M_i to $\mathcal{B}$ for signing. For each query, $\mathcal{B}$ does the following:

- Choose a random $\tilde{m}_i$ and $\tilde{r}_i \leftarrow D_{\mathcal{R}}$ and compute $C_i = h(\tilde{m}_i, \tilde{r}_i)$.
- Sign C_i by submitting it to $\mathcal{C}$ as a signing query, obtaining σ_i.
- Query $M_i||\sigma_i$ to the random oracle, obtaining $m_i = H(M_i||\sigma_i)$.
- Using the trapdoor information td, find an r_i such that $h(m_i, r_i) = C_i$.
- Let $\sigma_i' = (\sigma_i, r_i)$ and send σ_i' to $\mathcal{A}$.

Eventually, $\mathcal{A}$ will submit a valid forgery $M^*, \sigma'^* = (\sigma^*, r^*)$. Then, $\mathcal{B}$ computes $C^* = h(H(M^*||\sigma^*), r^*)$. Noting that $C^* \neq C_i$ for all i, and the C_i's are precisely what was submitted to $\mathcal{C}$ for signing queries, and finally, seeing as this is a valid forgery, so $\mathsf{Vrfy}(C^*, \sigma^*) = \text{`}accept\text{'}$, we can see that $\mathcal{B}$ submits (C^*, σ^*) as a valid new forgery, breaking the existential unforgeability of Σ and winning his game with $\mathcal{C}$.

Thus in this case, whenever $\mathcal{A}$ succeeds so does $\mathcal{B}$, and so the probability that $\mathcal{B}$ succeeds given that they correctly guessed that they were in this case is ϵ.

Case 2: $C^* = C_i$ **for some** i

In this case we will show a reduction to break the collision resistance of the chameleon hash function.

To start with, $\mathcal{C}$ sends $\mathcal{B}$ the description of a chameleon hash function h, which $\mathcal{B}$ will find a collision for.

$\mathcal{B}$ then runs the key generation algorithm of the signature scheme Σ, obtaining (pk, sk). They then sends $pk' = (pk, h)$ to $\mathcal{A}$.

For each signing query M_i that $\mathcal{A}$ sends to $\mathcal{B}$, $\mathcal{B}$ does the following:

- Choose a random m_i and $r_i \leftarrow D_{\mathcal{R}}$ and compute $C_i = h(m_i, r_i)$.
- Sign C_i using the signing algorithm Sign, obtaining $\sigma = \mathsf{Sign}(C_i, sk)$.
- Reprogram the random oracle so that $H(M_i||\sigma_i) = m_i$.
- Send $\sigma_i' = (\sigma_i, r_i)$ to $\mathcal{A}$.

Note that we have now permitted $\mathcal{B}$ to reprogram the random oracle for the purposes of this proof. Thus it is necessary to show that $\mathcal{A}$ will still output a valid forgery.

When $\mathcal{A}$ eventually submits her forgery, (M^*, σ^*), we can see that if $\mathcal{B}$ was correct in their guess of being in case 2, $C^* = C_i$ for some i. This implies that $h(H(M_i||\sigma_i), r_i) = h(H(M^*||\sigma^*), r^*)$ for that i. This yields a collision for the chameleon hash function h, which is what $\mathcal{B}$ is looking for. But we must show that the inputs are distinct.

Note that $(M_i, \sigma_i, r_i) \neq (M^*, \sigma^*, r^*)$, simply because both the message and signature of the forgery can't be the same as that of one of the M_i's. So at least one of these values is different.

If $r_i \neq r^*$, we are done. Otherwise, it must be the case that $M^*||\sigma^* \neq M_i||\sigma_i$. In this case, since the values for the random oracle are chosen uniformly at random, with overwhelming probability, $H(M^*||\sigma^*) \neq H(M_i||\sigma_i)$, giving $\mathcal{B}$ a collision for h.

So in this case, $\mathcal{B}$ will succeed as long as $\mathcal{A}$ does up to a negligible probability by Lemma 4.3.1. So the probability that $\mathcal{B}$ succeeds is $\geq \epsilon - \text{negl}(n)$.

Lemma 4.3.1. *For a forger $\mathcal{A}$, let $\mathcal{B}_1$ and $\mathcal{B}_2$ be as below, and have them play a game of strong unforgeability with $\mathcal{A}$. Then*

$$|Pr_{\mathcal{B}_1}(\mathcal{A} \text{ wins}) - Pr_{\mathcal{B}_2}(\mathcal{A} \text{ wins})| \leq negl(n),$$

as long as the underlying signature scheme is existentially unforgeable.

$\mathcal{B}_1$ is defined to operate exactly as the transformation dictates. $\mathcal{B}_2$ will operate as $\mathcal{B}$ was defined in Case 2 above.

Proof. Say the difference in probability that $\mathcal{A}$ wins was not negligible. As the distribution of all values is the same, the only difference from $\mathcal{A}$'s perspective was that the value of $H(M_i||\sigma_i)$ was changed for each i.

