Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Aurora Lucie Henriksen

Security of Signatures in The Quantum Random Oracle Model

Master's thesis in Mathematical Sciences
Supervisor: Kristian Gjøsteen
June 2023

NTNU
Norwegian University of
Science and Technology

Aurora Lucie Henriksen

Security of Signatures in The Quantum Random Oracle Model

Master's thesis in Mathematical Sciences
Supervisor: Kristian Gjøsteen
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Abstract / Sammendrag

English

The goal for this thesis is to study provable security in Post-Quantum Cryptography (PQC) and apply this to the lattice based CRYSTAL's Dilithium signature. We will develop and explore the necessary theory of lattices and what we consider to be hard lattice problems, before studying identification schemes and signature schemes. Then proceed to explore what security means — both in the Random Oracle Model (ROM) and the Quantum Random Oracle Model (QROM). We will then look at the QROM security of the Dilithium signature and offer a comparison with the ROM secure Lyubashevsky signature. Lastly we discuss whether the QROM is necessary for proving quantum security or if proofs in the standard ROM is sufficient.

Norsk

Målet med denne oppgaven er å studere beviselig sikkerhet i *post-kvante kryptografi* (PQC) og andvende dette mot CRYSTALs gitter baserte signatur Dilithium. Først definerer vi den nødvendige teorien for å forstå gittere og det vi anser som vanskelige gitter problemer. Vi vil videre utforske identifikasjon- og signatursystemer og hva sikkerhet betyr. Vi skal se på sikkerhet i både *tilfeldig orakle modellen* (ROM) og *kvante tilfeldig orakel modellen* (QROM). Deretter ser vi på Dilithiums QROM bevis og for sammenligning Lyubashevskys ROM bevis. Til slutt diskuterer vi om QROM er en nødvendig modell for å bevise sikkerhet i kvantesammenheng eller om standard ROM er tilstrekkelig.

Acknowledgements

This thesis concludes my time at the Norwegian University of Technology (NTNU) and my Master of Science. It has been three interesting years, filled with differences, learning and evolving. In this occasion I would like to give a few thanks.

Thank you to my undergraduate professors at William Woods University, USA, Dr. Chris Schneider and Dr. Sean Baldridge for encouraging my academic curiosity towards mathematics and physics. Thank you to my supervisor Dr. Kristian Gjøsteen for our weekly meetings, guiding me through this thesis and for your patience while drawing a finite, but seemingly endless, amount of sigma protocols on the blackboard.

Thank you to room 395B for encouraging breaks, relevant and irrelevant mathematical discussions. An extra thank you Ross Juvik for helping me into the cryptography world.

Lastly, thank you to my family for pushing me and helping me believe that finishing this thesis was possible, particularly when I did not quite believe it myself. Thank you!

Aurora Lucie Henriksen
Trondheim, June 2023

Contents

Chapter 1

Introduction

Cryptography and cyber-security as we know it today is mostly based on the hardness of factoring large prime numbers. For the computers of today this is considered to be a "sufficiently impossible" problem to build security upon. However, research on quantum computers are taking steps and with that calling for the cryptographic field to develop new quantum safe protocols.

The reason that we need new protocols as quantum computers are emerging is because of the vastly increased computational power that is unlocked. In 1994 Shor introduced a quantum algorithm able to factor large numbers at a speed unknown to classical computers [1]. In more recent time Gidney and Ekerå, working of Shor's algorithm, published a quantum algorithm capable of factoring 2048 RSA[1] integers [2]. This shows that as quantum computers are evolving, our current cryptographic protocols will become useless against a sufficiently strong quantum computer.

This raises the question — what exactly is a quantum computer? The quick answer to this is that a quantum computer operate on qubits $|\phi\rangle$, where a classical computer operate on bits $\{0, 1\}$. A qubit can be represented as a normal bit, namely 0 or 1, but it can also be represented as superpositions of both. This ability to represent a superposition is what makes quantum computers capable of using algorithms such as Shor's and thus break the factoring based cryptographic protocols we use today, such as the RSA protocol.

How far has the research on quantum computers come today? In November 2021 International Business Machines (IBM) informed that they had broken the 100 qubit barrier as they announced their latest quantum computer the *Eagle* [3]. Only a year later, in November 2022, IBM released the 433 qubit quantum computer *Osprey* [4]. Does this rapid development of quantum computers mean that all classical cryptographic protocols are soon to be rendered useless? The answer to this is: No. These computers are designed for other purposes and are likely not even able to break RSA for smaller primes. In fact, the quantum computers that have been used to break RSA has far fewer qubits. In 2018, Dash et al. [5] factored biprimes: 4088459 and 966887, using

[1]We note that RSA is short for *Rivest–Shamir–Adleman*, namely the cryptographers behind the famous algorithm.

only a 5 qubit processor. When it comes to such factorization, no great breakthroughs have happened since. In other words, science is clearly moving at a rapid pace when it comes to quantum computers, but the best attempts of factorizing primes are far from the size of the 2048 bit primes usually used by RSA. Even though these computers are a stretch away from being widely commercialised and breaking all cryptographic security today, it is apparent that time has come for developing new quantum safe protocols.

In 2016 the American National Institute of Standards and Technology (NIST) called for protocols to create a Post-Quantum Cryptography (PQC) standard. Several candidates based on different problems were submitted. In July 2022, after three rounds of the competition, NIST announced four candidates to be standardized as well as four Key-Establishment Mechanism (KEM) candidates to go on to a fourth round [6]. The four candidates being standardized are:

CRYSTAL's[2] KYBER - lattice based KEM

CRYSTAL's Dilithium - lattice based signature scheme

FALCON - lattice based signature scheme

SPHINCS$^+$ - hash based signature scheme

and the four candidates moving forward to the fourth round are:

BIKE - code based KEM

Classic McElice - code based KEM

HQC - code based KEM

SIKE - isogeny based KEM

The main goal for this thesis is to explore what provable security means for a quantum safe signature. In order to explore just this we will begin by introducing the necessary background in the Preliminaries, before we move on to explore lattices, their properties and significance within cryptography. Next we move on to look at how identification and signature schemes work, as well as how we can transform an identification scheme into a signature scheme. With this knowledge, we look at different security models, before we go deeper into the QROM and look at security against quantum attackers. We look at different security definitions for signatures and apply these to two concrete signatures: the Lyubashevsky signature and the Dilithium signature. Lastly, we discuss the need for a security model for security against quantum attackers. In the discussion we also look back at the two signatures and compare their security.

[2]We note that CRYSTALS is short for *Cryptographic Suite for Algebraic Lattices*; an international team of cryptographers.

Chapter 2

Preliminaries

2.1 Notation

The notation throughout this thesis mainly follow that of Kiltz et al. [7]. We shall state it here in order to make it clear.

Rings Throughout this thesis, R will refer to the ring $\mathbb{Z}[X]/\langle X^n + 1\rangle$ and R_q refers to the ring $\mathbb{Z}_q[X]/\langle X^n + 1\rangle$ where $q \in \mathbb{Z}$. We assume $q \equiv 5 \bmod 8$ as this ensures all polynomials in R_q having coefficients less than $\sqrt{q/2}$ to be invertible in the ring ([8], Lemma 2.2). We write regular font letters to be elements in the rings R and R_q (including $\mathbb{Z}$ and $\mathbb{Z}_q$), lowercase letters with an arrow above to denote vectors with coefficients in the rings and matrices will be represented by a bold uppercase letter.

Modular reductions We let α denote an even (respectively odd) positive integer. We then define $r' = r \bmod^{\pm} \alpha$ to be the unique element r' in range $-\frac{\alpha}{2} < r' < \frac{\alpha}{2}$ (respectively $-\frac{\alpha-1}{2} \leq r' \leq \frac{\alpha-1}{2}$) such that $r' = r \bmod \alpha$, we call this *centered reduction modulo* q. We also define, for any positive α, $r' = r \bmod^{+} \alpha$ to be the unique element r' in range $0 \leq r' < \alpha$ such that $r' = r \bmod \alpha$. When being exact is redundant, we will simply write $r \bmod \alpha$. We extend this definition to the ring $\mathbb{Z}_q[X]/\langle X^n + 1\rangle$. Coefficient wise we get a mapping for elements in the ring to an integer polynomial of degree less than n.

Sampling For $n \in \mathbb{N}$, we define $[n] := \{1, ..., n\}$. For a set S we denote the cardinality of the set by $|S|$. Sampling a uniform random element x from the finite set is denoted by $x \leftarrow S$ and sampling according to some distribution D is denoted by $x \leftarrow D$. The context will make the difference apparent. For Boolean statement B, we denote $[\![B]\!]$ as the bit 1 if B is true and 0 otherwise.