But clearly the only way to have the information that they changed is if $\mathcal{A}$ had already queried $H(M_i||\sigma_i)$. But if $\mathcal{A}$ does this with non-negligible probability, then we could construct a reduction to break the existential forgeability of the signature scheme by playing strong unforgeability with $\mathcal{A}$, and before submitting each C_i to the signing oracle, checking to see if $\mathcal{A}$ had queried $M_i||\sigma_i$ to the random oracle. With non-negligible probability, the reduction finds a σ_i that is a valid forgery. So he submits this along with C_i and has broken the existential unforgeability of the scheme. □

Therefore in both cases, as long as $\mathcal{B}$ successfully guesses which case the forgery will fall under, he manages to successfully break either the collision resistance of the chameleon hash function h, or the existential unforgeability of the original signature scheme Σ. Since $\mathcal{B}$ correctly guesses which case he is in half of the time, his probability of success is $\geq \frac{1}{2}\epsilon - \text{negl}(n)$.

4.4 Quantum Challenges and Tools

The bulk of the classical proof carries through to the quantum setting. The reduction algorithm $\mathcal{B}$ can be constructed in the same way and the reduction will still work. However there is a considerable problem in justifying the security of the scheme. The only problem is in Lemma 4.3.1. The proof of the lemma is inherently classical, as it creates an upper bound by looking at the queries $\mathcal{A}$ makes to the random oracle. But in a quantum setting it is impossible to look at a query in its entirety, and even extracting some partial classical information about the query necessarily disturbs the query.

The intuition however, remains the same. If an adversary was able to notice reprogramming, it must be because they had made a query that included $|M_i||\sigma_i\rangle$ in some large amplitude, where σ_i is a signature under Σ for C_i. The adversary can only do this if they were able to break the existential unforgeability of the scheme. Note that unlike in Chapter 3, the challenge here is that the point being reprogrammed is hidden from the adversary in a computational sense. The adversary can't find it because they (presumably) can't break the existential unforgeability of Σ.

To prove the reduction still works, we demonstrate a new scenario where we can adaptively program a quantum random-oracle. This extends existing works (e.g [43, 42, 44])

from information-theoretical settings to a computational setting, and we believe it is potentially useful elsewhere. We will formalize a probabilistic game which we call *witness-search*. It potentially captures the essence of numerous security definitions for cryptographic schemes (e.g. signatures). Then we show that the (computational) hardness of witness-search allows for adaptively programming a quantum random-oracle.

Let Samp be an instance-sampling algorithm. On input 1^n, Samp generates public information pk, description of a predicate P, and a witness w satisfying $P(pk, w) = 1$.

Define a witness-search game WS as below.

Game 4.4.1 (Witness-Search Game WS)**.**

1. Challenger $\mathcal{C}$ generates $(pk, w, P) \leftarrow \mathsf{Samp}(1^n)$. Ignore w. Let $W_{pk} := \{w : P(pk, w) = 1\}$ be the collection of valid witnesses.
2. $\mathcal{A}$ receives pk and produces a string $\hat{w}$ as output.
3. We say $\mathcal{A}$ wins the game if $\hat{w} \in W_{pk}$.

We say $\mathsf{WS}(\mathsf{Samp})$ is hard if, for any polytime $\mathcal{A}$, $\Pr[\mathcal{A} \text{ wins}] \leq \text{negl}(n)$. For instance, Samp could be the KeyGen algorithm of a signature scheme. pk consists of the public key and description of the signature scheme. Predicate P is the verification algorithm and a witness consists of a valid message-signature pair. Security of the signature scheme implies hardness of $\mathsf{WS}(\mathsf{Samp})$.

Lemma 4.4.1 (Hardness of Witness-Search to Programming QRO)**.** *Let two experiments E and E' be as below. If WS is hard, then $Adv := |\Pr_E[b = 1] - \Pr_{E'}[b = 1]| \leq negl(n)$.*

Note that E' differs from E only in that we reprogram the random oracle at some point in E'.

Game 4.4.2 (Experiment E)**.**

1. Generate $(pk, w, P) \leftarrow \mathsf{Samp}(1^n)$.
2. $\mathcal{O} \leftarrow \mathcal{F}$ is drawn uniformly at random from the collection of all functions $\mathcal{F}$.
3. $\mathcal{A}_1$ receives pk as input and makes at most q_1 queries to $\mathcal{O}$. $\mathcal{A}_1$ produces a classical string x.

4. Set $z = \mathcal{O}(x\|w)$.

5. $\mathcal{A}_2$ gets (x, w, z) and may access the final state of $\mathcal{A}_1$. $\mathcal{A}_2$ makes at most q_2 queries to $\mathcal{O}$. It outputs $b \in \{0, 1\}$ at the end.

Game 4.4.3 (Experiment E')**.**

1. Generate $(pk, w, P) \leftarrow \mathsf{Samp}(1^n)$.

2. $\mathcal{O} \leftarrow \mathcal{F}$ is drawn uniformly at random from the collection of all functions $\mathcal{F}$.

3. $\mathcal{A}_1$ makes at most q_1 queries to $\mathcal{O}$. It produces a classical string x.