Algorithms We assume all algorithms to be probabilistic, unless otherwise stated. For algorithm A, we denote the probabilistic computation of the algorithm on input x as $y \leftarrow A(x)$, where $y \in A(x)$ indicates all possible outcomes y of A given input x. If A is deterministic we denote the computation of the algorithm on input x as $y := A(x)$. Using fixed randomness, we can make any probabilistic algorithm A deterministic. We write $y := A(x; r)$ to denote A running on input x with randomness r. To denote the event that A returns y on input x, we write $A(x) \Rightarrow y$.

Sizes of Elements For $w \in \mathbb{Z}_q$, we write $\|w\|_\infty$ to be $|w \bmod^{\pm} q|$. For $w = w_0 + w_1 X + ... + w_{n-1} X^{n-1} \in R$, we then define the l_∞ and the l_2 norms as:

$$\|w\|_\infty = \max_i \|w_i\|_\infty, \qquad \|w\| = \sqrt{\|w_0\|_\infty^2 + ... + \|w_{n-1}\|_\infty^2}.$$

Similarly, for $\vec{w} = (w_1, ..., w_k) \in R^k$, we have:

$$\|\vec{w}\|_\infty = \max_i \|w_i\|_\infty, \qquad \|\vec{w}\| = \sqrt{\|w_1\|_\infty^2 + ... + \|w_k\|_\infty^2}.$$

We will use S_η to denote elements $w \in R$, where $\|w\|_\infty \leq \eta$.

Quantum Notation We consider the state of a qubit $|\phi\rangle$ to be a two-dimensional complex vector s.t. $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$ and $\{|0\rangle, |1\rangle\}$ forms an orthogonal basis of $\mathbb{C}^2$. We say that α and β are the complex amplitudes of the qubit $|\phi\rangle$. A classical bit $b \in \{0, 1\}$ expressed as a qubit is denoted as $|b\rangle$. A qubit $|\phi\rangle$ is in *superposition* if $0 < |\alpha| < 1$.

For classical oracles we write $\mathcal{O} : \{0,1\}^n \to \{0,1\}^m$, where we consider the execution of this function to be a reversible unitary transformation. If we have quantum access to $\mathcal{O}$, we model this as:

$$U_\mathcal{O} : |x\rangle|y\rangle \mapsto |x\rangle|y \oplus \mathcal{O}(x)\rangle,$$

where $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$. We see that $U_\mathcal{O}$ is its own inverse. An adversary $\mathcal{A}$ with quantum access to an oracle $\mathcal{O}$ is denoted as $\mathcal{A}^{|\mathcal{O}\rangle}$ and can access the oracle $\mathcal{O}$ in superposition using $U_\mathcal{O}$. An adversary with classical access to $\mathcal{O}$ is similarly denoted as $\mathcal{A}^\mathcal{O}$. We also note that the time it takes to run $U_\mathcal{O}$ is linear to the time it takes to run $\mathcal{O}$ classically.

2.2 Hash Functions and Pseudo Random Functions

Hash functions are readily used within cryptography, but what exactly are hash functions? The quick answer will be that hashes are binary functions designed to have collision resistance, preimage resistance and some more properties. A hash takes a message, in form of a bit-string, as input and outputs a hash value.

2.2.1 Unstructured Hash Functions

Throughout the years NIST has standardized hash algorithms called *Secure Hash Algorithms (SHA)*. In 2007 NIST asked for submissions in order to find the next standard, namely SHA3. In 2012 the winners of the *SHA-3 Cryptographic Hash Algorithm Competition* [9] was published. The winning algorithm was KECCAK, a sponge function hash. Six of the instances for the algorithm were standardized, four fixed length output functions and two eXtendable Output Functions (XOF).

The fixed length functions are what we know as hash functions. Hash functions behave like a random one-to-one function that is designed to be collision resistant — one wants it to be near impossible to find a collision. The hash takes the bit-string message from a larger domain to a fixed smaller domain. This makes hash functions very desirable for cryptography, as it allows us to decrease message size, while also hiding the message and making it look random. NIST standardized the KECCAK algorithm for four instances, namely for $B = 224, 256, 384$ and 512, where B is the length of the output hash in bits. The SHA3 function is described below for input message M and bit-length B:

$$\text{SHA3}(M) = KECCAK[2B](M \parallel 01, B).$$

XOF's have several of the same properties as hash functions, such as randomness and resistance against collision. However, XOF's differ in the fact that the output domain is not fixed, usually it is extended. NIST standardized two instances for XOFs, namely $B = 128$ and 256. For XOFs B denotes the effort it takes to break the various security goals — for $B = 256$ this means 2^{256} permutations. These SHA3 standardized XOFs are called SHAKE, and the function is described below for input message M, security B and desired output length d:

$$\text{SHAKE}(M, d) = KECCAK[2B](M \parallel 1111, d).$$

2.2.2 Pseudorandom Functions

A *psedorandom function* is a mapping $PRF : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^k$ where $n, k \in \mathbb{Z}$ and $\mathcal{K}$ is a finite space, in our setting a finite key space. The advantage function for quantum adversary $\mathcal{A}$ and the PRF is then:

$$Adv^{PR}_{PRF}(\mathcal{A}) := |\Pr[\mathcal{A}^{PRF(K,\cdot)} \Rightarrow |K \leftarrow \mathcal{K}] - \Pr[\mathcal{A}^{RF(\cdot)} \Rightarrow 1]|,$$

assuming the adversary $\mathcal{A}$ only has classical access to oracles $PRF(K, \cdot)$ and $RF(\cdot)$, where RF is a perfect random function $RF : \{0,1\}^n \to \{0,1\}^k$.

2.3 Useful Mathematical Concepts

In this section we will introduce some seemingly random mathematical concepts. Throughout the thesis we will refer back to this section and, hopefully the concepts will seem less random.

Generic Search Problem

For $\lambda \in [0,1]$ we let $\mathfrak{B}_\lambda$ denote the Bernoulli distribution $\mathfrak{B}_\lambda$ such that $\Pr[b = 1] = \lambda$ for bit $b \leftarrow \mathfrak{B}_\lambda$. The *Generic Search Problem (GSP)* is to find $x \in X$, with acess to random oracle $g : X \to \{0,1\}$, where X is a finite set, such that $g(x) = 1$, where $g(x)$ is distributed according to the Bernoulli distribution [10–12]. Following this, we further define the *Generic Search Problem with Bounded probabilities (GSPB)*. Here the Bernoulli parameter $\lambda(x)$ is chosen by an adversary, but is only accepted if $\lambda(x) < \lambda$ for all $x \in X$. The problem remains the same as for GSP, but $g(x)$ is now distributed according to Bernoulli distribution $\mathfrak{B}_{\lambda(x)}$.

Game: $GSPB_\lambda$

1 : $(\lambda(x))_{x\in X} \leftarrow \mathcal{A}_1$
2 : **if** $\exists\ x \in X : \lambda(x) > \lambda$, **then**
3 : **return** 0
4 : $\forall\ x \in X : g(x) \leftarrow \mathfrak{B}_{\lambda(x)}$
5 : $x \leftarrow \mathcal{A}_2^{|g\rangle}$
6 : **return** $g(x)$

Figure 2.1: Quantum adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the $GSPB_\lambda$ game for bounded Bernoulli distribution $\mathfrak{B}_{\lambda(x)}$.

In Figure 2.1 we define the *generic search game* $GSPB_\lambda$ in order to consider an adversary wanting to solve the GSPB. Then, for quantum adversary $\mathcal{A}$ having quantum access to the oracle $g : X \to \{0,1\}$ and playing the $GSPB_\lambda$ game, by Lemma 2.3.1, we see that $Pr[GSPB_\lambda^{\mathcal{A}} \Rightarrow 1] \leq 8 \cdot \lambda \cdot (Q+1)^2$. It can be hard to see the intuition behind the bound here, we refer to Lemma 3.2 of Zhandry [12], instantiated for $\Delta = 1$ and $d = 2(Q+1)$.

To further illustrate the upper hand of a quantum adversary to that of a classical adversary, we consider classical adversary $\mathcal{B}$ with classical access to the oracle $g : X \to \{0,1\}$ playing the $GSPB_\lambda$ game. It is trivial to see that $\Pr[GSPB_\lambda^{\mathcal{B}} \Rightarrow 1] = \lambda \cdot Q$.

Lemma 2.3.1. *For $\lambda \in [0,1]$ an unbounded quantum algorithm $\mathcal{A}$ issuing no more than Q quantum queries to quantum oracle g, we have $Pr[GSPB_\lambda^{\mathcal{A}} \Rightarrow 1] \leq 8 \cdot \lambda \cdot (Q+1)^2$, where the $GSPB_\lambda$ game is defined in Figure 2.1.*

Min-Entropy

Entropy can be considered a measure of the unpredictability or randomness of an outcome over a distribution. The instance of entropy we shall define for this thesis is *min-entropy*, which can be viewed as the minimum amount of uncertainty of a distribution. From a cryptographic point of view, this it is a sort of worst case scenario. Patranabis et al. [13] defines the min-entropy of a random variable Y as min-entropy$(Y) = -\log_2 \left(\max_y \Pr[Y = y]\right)$. We specify for this thesis below.

Definition 2.3.1. If the most likely value of a random variable W is chosen from a discrete distribution D occurs with probability $2^{-\alpha}$ we have min-entropy$(W \mid W \leftarrow D) = \alpha$. If an ID-scheme has α bits of *min-entropy*, then:

$$\Pr_{(pk,sk)\leftarrow IGen}[\text{min-entropy}(W \mid (W, St) \leftarrow P_1(sk)) \geq \alpha] \geq 1 - 2^{-\alpha}. \tag{2.1}$$

Chapter 3

Lattices

As an effort to find new harder problems for cryptography, lattices have made their entry to the field. Lattice based cryptography has shown itself a promising field for security against quantum computers. In NIST's competition to standardize protocols for PQC, three out of four winning algorithms are lattice based. We shall here explore lattices and lattice based problems that will be relevant in this thesis. The definitions and problems introduced here are standardized; we mainly follow Gjøsteen [14–17].

3.1 Lattice Basics

Even within mathematics the word *lattice* carries several definitions. The type of lattice we will be exploring here is what we can view as integer linear combinations of linearly independent vectors in a real vector space.

Definition 3.1.1. An n-dimensional *lattice*, Λ, is a subgroup of $\mathbb{R}^n$ such that:

$$\Lambda = \{\sum_{i=1}^{r} a_i \vec{v_i} \mid a_1, ..., a_r \in \mathbb{Z}\},$$

where vectors $\vec{v_1}, ..., \vec{v_r} \in \mathbb{R}^n$ are linearly independent.

If Λ is a subgroup of $\mathbb{Z}^n$, we call it an *integer lattice*.

A lattice is generated by a *basis* if there exists a set of linearly independent vectors $B = \{\vec{b_1}, ..., \vec{b_r}\}$ such that $\Lambda_B = \{\sum_{i=1}^{r} a_i \vec{b_i} \mid a_i \in \mathbb{Z} \text{ and } \vec{b_i} \in B\}$, we denote such lattice by Λ_B . Further, we can construct a $r \times n$ matrix from basis B, by assigning the i^{th} row of the matrix to be the vector $\vec{b_i}$ for $i = 1, 2, ..., r$. Then:

$$\Lambda_{\mathbf{B}} = \{\vec{a}\mathbf{B} \mid \vec{a} \in \mathbb{Z}^n \text{ and } \mathbf{B} \in \mathbb{Z}^{r \times n}\}$$

is the lattice generated by the basis matrix $\mathbf{B}$. The rank of the matrix $\mathbf{B}$ is r as the rows are linearly independent and we say the lattice has *full rank* if $r = n$.

Throughout this thesis we will be working with polynomials in the ring $\mathbb{Z}_q[X]/\langle X^n + 1\rangle$. Conveniently we then consider the isomorphism below, where $f(x)$ is a polynomial of degree n in $\mathbb{F}[X]$:

$$\varphi : \mathbb{F}[X]/\langle f(x)\rangle \to \mathbb{F}^n$$
$$v_0 + v_1 x + v_2 x^2 + ... + v_{n-1}x^{n-1} \mapsto (v_0, v_1, ..., v_{n-1})$$

We see that the polynomial $v_0+...+v_{n-2}\vec{x}^{n-2}+v_{n-1}\vec{x}^{n-1}$ corresponds to the vector $(v_0, ..., v_{n-2}, v_{n-1})$. Dependant on context we will use the terms polynomial and vector interchangeably. To see why we wish to consider this isomorphism, we define *cyclic lattices*:

Definition 3.1.2. The lattice Λ is a *cyclic lattice* if for every vector $\vec{v} = (v_0, ..., v_{n-2}, v_{n-1}) \in \Lambda$ all vectors $\vec{v}_{rot} = (-v_{n-1}, v_0, ..., v_{n-2}) \in \Lambda$.

We notice that a $(\vec{x}^n + 1)$-cyclic lattice is an ideal in both $\mathbb{Z}[X]/\langle X^n + 1\rangle$ and $\mathbb{Z}_q[X]/\langle X^n + 1\rangle$ as $(v_0 + ... + v_{n-2}\vec{x}^{n-2} + v_{n-1}\vec{x}^{n-1}) \cdot \vec{x} = -v_{n-1} + v_0\vec{x} + ... + v_{n-2}\vec{x}^{n-1}$. The notation $\vec{v}_{rot}$ denotes the cyclic rotation of vector $\vec{v}$, described in the definition.

Definition 3.1.3. Let p be a prime and let $p\mathbb{Z}^n = \{p\vec{x} \mid \vec{x} \in \mathbb{Z}^n\}$. A lattice Λ is then called *p-ary lattice* if $p\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$.

Using this definition, we let $\mathbf{M}$ be any $n \times r$ integer matrix and see that:

$$\Lambda_p(\mathbf{M}) = \{\vec{x} \in \mathbb{Z}^n \mid \exists\ \vec{a} \in \mathbb{Z}^r : \vec{a}\mathbf{M} \equiv \vec{x} \bmod p\} \tag{3.1}$$

$$\Lambda_p^{\perp}(\mathbf{M}) = \{\vec{x} \in \mathbb{Z}^n \mid \mathbf{M}\vec{x}^T \equiv 0 \bmod p\} \tag{3.2}$$

and get the following proposition:

Proposition 3.1.4. *For lattice Λ, the following are equivalent:*

i Λ *is p-ary lattice.*
ii There exists a matrix $\boldsymbol{M}$ *st.* $\Lambda = \Lambda_p(\boldsymbol{M})$.
iii There exists a matrix $\boldsymbol{M}$*'* *st.* $\Lambda_p^{\perp}(\boldsymbol{M})$.

As we study lattices with the intention of looking at lattice based cryptography, we need to define the *successive minima*:

Definition 3.1.5. For a lattice Λ, the i^{th} *successive minimum* $\lambda_i(\Lambda)$ is the smallest real number such that there are i linearly independent vectors no longer than $\lambda_i(\Lambda)$ in Λ.

For a lattice of rank r, we get r successive minima, such that: $0 < \lambda_1(\Lambda) \leq \lambda_2(\Lambda) \leq \cdots \leq \lambda_r(\Lambda)$, where $\lambda_1(\Lambda)$ is shortest length of a non-zero vector in Λ. We note that $\lambda_2(\Lambda)$ need not be the second shortest length.

3.2 Lattice-Based Problems

As we have introduced lattices we want to show how they can be used with respect to cryptography. There are several lattice-based problems currently being explored within the cryptographic community, particularly in context of quantum computers. We shall mention and describe several of these lattice-based problems, but will not explore all of them further. However, we still wish to see how they relate to each other.

Shortest Vector Problem

Finding a short lattice vector with length $\lambda_1(\Lambda)$.

Definition 3.2.1. For a lattice Λ the *shortest vector problem*, denoted SVP, is to find a vector $\vec{x} \in \Lambda$ such that: $\|\vec{x}\|_\infty = \lambda_1(\Lambda)$.

Sometimes there are more than one non-zero vector of length $\lambda_1(\Lambda)$. In these cases just finding a short vector is sufficient.

Definition 3.2.2. For a lattice Λ the γ-*approximate shortest vector problem*, denoted SVP_γ, is to find a vector $\vec{x} \in \Lambda$ such that $\|\vec{x}\|_\infty = \gamma \cdot \lambda_1(\Lambda)$.

Closest Vector Problem

Finding a lattice vector close to a specific point.

Definition 3.2.3. For a lattice $\Lambda \subseteq \mathbb{R}^n$ and $\vec{z} \in \mathbb{R}^n$ the *closest vector problem*, denoted CVP, is to find a vector $\vec{x} \in \Lambda$ such that for any $\vec{y} \in \Lambda$, $\|\vec{x} - \vec{z}\|_\infty \leq \|\vec{y} - \vec{z}\|_\infty$.

Definition 3.2.4. For a lattice $\Lambda \subseteq \mathbb{R}^n$ and $\vec{z} \in \mathbb{R}^n$ the *approximate closest vector problem*, denoted CVP_γ, is to find a vector $\vec{x} \in \Lambda$ such that for any $\vec{y} \in \Lambda$, $\|\vec{x} - \vec{z}\|_\infty \leq \gamma \cdot \|\vec{y} - \vec{z}\|_\infty$.