4. Pick $z \in_R \mathsf{Range}(\mathcal{O})$. Reprogram $\mathcal{O}$ to $\mathcal{O}'$: $\mathcal{O}'(y) = \mathcal{O}(y)$ except that $\mathcal{O}'(x\|w) = z$.

5. $\mathcal{A}_2$ gets (x, w, z) and may access the final state of $\mathcal{A}_1$. $\mathcal{A}_2$ makes at most q_2 queries to $\mathcal{O}'$. It outputs $b \in \{0, 1\}$ at the end.

To prove Lemma 4.4.1, we need another lemma below to pave the road. Roughly we want to argue that if witness-search is hard, then given an oracle which is either the all-zeros function or a function that marks the witness set W_{pk}, no efficient algorithms can distinguish them. This may be intuitively interpreted as a computational analogue of Grover search lower bound.

Lemma 4.4.2. *Let f be the all-zeros function, and f_S be the characteristic function of a set S. Namely, $f_S(x) = 1$ iff $x \in S$. Define two experiments G and G' as below. If $\mathsf{WS}(\mathsf{Samp})$ is hard, then for any efficient $\mathcal{A}$ making $q \leq poly(n)$ queries, $|\Pr_G[b = 1] - \Pr_{G'}[b = 1]| \leq negl(n)$.*

Game 4.4.4 (Experiment G)**.**

1. Generate $(pk, w, P) \leftarrow \mathsf{Samp}(1^n)$.

2. $\mathcal{A}$ is given pk and (quantum) access to f. $\mathcal{A}$ makes at most q queries to f and afterwards w is given to $\mathcal{A}$. It outputs $b \in \{0, 1\}$ and aborts.

Game 4.4.5 (Experiment G')**.**

1. Generate $(pk, w, P) \leftarrow \mathsf{Samp}(1^n)$. Let $f_{pk} = f_{W_{pk}}$, where $W_{pk} = \{w : P(w) = 1\}$ (i.e., $f_{pk}(x) = 1$ iff $x \in W_{pk}$).
2. $\mathcal{A}$ is given pk and (quantum) access to f_{pk}. $\mathcal{A}$ makes at most q queries to f_{pk} and afterwards w is given to $\mathcal{A}$. It output $b \in \{0, 1\}$ and aborts.

Proof. Let $\mathcal{A}$ be an arbitrary algorithm running in G (or G'). Consider another algorithm B that runs in an experiment EXT as follows:

Game 4.4.6 (Extraction Experiment EXT)**.**

1. Generate $(pk, w, P) \leftarrow \mathsf{Samp}(1^n)$. Ignore w.
2. B receives pk and picks $j \in_R \{1, \ldots, q\}$ at random.
3. B simulates $\mathcal{A}$ on pk and (quantum) access to f. Just before $\mathcal{A}$ making the jth query to f, B measures the register that contains $\mathcal{A}$'s query. Let z be the measurement outcome.

Let $p_B = \Pr_{EXT}[z \in W_{pk}]$ be the probability that the output of E is a valid witness. Let $\epsilon = |\Pr_G[b = 1] - \Pr_{G'}[b = 1]|$. In both experiments G and G', pk is selected at random according to Samp. Let P_{pk} be the probability that pk is outputted. Then

$$\begin{aligned}
\epsilon &= \left| \Pr_G[b = 1] - \Pr_{G'}[b = 1] \right| \\
&= \left| \sum_{pk} \Pr_G[b = 1|pk] \cdot P_{pk} - \sum_{pk} \Pr_{G'}[b = 1|pk] \cdot P_{pk} \right| \\
&= \sum_{pk} P_{pk} \left| \Pr_G[b = 1|pk] - \Pr_{G'}[b = 1|pk] \right| .
\end{aligned}$$

Let $\epsilon_{pk} = |\Pr_G[b = 1|pk] - \Pr_{G'}[b = 1|pk]|$. Let $|\phi_i\rangle$ be the superposition of $\mathcal{A}^G$ on input pk when the i'th query is made. Then let $q_y(|\phi_i\rangle)$ be the sum of squared magnitudes in $\mathcal{A}$ querying the oracle on the string y.

Let $S = [q] \times W_{pk}$ and $\delta_{pk} = \sum_{(i,y) \in S} q_y(|\phi_i^{pk}\rangle)$. We employ a theorem by Bennet et al. [8] which states that $\| |\phi_i^{pk}\rangle - |\tilde{\phi}_i^{pk}\rangle \| \leq \sqrt{q \cdot \delta_{pk}}$. Here $|\tilde{\phi}_i^{pk}\rangle$ is defined in the same way as $|\phi_i^{pk}\rangle$ but with G' rather than G.