Learning with Errors (LWE)

The Learning With Errors (LWE) problem, is one of the most used lattice problems in cryptography. There are several variants of the problem, we shall define some of the instances here and note that we will refer to the Modular Learning With Errors (MLWE) problem later in this thesis. The LWE problems are based on the hardness of distinguishing between a uniformly random matrix/vector pair and an actually constructed matrix/vector pair.

Definition 3.2.5. For positive $n, m, q, \beta \in \mathbb{Z}$ such that $\beta \ll q$. The *learning with errors* (LWE) problem wants to distinguish between the following:

i $(\mathbf{A}, \mathbf{A}\vec{s} + \vec{e})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\vec{s} \leftarrow [\beta]$ and $\vec{e} \leftarrow [\beta]^n$

ii $(\mathbf{A}, \vec{u})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and $\vec{u} \leftarrow \mathbb{Z}_q^n$

We have *decision-LWE* where we wish to distinguish between which instance, i or ii, was used. We also have *search-LWE* where given the matrix/vector pair generated as instance i, one wishes to find the vectors $\vec{e}$ and $\vec{s}$.

Definition 3.2.6. For positive $m, k, q \in \mathbb{Z}$ and probability distribution $D : R_q \to [0, 1]$, the *modular learning with errors* (MLWE) problem, wants to distinguish between the following:

i $(\mathbf{A}, \mathbf{A}\vec{s_1} + \vec{s_2})$, where $\mathbf{A} \leftarrow R_q^{m \times k}$, $\vec{s_1} \leftarrow D_q^k$ and $\vec{s_2} \leftarrow D_q^m$

ii $(\mathbf{A}, \vec{t})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times k}$ and $\vec{t} \leftarrow R_q^m$

Here too, *decision-MLWE* wishes to distinguish whether i or ii was used in constructing the matrix/vector pair, and *search-MLWE* wishes to find the secret elements $\vec{s_1}$ and $\vec{s_2}$ when the matrix/vector pair is constructed as in i.

One instance of LWE we will not define here is the Ring Learning With Errors (RLWE). The reason we are mentioning it here is because the MLWE is a generalization of LWE and RLWE, and we wish to display the context. The RLWE is LWE with a polynomial ring instead of $\mathbb{Z}$, and RLWE is MLWE for $k = 1$.

Short Integer Solution

The Short Integer Solutiom (SIS) problem is, along with the LWE problem, one of the most used lattice problems for cryptography. Naturally the problem relates to finding a short lattice vector.

Definition 3.2.7. For a lattice $\Lambda(\mathbf{M})$ where $\mathbf{M} \leftarrow \mathbb{Z}^{n \times m}$ the *short integer solution*, denoted SIS, is to find vector $\vec{x} \in \mathbb{Z}^m$ such that $\mathbf{M}\vec{x} = 0$ and $\|\vec{x}\| \leq \beta$.

The value of β will depend on the parameters for the specific schemes. The smaller β gets, the harder the problem gets. Similarly to the LWE problems, we also have the Ring Shortest Integer Solution (RSIS) and the Modular Shortest Integer Solution (MSIS) problems. Here too, MSIS is a generalization of RSIS and SIS.

Definition 3.2.8. For $\vec{a_1}, ..., \vec{a_m} \in R_q^d$ chosen independantly from the uniform distribution, the *modular shortest integer solution* (MSIS) problem is to find $z_1, ..., z_m \in R$ such that $\sum_{i=1}^m \vec{a_i} \cdot z_1 = 0 \bmod q$ and $0 < \|\vec{z}\| \leq \beta$, where $\vec{z} = (z_1, ..., z_m) \in R^m$.

3.2.1 Relation among Problems

The similarity of the lattice problems can seem somewhat coincidental, however the problems are all connected. Following Laarhoven et al.'s [15] reduction of problems we see, in Figure 3.1, the connection of the problems we have introduced in this chapter.

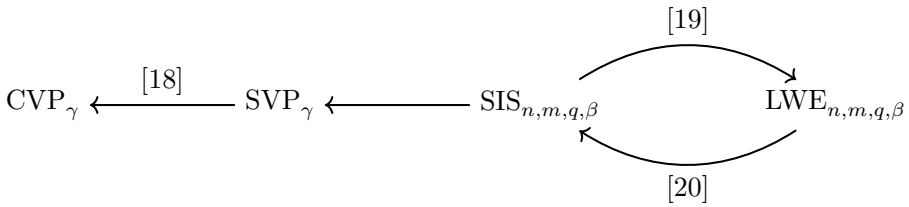

Figure 3.1: Relation among lattice based problems.

It is immediately noticeable that solving the SIS problem means solving the LWE problem, and vice versa. Following the arrows of the chart we see that if we can solve the SIS problem, we can also solve the SVP and CVP problem. We also note that if it is considered hard to solve the CVP problem, it is considered hard to solve SVP and SIS. However, it remains an open problem to show this implication.

3.3 Hashes and Lattices

3.3.1 Structured Hash Functions

Although the randomness of hash functions are very desirable for cryptography, we sometimes want less random looking functions. We have what is called structured hash functions to fulfill this need. Especially useful when we wish to interchange between the message and its hash. These functions have more algebraic structure and we remind ourselves that $R_q = \mathbb{Z}_q[X]/\langle x^n + 1\rangle$.

Definition 3.3.1. For $m \in \mathbb{Z}$ and $D \subseteq R_q$, we have a *hash function family*, denoted $\mathcal{H}(R_q, D, m)$, that maps D^m to R_q . We define this by:

$$\mathcal{H}(R_q, D, m) := \{h_{\vec{a}} : \vec{a} \in R_q^m \mid h_{\vec{a}}(\vec{z}) = \vec{a} \cdot \vec{z}, \text{ for any } \vec{z} \in D^m\}.$$

This means that for $\vec{a} = (a_1, a_2, ..., a_m)$ and $\vec{z} = (z_1, z_2, ..., z_m)$ we get $h_{\vec{a}}(\vec{z}) = a_1z_1 + a_2z_2 + ... + a_mz_m$, where all operations are in R_q . It becomes apparent that the following properties for hash functions $h \in \mathcal{H}(R, D, m)$ hold:

$$h(\vec{y} + \vec{z}) = h(\vec{y}) + h(\vec{z}) \tag{3.3}$$

$$h(\vec{y}c) = h(\vec{y})c \tag{3.4}$$

These properties along with the collision problem from Definition 3.3.2 will become important as we study the security of different cryptographic schemes later.

Definition 3.3.2. Given hash $h \in \mathcal{H}(R_q, D, m)$, the *collision problem* asks to find two distinct elements $\vec{z}, \vec{z}' \in D$ such that there is a collision of the hashed elements $h(\vec{z}) = h(\vec{z}')$. We denote this problem as $Col(h, D)$, where $D \subseteq R_q$.

Chapter 4

Identification and Signature Schemes

In this section we will look at identification schemes (ID-schemes) and signature schemes. These schemes are used in order to prove one's identity with respect to a message, or to sign a message. We will discuss the construction of these schemes as well as the requirements for security. Signature schemes and ID-schemes have many similarities and, in fact, many signature schemes are constructed from ID-schemes. It is therefore natural to start with exploring ID-schemes. The definitions in this chapter follows those of Kiltz et al. [7].

4.1 Identification Scheme

An ID-scheme is an interactive protocol between a prover and a verifier, consisting of three primitives: The key-generation algorithm $IGen$, the prover algorithm $P = (P_1, P_2)$, and the verifier algorithm V.

The protocol is initiated by the key-generation algorithm $IGen$ distributing the secret key sk and the public key pk to the prover and the verifier, respectively. The prover then uses the first part of the prover algorithm P_1 to produce a random value W called a commitment and a state St and sends the commitment to the verifier. Upon receiving the commitment the verifier picks a random value c, called a challenge, and sends it to the prover. The prover then uses the P_2 of the prover algorithm to generate a response Z upon the challenge received from the verifier and returns it to the verifier. The final step is the verifier using the verifier algorithm V to either accept or reject. Accept meaning the verifier believes the prover to know the secret key.

Definition 4.1.1. A tuple of algorithms $ID := (IGen, P, V)$ is defined as an *identification scheme* if there is:

- A key-generation algorithm *IGen* outputting the pair (sk, pk) of corresponding secret key and public key.
  It is assumed the public key defines the set of challenges *ChSet*, the set of commitments *WSet* and the set of responses *ZSet*.
- A split prover algorithm $P = (P_1, P_2)$, where P_1 takes input sk and outputs commitment $W \in WSet$ along with state St, and P_2 takes input sk , commitment W, state St and challenge c and outputs response $Z \in ZSet \cup \{\perp\}$.
  Without loss of generality P_1 is probabilistic and P_2 is deterministic.
- A deterministic verifier algorithm V that takes input pk and conversation transcript (W, c, Z) and outputs either 1 or 0, respectively accept or reject.

A *transcript* $(W, c, Z) \in WSet \times ChSet \times ZSet \cup \{\perp, \perp, \perp\}$ is *valid* with respect to pk, if $V(pk, W, c, Z) = 1$. In Figure 4.1 we define the transcript oracle $Trans(sk)$ that takes input sk and outputs the transcript (W, c, Z) of an interaction between a prover and a verifier. We note that a transcript is defined as $(\perp, \perp, \perp)$ if the response $Z = \perp$. The interactive protocol we have described between a prover and a verifier is often referred to as a *sigma protocol*. This is due to its three-step structure of: commitment, challenge and response. A diagram of the interaction is given in Figure 4.2a.

$Trans(sk)$:

1 : $(W, St) \leftarrow P_1(sk)$
2 : $c \leftarrow ChSet$
3 : $Z \leftarrow P_2(sk, W, c, St)$
4 : **if** $Z = \perp$, **then**
5 : **return** $(\perp, \perp, \perp)$
6 : **return** (W, c, Z)

Figure 4.1: Transcript oracle $Trans(sk)$ for transcripts honestly generated between a prover and a verifier.

As a way of knowing an ID-scheme executes in the correct manner, when executed properly between two honest parties, we define correctness. In the literature we sometimes see the terms correctness and completeness used interchangeably; in this thesis we define and use correctness.

Definition 4.1.2. If a prover knowing sk uses sk to produce a transcript, then an honest verifier with the corresponding pk rejects with negligible probability, we have *correctness*.

In cryptography the term *negligible* is often defined to have a technical meaning. In this thesis we informally use it to denote a probability that is small enough to be ignored. To measure the correctness of a scheme, we further define:

Definition 4.1.3. An ID-scheme *ID* has *correctness error* δ for all $(sk, pk) \in IGen$ if the following conditions are met:

- All possible transcripts, where $Z \neq \perp$, are valid. That is for all $(W, St) \in P_1(sk)$, $c \in ChSet$ and $\perp \neq Z \in P_2(sk, W, c, St)$, we have $V(pk, W, c, Z) = 1$.
- This probability that $Z = \perp$ is contained in an honestly generated transcript (W, c, Z) is bounded by δ. That is $Pr[Z = \perp \mid (W, c, Z) \leftarrow Trans(sk)] \leq \delta$.

Some ID-schemes are what we call *commitment recoverable*. This means that we can recover a commitment W if we have the corresponding challenge c and response Z as well as the pk.

Definition 4.1.4. Let *ID* be an ID-scheme such that for any $(sk, pk) \in IGen$, $c \in ChSet$ and $Z \in ZSet$ there exists an unique $W \in WSet$ such that $V(pk, W, c, Z) = 1$. We say the *ID* is *commitment recoverable* if W can be publicly computed using a commitment recovery algorithm Rec such that $W := Rec(pk, c, Z)$.

4.1.1 Security Properties

In an ideal world, we would not have to worry about malicious adversaries wanting to interrupt our communication. Unfortunately, we do not live in such a world and thus need to define some properties which we can use to prove security for ID. We shall here define: Soundness, Witness Indistinguishability (WI) and Zero-Knowledge (ZK).

While correctness assures us that a honestly generated transcript is unlikely to be rejected; soundness assures us that a non-valid transcript is unlikely to be accepted.

Definition 4.1.5. If a prover only knowing pk somehow produces a transcript, then an honest verifier with pk will only accept with negligible probability, we have *soundness*.

WI and ZK are two different ways of describing the information a prover leaks. As we will study two signatures, where one uses WI and the other uses ZK, we will define both here. The version of ZK we will need for one of these signatures is the No-Abort Honest Verifier Zero-Knowledge (naHVZK), which also will be defined here. For the following definitions the oracle *Trans* is defined in Figure 4.1.

Definition 4.1.6. If an ID-scheme has a pk with two corresponding secret keys, namely sk and sk', such that $V(pk, (Trans(sk)) = V(pk, (Trans(sk')) = 1$ we say that the ID-scheme is perfectly *witness indistinguishable* if a verifier cannot distinguish whether a transcript was produced using sk or sk'.

Definition 4.1.7. An ID-scheme *ID* is perfectly *zero-knowledge* if there exists a simulator *SimTrans* such that given input pk the algorithm outputs transcript (W', c', Z') such that a verifier will accept $(W', c', Z') \leftarrow SimTrans(pk)$ with the same probability as it accepts an honestly generated transcript $(W, c, Z) \leftarrow Trans(sk)$.

Definition 4.1.8. An ID-scheme *ID* is ϵ_{zk}-perfect *no-abort honest verifier zero-knowledge*, if there exist a simulator *SimTrans* such that given input pk the algorithm outputs transcript (W', c', Z') satisfying the following conditions:

- The distribution of the simulated transcript $(W', c', Z') \leftarrow SimTrans(pk)$ has statistical distance at most ϵ_{zk} to that of $(W, c, Z) \leftarrow Trans(pk)$ is no more than ϵ_{zk}.
- The distribution of c' generated by *SimTrans* is uniformly random in *ChSet*, for $c' \neq \perp$.

4.2 Signature Scheme

Signature schemes are used for signing digital messages. Similarly to ID-schemes we refer to the prover and the verifier, but unlike the ID-scheme, the signature scheme is non-interactive.

We have a prover wishing to sign a message and thus uses the signing algorithm *Sign* to produce a signature σ for this message M. The prover then sends this message/signature pair to the verifier. The verifier then uses the verification algorithm *Ver* to either accept or reject the message/signature pair received. Accept meaning the verifier believes the pair to be sent by a known sender and to not have been altered on the way. If the pk is public "anyone" will have the ability to verify the authenticity of the prover.

Definition 4.2.1. A tuple of algorithms $SIG := (IGen, Sign, Ver)$ is defined as a *signature scheme* if there is:

- A key-generation algorithm *IGen* that outputs the pair (sk, pk) of corresponding secret key and public key.
  It is assumed that pk defines the message space $MSet$.
- A signing algorithm *Sign* that takes input message M and secret key sk and outputs signature σ.
- A deterministic verification algorithm *Ver* that takes input message/signature pair (M, σ) and public key pk and outputs 1 or 0, respectively accept or reject.

A message/signature pair $(M, \sigma) \in MSet \times MSet \cup \{\perp, \perp\}$ is valid with respect to pk , if $Ver(pk, M, \sigma) = 1$. We also have correctness for signature schemes; a signature scheme has correctness error δ, if $\Pr[Ver(pk, M, \sigma := Sign(sk, M)) = 0] \leq \delta \ \forall \ (pk, sk) \in IGen$ and $\forall$ $M \in MSet$.

4.2.1 The Fiat-Shamir Transformation

As mentioned many signature schemes are constructed from ID-schemes. In fact, ID-schemes are not actually used as a primitive on their own, rather they are used to build non-interactive signature schemes. We shall explore one common way to turn an ID-scheme into a signature scheme called the Fiat-Shamir transformation. Signature schemes constructed through this transformation will be central in this thesis and we shall therefore define and explain these signatures thoroughly. We note that signature schemes constructed through the Fiat-Shamir transformations derive the security properties we defined in Section 4.1.1 from their underlying ID-schemes.

The idea is that we replace the first part of the verifier with a hash function. Instead of communicating with the verifier, the prover now gets the challenge c from this hash function. We model this as a random function using a Random Oracle (RO)[1]. We will later explore the concept of a RO in greater detail, but for now it is sufficient to view this as a hash function accessible to the prover. Instead of querying the verifier for a challenge, the prover can now receive a challenge from the hash oracle and no outside interaction is required.

Definition 4.2.2. Let $ID := (IGen, P, V)$. Using the *Fiat-Shamir transformation*, described in Figure 4.2b, on ID gives us the signature scheme $SIG := (IGen, Sign, Ver)$. This is denoted $FS[ID, H]$, where ID is the underlying ID-scheme and $H : \{0,1\}^* \to ChSet$ is the hash function used for challenges.

The Fiat-Shamir transformation splits the interactive protocol into the signing algorithm *Sign* and verification algorithm *Ver*. More exactly the split prover algorithm is contained in the signing algorithm and the verifier algorithm is contained in the verification algorithm. Conveniently, correctness is preserved through the transformation, i.e. if the ID-scheme has correctness error δ, so does the signature $FS[ID, H]$ constructed on it.

4.2.2 The Fiat-Shamir with Aborts Transformation

A significant step in the sigma protocol is the prover's response Z upon the challenge c. Sometimes responding to a challenge can cause information to be leaked. More exactly, if the response Z computed for a challenge c is not in the desired range, sending that Z to the verifer will ruin the uniform distribution of the responses sent. This will make the prover seem less random and can cause an adversary to pick up unwanted information. Instead of having to respond, we give the prover the option of what to do — we introduce aborting.