The same paper [8] also bounds the probability of being able to distinguish the two states, which corresponds to our probability of distinguishing the two experiments, ϵ_{pk}, telling us that

$$\epsilon_{pk} \leq 4 \cdot \left\| |\phi_i^{pk}\rangle - |\tilde{\phi}_i^{pk}\rangle \right\| \leq 4\sqrt{q \cdot \delta_{pk}}\,.$$

Now note that P_B^{pk} (that is, the probability that EXT outputs a valid witness given pk is chosen) can be written as

$$\begin{aligned} P_B^{pk} =& \sum_{i\in[0,q]} \left(\Pr[i \text{ chosen}] \cdot \sum_{(j,y)\in S: j=i} q_y(|\phi_j^{pk}\rangle) \right) \\ =& \frac{1}{q} \sum_{i\in[0,q]} \sum_{(j,y)\in S: j=i} q_y(|\phi_j^{pk}\rangle) \\ =& \frac{1}{q} \sum_{(i,y)\in S} q_y(|\phi_i^{pk}\rangle) = \frac{1}{q}\delta_{pk}. \end{aligned}$$

So we can see that $\epsilon_{pk} \leq 4q\sqrt{P_B^{pk}}$. Then

$$\epsilon = \sum_{pk} P_{pk}\epsilon_{pk} \leq 4q \sum_{pk} P_{pk}\sqrt{P_B^{pk}} \overset{(*)}{\leq} 4q\sqrt{\sum_{pk} P_{pk}P_B^{pk}} = 4q\sqrt{P_B}\,,$$

where (*) applies Jensen's inequality. Finally, notice that B can be viewed as an adversary in the witness-search game $\mathsf{WS}(\mathsf{Samp})$. Therefore, we conclude that $p_B \leq \mathrm{negl}(n)$ by the hypothesis that $\mathsf{WS}(\mathsf{Samp})$ is hard and hence $|\Pr_G[b=1] - \Pr_{G'}[b=1]| \leq \mathrm{negl}(n)$. □

Proof of Lemma 4.4.1. We use a hybrid argument to prove the lemma. Define $E_i, i = 1, \ldots, 4$ as follows.

- $E_1 = E$. ($\mathcal{A}_1^{\mathcal{O}}/\mathcal{A}_2^{\mathcal{O}}$ in short.)
- E_2: identical to E_1 except that in step 3, $\mathcal{O}$ is replaced by $\bar{\mathcal{O}}$ where $\bar{\mathcal{O}}(y) = \mathcal{O}(y)$ except $\bar{\mathcal{O}}(y) = 0$ for any $y = \cdot\|w$ with $w \in W_{pk}$. ($\mathcal{A}_1^{\bar{\mathcal{O}}}/\mathcal{A}_2^{\mathcal{O}}$ in short.)
- E_3: identical to E_2 except that after step 3, we use $\mathcal{O}'$ as defined in E' instead of $\mathcal{O}$. Observe that E_3 can also be obtained from E' by substituting $\bar{\mathcal{O}}$ for $\mathcal{O}$ in step 3. ($\mathcal{A}_1^{\bar{\mathcal{O}}}/\mathcal{A}_2^{\mathcal{O}'}$ in short.)

- $E_4 = E'$. ($\mathcal{A}_1^{\mathcal{O}}/\mathcal{A}_2^{\mathcal{O}'}$ in short.)

Define $\mathrm{Adv}_i = \left|\Pr_{E_i}[b=1] - \Pr_{E_{i+1}}[b=1]\right|$. We will show that Adv_1 and Adv_3 are both negligible using Lemma 4.4.2. $\mathrm{Adv}_2 = 0$ since in both E_2 and E_3, the function values for W_{pk} are assigned uniformly at random and independent of anything else. Therefore we conclude that $\mathrm{Adv} = |\Pr_E[b=1] - \Pr_{E'}[b=1]| \leq \sum \mathrm{Adv}_i = \mathrm{negl}(n)$.

We are only left to prove that $\mathrm{Adv}_1 \leq \mathrm{negl}(n)$, and $\mathrm{Adv}_3 \leq \mathrm{negl}(n)$ follows by similar argument. Suppose for contradiction that there exist $(\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathrm{Adv}_1 \geq 1/p(n)$ for some polynomial $p(\cdot)$. We show that this will lead to a contradiction to Lemma 4.4.2 that $|\Pr_G[b=1] - \Pr_{G'}[b=1]| \leq \mathrm{negl}(n)$, which in turn contradicts the hardness of witness-search. To see this, we construct an algorithm D from $(\mathcal{A}_1, \mathcal{A}_2)$ that runs in G and G' such that $|\Pr_G[b=1:D] - \Pr_{G'}[b=1:D]| \geq 1/p(n)$. Let F be an oracle which ignores the first part of the input and then applies either the all-zeros function f or f_{pk} (as defined in G') on the second part. Let g be a random function. Define another oracle $H = g \circ F$ that implements the following transformation:

$$
\begin{aligned}
|x, y\rangle \mapsto & |x, y\rangle \otimes |0\rangle \quad \text{append an auxiliary register} \\
\mapsto & |x, y\rangle \otimes |\overline{F(x)}\rangle \quad \text{compute the negation of } F \text{ on aux.} \\
\mapsto & |x, y \oplus \overline{F(x)} \cdot g(x)\rangle \otimes |\overline{F(x)}\rangle \quad \text{controlled-}g \\
\mapsto & |x, y \oplus \overline{F(x)} \cdot g(x)\rangle \quad \text{uncompute negation of } F \text{ and disgard aux.}
\end{aligned}
$$

Observe that if F is induced from f then H is identical to a random function $\mathcal{O}$. On the other hand, if F comes from f_{pk} then H is identical to $\bar{\mathcal{O}}$ as in E_2. For an algorithm that queries at most q times to H, we can sample h from a family of $2q$-wise independent functions and simulate H efficiently (with access to F) without any noticeable difference.

Construction of D

1. D receives pk and an oracle F (one of the two candidates above).
2. D simulates oracle $H = g \circ F$ as defined above. D then simulates $\mathcal{A}_1$; each query from $\mathcal{A}_1$ is answered by H with (two) oracle calls to F. Let x be the output of $\mathcal{A}_1$.
3. D receives w (from external challenger). It then simulates $\mathcal{A}_2$ on input $(x, w, z := H(x\|w))$ and oracle queries are answered by h.
4. D outputs the output of $\mathcal{A}_2$.

It is easy to see that if F is induced from f, the views of $\mathcal{A}_1$ and $\mathcal{A}_2$ are identical to that of E_1. Likewise, if F is induced by f_{pk} then it is the same view as in E_2. Therefore $|\Pr_G[b = 1 : D] - \Pr_{G'}[b = 1 : D]| = |\Pr_{E_1}[b = 1 : (\mathcal{A}_1, \mathcal{A}_2)] - \Pr_{E_2}[b = 1 : (\mathcal{A}_1, \mathcal{A}_2)]| \geq 1/p(n)$. This gives a contradiction.

□

4.5 TOO Quantum Proof

Theorem 4.5.1. *Assuming that Σ is an existentially-unforgeable quantum-safe signature scheme, and that we have a family of quantum-safe collision-resistant chameleon hash functions $\mathcal{H}$, the TOO signature scheme (signature scheme 4.2.1) is strongly-unforgeable against adaptive chosen-message attack in the quantum random-oracle model.*

Recall that the classical proof roughly goes as follows: consider a forger $\mathcal{A}$. If (M^*, σ'^*) is the forgery that $\mathcal{A}$ eventually submits, we will let $C^* = h(\mathcal{O}(M^*\|\sigma^*), r^*)$. Similarly, for a signing query made by the forger M_i, we let $C_i = h(\mathcal{O}(M_i\|\sigma_i), r_i)$. We then analyze two separate cases. First the instance where $C^* \neq C_i$ for all i. In this case we show that this gives a break to the existential unforgeability of the signature scheme Σ, by way of (C^*, σ^*). Next, we examine the case where $C^* = C_i$ for some i. In this case we show that $(\mathcal{O}(M^*||\sigma^*), r^*)$ and $(\mathcal{O}(M_i||\sigma_i), r_i)$ provide a break to the collision resistance of the chameleon hash function.

Proof. Let $\mathcal{A}$ be the forger making at most q queries, and let ϵ be the probability that $\mathcal{A}$ succeeds in her forgery. We construct $\mathcal{B}$ that either breaks existential unforgeablity of Σ or finds collisions for $\mathcal{H}$.

4.5.1 Case 1

We define this case as occurring when $C^* \neq C_i$ for all i.

First, $\mathcal{B}$ will be acting as a quantum random oracle for $\mathcal{C}$. To do this, $\mathcal{B}$ simply chooses a $2q$-wise independent hash function $\mathcal{O}$, and for any query $\Sigma\alpha_{x,z}|x, z\rangle$ that $\mathcal{A}$ makes, $\mathcal{B}$ responds with $\Sigma\alpha_{x,z}|x, \mathcal{O}(x) \oplus z\rangle$.

Construction of Existential Forger $\mathcal{B}$

1. $\mathcal{B}$ receives a public key pk from the challenger $\mathcal{C}$.

2. $\mathcal{B}$ simulates a variant of the strongly-unforgeable game with $\mathcal{A}$:

   (a) $\mathcal{B}$ generates $(h, td) \leftarrow HG(1^n)$. Initiate $\mathcal{A}$ with $pk' = (pk, h)$.

   (b) $\mathcal{B}$ simulates a random oracle using a $2q$-wise independent hash function.

   (c) On the ith signing query M_i from $\mathcal{A}$, $\mathcal{B}$ chooses a random C_i. It then signs C_i by submitting it to $\mathcal{C}$, obtaining σ_i. It computes $m_i = \mathcal{O}(M_i||\sigma_i)$, and using the trapdoor information td, finds an r_i such that $h(m_i, r_i) = C_i$. It sends $\sigma_i' = (\sigma_i, r_i)$ to $\mathcal{A}$.

3. Let $(M^*, (\sigma^*, r^*))$ be the final forgery produced by $\mathcal{A}$. Output (C^*, σ^*) as the forgery.

From $\mathcal{A}$'s point of view, a $2q$-wise independent function is identical to a random function (theorem 1.6.1). Noting that $C^* \neq C_i$ for all i, and the C_i's are precisely what was submitted to $\mathcal{C}$ for signing queries, and finally, seeing as this is a valid forgery, so $V(C^*, \sigma^*) = \text{`}accept\text{'}$, we can see that $\mathcal{B}$ submits (C^*, σ^*) as a valid new forgery, breaking the existential unforgeability of Σ and winning his game with $\mathcal{C}$. Thus in this case whenever $\mathcal{A}$ succeeds, so does $\mathcal{B}$, and so the probability that $\mathcal{B}$ succeeds given we are in this case is ϵ.