If equipped with the option to abort, the prover can simply send a new commitment to the verifier if $Z \notin ZSet$. This allows the prover to make a new commitment, receive a new challenge and avoid responding to the unwanted challenge. If the protocol is interactive the verifier could become suspicious if the prover aborts many times. However, as ID-schemes are merely used in order to build signature schemes, this is not a problem as the prover never have to inform the verifier that it aborted a challenge.

Definition 4.2.3. Let $ID := (IGen, P, V)$. Using the *Fiat-Shamir with aborts transformation*, described in Figure 4.2c, on ID gives us the signature scheme $SIG := (IGen, Sign, Ver)$. This is denoted $FS[ID, H, \kappa_m]$, where ID is the underlying ID-scheme, $H : \{0,1\}^* \to ChSet$ is the hash function used for challenges and κ_m is the maximum amount of times the scheme has to run in order to produce a valid transcript.

In Figure 4.2 we compare the different executions of the underlying ID-scheme for the ID-scheme itself, the $FS[ID, H]$ derived signature and the $FS[ID, H, \kappa_m]$ derived signature.

[1]We will interchangeably use the full definition and the abbreviation.

Prover		**Verifier**
$(W, St) \leftarrow P_1(sk)$	$\xrightarrow{W}$	
	$\xleftarrow{c}$	$c \leftarrow ChSet$
$Z \leftarrow P_2(sk, W, c, St)$	$\xrightarrow{Z}$	
		$\{0,1\} \leftarrow Ver(pk, W, c, Z)$

(a) An interactive ID-scheme running between a prover and a verifier.

Prover		**Verifier**
$(W, St) \leftarrow P_1(sk)$		
$c \leftarrow \boxed{H(W, M)}$		
$Z \leftarrow P_2(sk, W, c, St)$		
	$\xrightarrow{(c, Z)}$	
		$\{0,1\} \leftarrow Ver(pk, W, c, Z)$

(b) The execution of the underlying ID-scheme of the Fiat-Shamir derived $FS[ID, H]$ signature, where ID is the ID-scheme in 4.2a. In the box we see the RO being queried for a challenge instead of the verifier.

Prover		**Verifier**
$(W, St) \leftarrow P_1(sk)$		
$c \leftarrow \boxed{H(W, M)}$		
$Z \leftarrow P_2(sk, W, c, St)$		
if $Z \notin ZSet$, **then**		
$Z = \perp$		
	$\xrightarrow{(c, Z)}$	
		$\{0,1\} \leftarrow Ver(pk, W, c, Z)$

(c) The execution of the underlying ID-scheme of the Fiat-Shamir derived $FS[ID, H, \kappa_m]$ signature, where ID is the ID-scheme in 4.2a. In the gray boxes we see the prover's option to abort.

Figure 4.2: The different Fiat-Shamir transformations illustrated.

Chapter 5

Security in ROM and QROM

We have now looked at what ID-schemes and signature schemes are and different properties relating to how they run. This leaves us with the most interesting aspect remaining, namely the security of the schemes. In this chapter we shall explore how different types of signature schemes achieve security and what this looks like — both in a classical setting and in a quantum setting.

5.1 ROM and QROM

In order to prove security within a well defined setting we operate with different security models. If a signature has a security proof in the Standard Model (SM) it is considered to be truly secure. In order to imitate the real world the SM only restricts the adversary in terms of computational power and time. Because of the few restrictions of the model; signatures in the SM tend to be of low efficiency. As a response to the SM, along with the discovery of hash functions, the Random Oracle Model (ROM) was created [21]. The ROM model does not ensure security in the real world, however, the test-of-time suggest that a ROM proof is sufficient. In fact, many of the well known and established schemes we use today — like the RSA — are only proved secure in the ROM.

While exploring the Fiat-Shamir transformation in Chapter 4 we briefly mentioned what we call a *random oracle*. In fact, the ROM is structured around this mysterious oracle, often described as "a gnome sitting in a closed box with a dice and a book". That is, the gnome rolls the dice and outputs a hashed value for that query, as well as writing down the query along with the hashed output. The gnome does such for all queries and if a query is made twice, the gnome outputs the hash value it wrote down for that query the first time. Naturally, this does not translate to the real world and ROM has become a way of modelling this through random functions. In a ROM proof an adversary can access all oracles, but the challenger controls all of them. This means that an adversary can only see that something is up if the challenger chooses to manipulate the RO for a query the adversary has already made, and the adversary sends the same query again after

the manipulation. This is not necessarily a problem in ROM as a technique called rewinding is used. Rewinding essentially allows the challenger to rewind the adversary to an earlier point, making it unaware that the RO has been manipulated. We see a flowchart of security in the ROM in Figure 5.1 and note that we will define the different securities later in this chapter [7].

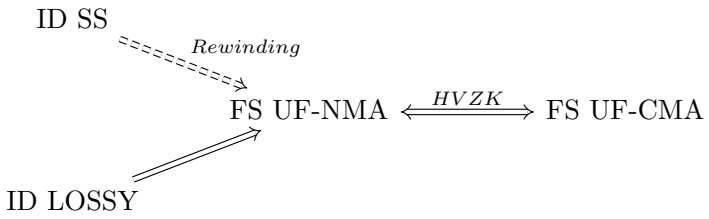

Figure 5.1: Security of $FS[ID, H]$ signatures in the ROM. Dashed arrows denotes non-tight reductions and solid arrows denotes tight reductions. The different security notions are explained in Section 5.2. We note that SS is short for Special Soundness without going into greater detail.

As cryptography is creating new quantum safe protocols Boneh et al. [22] argued the need for a new model for security of these protocols. They introduced the Quantum Random Oracle Model (QROM), built on the existing ROM. In the QROM an adversary can access all oracles, but now has quantum access to the RO controlled by the challenger. The new model allows us to account for adversaries with quantum capabilities. This becomes slightly problematic regarding some of the techniques we have used in the classical ROM. For instance, an adversary is now able to tell if the RO has been manipulated in a position, regardless if the adversary has queried that position prior. If a challenger manipulates the RO for a specific query, it would collapse the superposition the adversary normally would see, thus revealing it has manipulated the RO. Rewinding no longer helps the challenger. To solve this we create a layered patching mechanism using an indicator function. The indicator function allows us to retrieve the necessary information from the queried superposition without collapsing it. We see how different securities relate to each other in the QROM in Figure 5.2 [7].

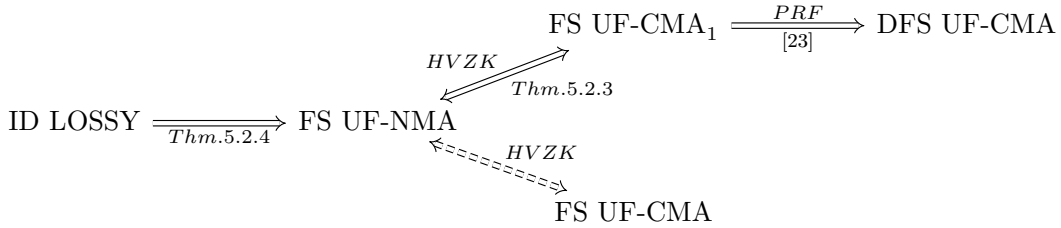

Figure 5.2: Security of $FS[ID, H]$ and $FSD[ID, H, PRF]$ signatures in the QROM. Dashed arrows denotes non-tight reductions and solid arrows denotes tight reductions. The different security notions are explained in Section 5.2 and $FSD[ID, H, PRF]$ is defined in Section 5.2.2.

5.2 UF-CMA Security

When working with digital signatures UnForgeability against Chosen Message Attack (UF-CMA) security has become the standard security notion for signature schemes. In a UF-CMA game the adversary is allowed Q_S signing queries $M' \in \mathit{MSet}$ to which the challenger responds with the signature $\sigma' \leftarrow \mathit{Sign}(sk, M')$. The end goal for the adversary is to produce a valid message/signature pair (M^*, σ^*), such that $\mathit{Ver}(pk, M^*, \sigma^*) = 1$. A diagram of this interaction between the challenger and an adversary is given in Figure 5.3.

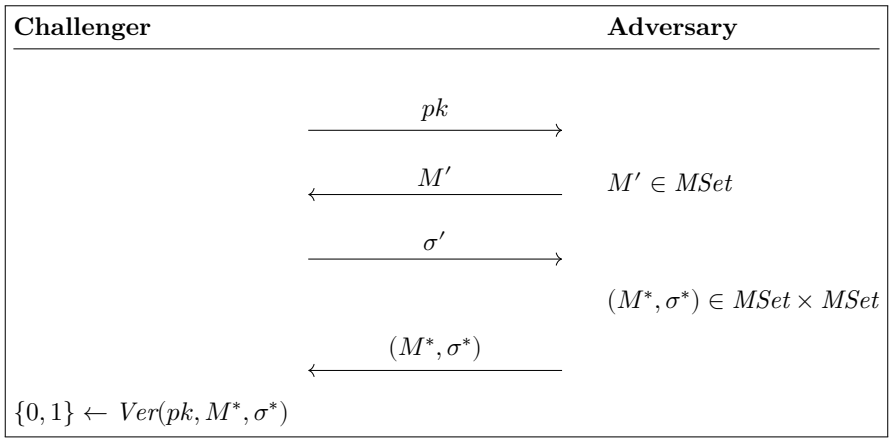

Figure 5.3: The UF-CMA security game played between a challenger and an adversary. The challenger accepts if the adversary submits a valid message/signature pair, and rejects otherwise.

If the adversary successfully produced a valid pair (M^*, σ^*), it seems the adversary has been able to extract information about the signing algorithm through the signing queries — the adversary wins. On the other hand, we say the signature scheme is UF-CMA secure if the chance of the adversary producing such a valid pair is negligible. There are several different kinds of UF-CMA security and we will be discussing all of them in some capacity during this thesis. In order to distinguish the difference of the variants, we shall list and describe all here:

UF-CMA: *UnForgeability against Chosen Message Attack.* An adversary is allowed to make Q_S signing queries to a challenger before responding with a message/signature pair. The adversary wins the game if the challenger accepts the message/signature pair. It is the most used version of security for signatures.

UF-CMA$_1$: *UnForgeability against one-query-per-message Chosen Message Attack* is the same as UF-CMA security, but the adversary can only query the challenger one time per message.

sUF-CMA: *Strong UnForgeability against Chosen-Message Attack* is the same as UF-CMA security, but the adversary can now win by returning a valid message/signature pair where the message has been queried before, but the signature is new. We call the security against "old" message/new signature pair for the *strong unforgeability requirement.*

sUF-CMA$_1$: *Strong UnForgeability against one-query-per-message Chosen Message Attack* is the same as UF-CMA$_1$ security with the strong unforgeability requirement.

UF-NMA: *UnForgeability against No Message Attack.* An adversary is not allowed any signing queries to the challenger before responding with a message/signature pair. The adversary wins the game if the challenger accepts the message/signature pair. It is essentially the same as UF-CMA security with $Q_S = 0$.

As we are interested in security both in ROM and in QROM, we observe the difference in how the UF-CMA game is executed in Figure 5.4. In line 3 and 4 of the UF-CMA game in the QROM, we note that the forged signature is parsed in order to retrieve the commitment W^* and the response Z^*, before constructing the challenge c^* using the random oracle H.

UF-CMA: //ROM	UF-CMA: //QROM
1 : $(pk, sk) \leftarrow IGen$	1 : $(pk, sk) \leftarrow IGen$
2 : $(M^*, \sigma^*) \leftarrow \mathcal{A}^{SIGN}(pk)$	2 : $(M^*, \sigma^*) \leftarrow \mathcal{A}^{\|H\rangle, SIGN}(pk)$
3 : **return** $[\![M^* \notin MSet]\!] \wedge [\![Ver(pk, M^*, \sigma^*)]\!]$	3 : **parse :** $\sigma^* = (W^*, Z^*)$
	4 : $c^* := H(W^* \parallel M^*)$
	5 : **return** $[\![M^* \notin MSet]\!] \wedge [\![Ver(pk, W^*, c^*, Z^*)]\!]$

Figure 5.4: UF-CMA security in the ROM and the QROM. *SIGN* denotes the signing oracle for the signature and H' denotes the hash oracle which is quantum accessible.

5.2.1 Signatures from Lossy ID-schemes

In 2012 Abdalla et al. [24] introduced lossy ID-schemes that could be proven tightly secure in the ROM. In 2017 Alkim et al. [25] proved that the lattice based signature TESLA could be proven UF-CMA secure in the QROM and left it as an open problem to prove the general case for signatures derived from lossy ID-schemes. In this thesis we follow Kiltz et al.'s [7] efforts to create a general QROM proof for lossy derived signatures. Following the flowchart in Figure 5.2 we look at how to achieve the different unforgeability securities for signatures constructed using Fiat-Shamir on lossy ID-schemes in the QROM.

Definition 5.2.1. Let $ID = (IGen, P, V)$ be an ID-scheme. If ID is *lossy* there exist a lossy key-generation algorithm such that $pk_{ls} \leftarrow LossyIGen$, where pk_{ls} is a lossy public key having no corresponding secret key.

We sometimes refer to lossy ID-schemes as $LID := (IGen, LossyIGen, P, V)$. There are two properties of lossy ID-schemes that we want to notice in particular. First, we notice that the keys generated by the lossy key generator *LossyIGen* and the real key generator *IGen* are indistin-

guishable from each other. We define the *LOSS* advantage function:

$$Adv_{LID}^{LOSS}(\mathcal{A}) := |\ \Pr[\mathcal{A}(pk_{ls}) \Rightarrow 1 \mid pk_{ls} \leftarrow LossyIGen] - \Pr[\mathcal{A}(pk) \Rightarrow 1 \mid (pk, sk) \leftarrow IGen]\ | \ . \tag{5.1}$$

Second, we wish to consider the probability that an unbounded adversary can impersonate the prover. We wish to use the fact that there is no secret key corresponding to the lossy public key and consider an unbounded adversary $\mathcal{C}$. We see that relative to lossy key pk_{ls} not even an adversary like $\mathcal{C}$ can impersonate the prover. Considering $\mathcal{C}$ playing the $LOSSY-IMP$ game defined in Figure 5.5, we find the lossy soundness ϵ_{ls} of the ID-scheme. We get $Pr[LOSSY-IMP^{\mathcal{C}} \Rightarrow 1] \leq \epsilon_{ls}$. To bound this probability, we take the expectation over $pk_{ls} \leftarrow LossyIGen$ which, along with the fact that $\mathcal{C}$ is unbounded, allows us to upper bound $\mathcal{C}$'s chance of winning:

$$\Pr[LOSSY-IMP^{\mathcal{C}} \Rightarrow 1] \leq \mathbf{E}\left[\max_{W \in WSet}\left(Pr[\exists\, Z \in ZSet : V(pk_{ls}, W, c, Z) = 1]\right)\right]. \tag{5.2}$$

We achieve equality in (5.2) if we have an adversary $\mathcal{C}$ making the "optimal" decisions. That is $\mathcal{C}$ submits the easiest commitment $W \in WSet$ with challenge $c \leftarrow ChSet$ such that it successfully finds a valid response $Z \in ZSet$.

$LOSSY-IMP$:

1 : $pk_{ls} \leftarrow LossyIGen$
2 : $(W^*, St^*) \leftarrow \mathcal{C}(pk_{ls})$
3 : $c^* \leftarrow ChSet$
4 : $Z^* \leftarrow \mathcal{C}(St, c^*)$
5 : **return** $[\![V(pk_{ls}, W^*, c^*, Z^*)]\!]$

Figure 5.5: The lossy impersonation game.

5.2.2 UF-CMA in the QROM

Following the flowchart in Figure 5.2, we see that signatures built on lossy ID-schemes can be proved UF-NMA, UF-CMA$_1$ and UF-CMA secure in the QROM. In this section we will introduce Theorem 5.2.2 and prove it in a modular fashion. First showing that UF-NMA security and naHVZK implies UF-CMA$_1$ security and then showing that having a $FS[ID, H, \kappa_m]$ signature, where *ID* is lossy implies UF-NMA security. Lastly we will look at what the requirements are in order to extend this to strong unforgeability and how we can build onto UF-CMA$_1$ security to achieve UF-CMA security. In Chapter 6 we will apply this approach to the CRYSTAL's Dilitihium signature scheme.

Theorem 5.2.2. *Let the ID-scheme ID be ϵ_{zk}-perfect naHVZK, ϵ_{ls}-lossy sound and having α bits of min-entropy. Then for any quantum adversary $\mathcal{A}$ playing the UF-CMA$_1$ game issuing at*

most Q_H queries to the quantum random oracle H and Q_S classical queries to the signing oracle $SIGN_1$, there exists a quantum adversary $\mathcal{B}$, such that:

$$Adv_{SIG}^{UF-CMA_1}(\mathcal{A}) \leq Adv_{ID}^{LOSS}(\mathcal{B}) + 8(Q_H+1)^2 \cdot \epsilon_{ls} + \kappa_m Q_S \cdot \epsilon_{zk} + 2^{-\alpha+1}, \tag{5.3}$$

where $Time(\mathcal{B}) = Time(\mathcal{A}) + \kappa_m Q_H \approx Time(\mathcal{A})$.