4.5.2 Case 2

This case is defined as occurring when $C^* = C_i$ for some i. In this case we will show a reduction to break the collision resistance of the chameleon hash function.

Construction of Collision-Finding Adversary $\mathcal{B}$

1. $\mathcal{B}$ receives h from the challenger, which is sampled from the Chameleon hash function family.

2. $\mathcal{B}$, playing the role of a challenger, simulates a variant of the strongly-unforgeable game with $\mathcal{A}$:

   (a) $\mathcal{B}$ generates $(pk, sk) \leftarrow G(1^n)$. Initialize $\mathcal{A}$ with $pk' = (pk, h)$. For $i = \{1, \ldots, q\}$, $\mathcal{B}$ generates m_i uniformly at random and $r_i \leftarrow \mathcal{R}$ (according to the specification of h). $\mathcal{B}$ computes $C_i = h(m_i, r_i)$ and $\sigma_i = S(sk, C_i)$.

   (b) $\mathcal{B}$ simulates a random oracle in the usual way (i.e., $2q$-wise independent hash function).

(c) On the ith signing query M_i from $\mathcal{A}$, $\mathcal{B}$ reprograms the random-oracle: $\mathcal{O}(M_i \| \sigma_i) \leftarrow m_i$ and returns (σ_i, r_i) to $\mathcal{A}$.

3. Let $(M^*, (\sigma^*, r^*))$ be the final forgery produced by $\mathcal{A}$. We know $C^* = C_i$ for some i. Output $(\mathcal{O}(M^*||\sigma^*), r^*), (\mathcal{O}(M_i||\sigma_i), r_i)$ as the collision.

It is easy to see that $\mathcal{B}$ finds a valid collision with overwhelming probability as long as $\mathcal{A}$ produces a valid forgery. This is because if $C^* = C_i$, then $h(\mathcal{O}(M^*||\sigma^*), r^*) = h(\mathcal{O}(M_i||\sigma_i), r_i)$. We simply need to ensure that this is not a trivial collision. Note that since this must be a new forgery, $(M^*, \sigma^*, r^*) \neq (M_i, \sigma_i, r_i)$. If $r^* \neq r_i$, we are done. Otherwise, we can see that $M^*||\sigma^* \neq M_i||\sigma_i$, and thus since the values for $\mathcal{O}(M_i||\sigma_i)$ were chosen uniformly at random, $\mathcal{O}(M^*||\sigma^*) \neq \mathcal{O}(M_i||\sigma_i)$ with overwhelming probability.

Therefore if we let EVT be the event that $\mathcal{A}$ produces a valid forgery, we only need to show that EVT occurs with probability $\Omega(\varepsilon)$ in the construction of $\mathcal{B}$. We prove it by a hybrid argument which transforms the standard strongly unforgeable game into the variant as in the construction of $\mathcal{B}$. We will show that the probability of EVT is essentially preserved in the hybrid argument.

Let Hyd_0 be the standard strongly-unforgeable game with $\mathcal{A}$. By hypothesis $\Pr[\mathsf{EVT} : \mathsf{Hyd}_0] \geq \varepsilon$. Consider the first hybrid Hyd_1 that makes only one change to Hyd_0: when the challenger answers a signing query, instead of querying the random oracle $\mathcal{O}$ to obtain $m_i = \mathcal{O}(M_i \| \sigma_i)$, it samples a random m_i and programs the random oracle so that $\mathcal{O}(M_i \| \sigma) = m_i$. Note that in particular the challenger still uses the trapdoor to find $r_i \leftarrow h^{-1}(C_i, m_i)$. By Lemma 4.4.1, we claim that[1] $\Pr[\mathsf{EVT} : \mathsf{Hyd}_0] - \Pr[\mathsf{EVT} : \mathsf{Hyd}_1]| \leq \mathrm{negl}(n)$. Specifically we instantiate Samp as follows. pk will consists of a public key for Σ, hash function h, and random messages C_i. P will be the verification algorithm of Σ. $w = \sigma_i = S(sk, C_i)$ is the signature generated by $\mathcal{B}$ in 2.a), and W_{pk} consists of all strings that form a valid signature of C_i under Σ. $\mathsf{WS}(\mathsf{Samp})$ is hard because Σ is existential-unforgeable.

Hyd_2 is obtained by a small change in Hyd_1. Instead of sampling a random C_i, it is obtained by computing $h(m_i, r_i)$ from random (m_i, r_i). This change only causes (statistically) a negligible error. This is because if $h \leftarrow \mathcal{H}$ and $r_i \leftarrow \mathcal{R}$, then $C_i = h(m_i, r_i)$ will be uniformly random by the uniformity property of $\mathcal{H}$. In addition the chameleon property of $\mathcal{H}$ tells us that $r_i \leftarrow h_{td}^{-1}(C_i, m_i)$ is distributed statistically close to sampling $r_i \leftarrow \mathcal{R}$. Therefore the order of generating C_i and r_i does not matter.

Thus we see that $\mathcal{B}$ is able to break the collision-resistance property of the Chameleon hash function.

[1] More precisely, we need to introduce sub-hybrids and each sub-hybrid makes such a change for just one signing query.

In summary, we have shown that if there is an adversary $\mathcal{A}$ breaking Σ', then there is an adversary who manages to break either the collision resistance of the chameleon hash function $\mathcal{H}$, or the existential unforgeability of the original signature scheme Σ with probability $\Omega(\varepsilon)$. This contradicts the security of Σ and $\mathcal{H}$ if $\varepsilon \geq 1/poly(n)$. Thus we conclude that Theorem 4.5.1 holds. □

4.6 Instantiation

The TOO transformation requires an existentially unforgeable quantum-safe signature scheme and a collision-resistant quantum-safe chameleon hash scheme. There are a variety of candidate quantum-safe signature schemes that are existentially-unforgeable, including those in chapters 2 and 3. For a quantum-safe chameleon hash scheme, there is a lattice-based scheme from [16]. This scheme satisfies the properties of a chameleon hash scheme, with a tight security reduction to the SIS problem.

Chapter 5

Conclusion

In this thesis, we discussed the necessity and use of the quantum random-oracle model, as opposed to the classical random-oracle model. We explored its motivations in classical cryptography and how it arises from the fundamentals of quantum physics and quantum computation. We explained how many techniques that can be easily used and justified in a classical sense can be complicated and difficult to justify in this new model. We reviewed some of the previous work that has been done to overcome these problems and the techniques that have been developed for establishing security reductions.

In chapter 2, we adapted Jonathan Katz's proof of the Leighton-Micali signature scheme from the random-oracle model to the quantum random oracle-model. This generalization of Katz's proof required several new techniques. These included new arguments to reduce the security of oracle composition to that of a single random oracle. It also included a discussion of the security of the Merkle tree construction against a quantum adversary.

Next, in chapter 3, we examined a reformulation of a signature scheme by Bai and Galbraith [4] in the quantum random-oracle model. By applying a result of Bennett et al. [8], we showed how one can justify reprogramming the quantum random oracle. This result applies to the setting where what is being reprogrammed is hidden information theoretically. We also applied some quantum lower bounds and lattice results in order to establish a tight security reduction between the existential unforgeability of the scheme and the LWE problem.

Finally, in chapter 4, we established the security of the TOO transformation, taking a chameleon hash scheme and an existentially unforgeable signature scheme and making it strongly unforgeable. Establishing the security of the scheme in the quantum random-oracle model required another result about oracle reprogramming. While chapter 3 dis-

cussed the problem of reprogramming when the reprogrammed item was hidden in an information-theoretic sense, we needed to consider reprogramming when that item is hidden in a computational sense. We defined new notions of quantum games, and this formalism allowed us to establish the security of the protocol.

5.1 Future Work

While many techniques that cryptographers frequently use in the random-oracle model have now been shown to be sound in the quantum random-oracle model, there has been little work to understand a general framework for working within the random-oracle model and using various techniques. As well, some techniques remain unproven. One of the most important of these is that of rewinding, as it is used in the Fiat-Shamir transformation. Many signature schemes are constructed using this transformation, and so exactly characterizing to what extent this technique is secure in the quantum random-oracle model would be very helpful for designing post-quantum secure signature schemes.

The most fundamental question remains: *If a cryptographic protocol has a security reduction in the random-oracle model, under what conditions does this imply the existence of a reduction in the quantum random-oracle model.* This question was originally addressed in [10], and the authors established some conditions (which they called a 'history-free reduction') under which classical security reductions also held in a quantum environment. However, as we have seen in the previous chapters, many security reductions that do not satisfy these conditions can still be modified to work against a quantum adversary.

Most techniques that have been shown to work in the quantum random-oracle model have only been used for specific, isolated schemes. It would be helpful for cryptographers who have less familiarity with quantum computational models to be able to quickly categorize a new security reduction in order to determine if it meets the conditions to imply a quantum security reduction, and what the nature (e.g., tightness) of this reduction is. This would mean that cryptographers could establish the security of a protocol by using only standard fundamental theorems and techniques, rather than having to reprove basic quantum information-theoretic results for each protocol.

Bibliography

[1] Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. Revisiting TESLA in the quantum random oracle model. In *PQCRYPTO 2017, Proceedings*, 2017. To appear.

[2] Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, and Peter Schwabe. TESLA: Tightly-secure efficient signatures from standard lattices. Cryptology ePrint Archive, Report 2015/755, 2015. https://eprint.iacr.org/2015/755/20161117:055833, revision from 16-Nov-2016.

[3] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, 2009.

[4] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *Topics in Cryptology – CT-RSA 2014*, volume 8366 of *LNCS*, pages 28–47. Springer, 2014.

[5] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, 2012.

[6] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *1st Conference on Computer and Communications Security*, pages 62–73, 1993.

[7] Michael Ben-Or. Probabilistic algorithms in finite fields. In *22nd Annual Symposium on Foundations of Computer Science*, pages 394–398, 1981.

[8] Charles Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

[9] Daniel J. Bernstein. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete? In *SHARCS'09: Special-purpose Hardware for Attacking Cryptographic Systems*, 2009.

[10] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology – Asiacrypt 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, 2011.

[11] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In *Advances in Cryptology – CRYPTO 2013*, volume 8043 of *LNCS*, pages 461–478. Springer, 2013.

[12] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1006–1018, 2016.

[13] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, 1998.

[14] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC 2013)*, pages 575–584. ACM, 2013.

[15] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, July 2004.

[16] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, 2010.

[17] Sanjit Chatterjee, Neal Koblitz, Alfred Menezes, and Palash Sarkar. Another look at tightness II: Practical issues in cryptography. Cryptology ePrint Archive, Report 2016/360, 2016. http://eprint.iacr.org/2016/360.

[18] Özgür Dagdelen, Rachid El Bansarkhani, Florian Göpfert, Tim Güneysu, Tobias Oder, Thomas Pöppelmann, Ana Helena Sánchez, and Peter Schwabe. High-speed signatures from standard lattices. In *Progress in Cryptology – LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 84–103. Springer, 2015.

[19] Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. The fiat-shamir transformation in a quantum world. In *Advances in Cryptology – Asiacrypt 2013*, volume 8270 of *LNCS*, pages 62–81. Springer, 2013.

[20] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *LNCS*, pages 40–56. Springer, 2013.

[21] Edward Eaton and Fang Song. Making existential-unforgeable signatures strongly unforgeable in the quantum random-oracle model. In *10th Conference on the Theory of Quantum Computation, Communication, and Cryptography (TQC)*, pages 147–162, 2015.

[22] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, 2013.

[23] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[24] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.

[25] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *LNCS*, pages 530–547. Springer, 2012.

[26] Qiong Huang, Duncan S Wong, and Yiming Zhao. Generic transformation to strongly unforgeable signatures. In *Applied Cryptography and Network Security*, volume 4251 of *LNCS*, pages 1–17. Springer, 2007.

[27] Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating multi-target attacks in hash-based signatures. In *Public-Key Cryptography – PKC 2016*, volume 9614 of *LNCS*, pages 387–416. Springer, 2016.

[28] Jonathan Katz. Analysis of a proposed hash-based signature standard. In *Security Standardisation Research: Third International Conference, SSR 2016*, volume 10074 of *LNCS*, pages 261–273. Springer, 2016.

[29] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. Oxford University Press, 2006.

[30] Hugo Krawczyk and Tal Rabin. Chameleon hashing and signatures. In *Proceedings of NDSS 2000*, pages 143–154. Internet Society, 2000.

[31] David McGrew, Michael Curcio, and Scott Fluhrer. Hash based signatures — draft-mcgrew-hash-sigs-06. Technical report, Cisco Systems, 03 2017.

[32] Daniele Micciancio. Improving lattice based cryptosystems using the Hermite normal form. In *Cryptography and Lattices*, volume 2146 of *LNCS*, pages 126–145. Springer, 2001.

[33] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and System Sciences*, 58(2):336 – 375, 1999.

[34] Daniel Panario. What do random polynomials over finite fields look like? In *Finite Fields and Applications: 7th International Conference*, volume 2948 of *LNCS*, pages 89–108. Springer, 2004.

[35] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC 2009)*, pages 333–342. ACM, 2009.

[36] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC 2008)*, pages 187–196. ACM, 2008.

[37] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC 2005)*, pages 84–93. ACM, 2005.

[38] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26:1484–1509, 1997.

[39] Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. In *Topics in Cryptology–CT-RSA 2007*, volume 4377 of *LNCS*, pages 357–371. Springer, 2006.

[40] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the fujisaki-okamoto and oaep transforms. In *Theory of Cryptography: TCC 2016-B*, volume 9986 of *LNCS*, pages 192–216. Springer, 2016.

[41] Isamu Teranishi, Takuro Oyama, and Wakaha Ogata. General conversion for obtaining strongly existentially unforgeable signatures. In *Progress in Cryptology-INDOCRYPT 2006*, volume 4329 of *LNCS*, pages 191–205. Springer, 2006.

[42] Dominique Unruh. Quantum position verification in the random oracle model. In *Advances in Cryptology – CRYPTO 2014*, volume 8617 of *LNCS*, pages 1–18. Springer, 2014.

[43] Dominique Unruh. Revocable quantum timed-release encryption. In *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 129–146. Springer, 2014.

[44] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *Advances in Cryptology-EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 755–784. Springer, 2015.

[45] John Watrous. Zero-knowledge against quantum attacks. *SIAM Journal on Computing*, 39(1):25–58, May 2009.

[46] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. *International Journal of Quantum Information*, 13(4), 2-15.

[47] Mark Zhandry. How to construct quantum random functions. In *Proceedings of FOCS 2012*, pages 679–687, 2012.