We observe that the bound of Theorem 5.2.2 is tight. Conducting the same proof in the ROM will give us the same result, with the only difference being the bound depending linearly rather than quadratic on the oracle queries Q_H.

UF-NMA to UF-CMA$_1$

Theorem 5.2.3. *Let the ID-scheme ID be ϵ_{zk}-perfect naHVZK and having α bits of min-entropy. Then for any quantum adversary $\mathcal{A}$ playing the UF-CMA$_1$ game issuing at most Q_H queries to the quantum random oracle H and Q_S classical queries to the signing oracle $SIGN_1$, then there exist a quantum adversary $\mathcal{B}$ playing the UF-NMA game issuing Q_H queries to its own quantum random oracle, such that:*

$$Adv_{SIG}^{UF-CMA_1}(\mathcal{A}) \leq Adv_{SIG}^{UF-NMA}(\mathcal{B}) + 2^{-\alpha+1} + \kappa_m Q_S \cdot \epsilon_{zk}, \tag{5.4}$$

where: $Time(\mathcal{B}) = Time(\mathcal{A}) + \kappa_m(Q_H + Q_S) \approx Time(\mathcal{A})$

Proof. The reduction of games and oracles are described in Figure 5.6. We let $\mathcal{A}$ be a quantum adversary playing UF-CMA$_1$ against signature scheme *SIG*, where $\mathcal{A}$ makes at most Q_H quantum queries to the oracle H and at most Q_S classical queries to the oracle $SIGN_1$, both oracles also described in Figure 5.6.

Game G_0 : Comparing the initial game G_0 to the UF-CMA game in Figure 5.4 it becomes clear that G_0 is the UF-CMA$_1$ game. The signing oracle $SIGN_1$ produces a signature by calling the *GetTrans* procedure for G_0. A real interaction (W_M, c_M, Z_M) is then generated using the deterministic prover algorithms P_1 and P_2. Randomness of the output is assured through the random function *RF*, which is not accessible for the adversary. Looking at line 1 of $SIGN_1$ we see that the adversary is only allowed one signing query per message, we get:

$$Adv_{SIG}^{UF-CMA_1}(\mathcal{A}) = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

Game G_1 : In G_1 we change which *GetTrans* procedure we call, and we see that the *GetTrans* procedure for G_1 uses the naHVZK transcript simulator *SimTrans* to produce a transcript and then patches the quantum oracle H for each query. Unlike G_0 the transcripts are no longer honestly generated.

We look at a classical signing query from the adversary $\mathcal{A}$. In line 1 of the *GetTrans* procedure the variable $\kappa := 0$ is initiated, before starting a while loop in lines 2–4 looking for the smallest

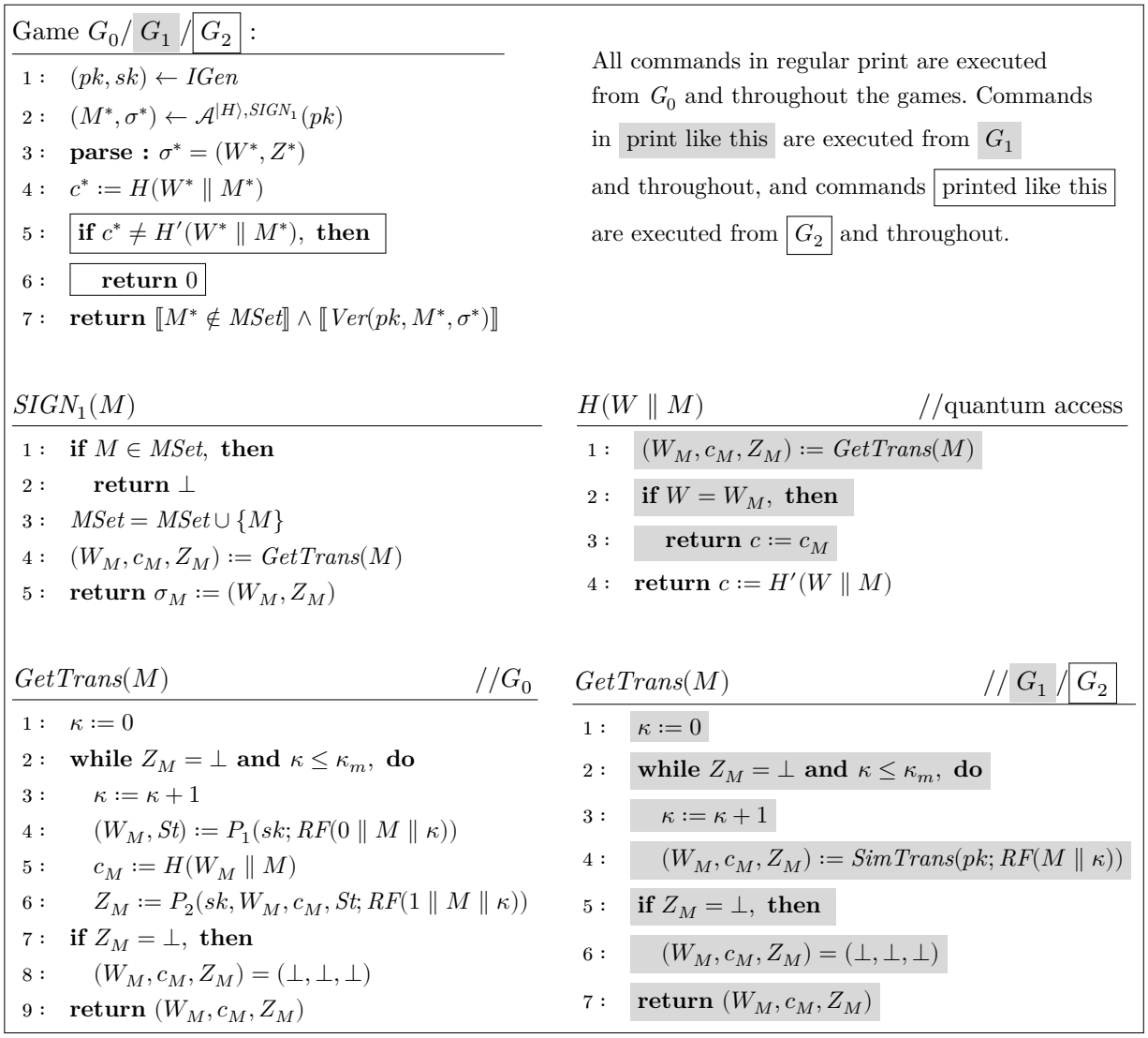

Figure 5.6: The games: G_0, G_1 and G_2, we see in the proof of Theorem 5.2.3.

integer $1 \leq \kappa_M \leq \kappa_m$ satisfying $(W, c, Z) := \mathit{SimTrans}(pk; \mathit{RF}(M \parallel \kappa)$. If no such integer exists the procedure sets $\kappa_M \neq \perp$ and initiates lines 5 and 6, thus returning transcript $(W_M, c_M, Z_M) = (\perp, \perp, \perp)$. If an integer κ_M does exist a transcript is computed, during the while loop, using the simulator on random function RF. In G_1 the transcript is deterministically computed as:

$$(W, c, Z) := \mathit{GetTrans}(M) = \begin{cases} \mathit{SimTrans}(pk; \mathit{RF}(M \parallel \kappa_M), & 1 \leq \kappa_M \leq \kappa_m \\ (\perp, \perp, \perp), & \kappa_M = \perp \end{cases}$$

The signature on a message M is then returned as $\sigma_M := (W_M, Z_M)$.

Recalling the Definition 4.1.8 for naHVZK, we know that the distribution of each σ_M produced by the $\mathit{SimTrans}$ has statistical distance at most $\kappa_m \epsilon_{zk}$ from one computed honestly. Thus every signature computed in G_1 has this statistical distance to that computed in the previous game. In lines 3 and 4 of the quantum random oracle H we see that it is patched in order to ensure that σ_M is a valid signature on M. Concretely, iff $W = W_M$, the oracle sets $c_M := H(W \parallel M)$, where c_M and W_M are both computed by the $\mathit{GetTrans}$ procedure. We avoid revealing to the adversary that a simulator produced the transcript. To further ensure this, we see that the output distribution of oracle H remains the same as the c_M generated by the simulator is uniformly distributed. Using the union bound, we have:

$$|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_0^{\mathcal{A}} \Rightarrow 1]| \leq \kappa_m Q_S \cdot \epsilon_{zk}.$$

Game G_2 : Looking at line 5 of the games, we see that G_2 rejects if $c^* \neq H'(W^* \parallel M^*)$. That means the only way G_1 and G_2 can differ is if $W^* = W_{M^*}$ and $M^* \notin \mathit{MSet}$. In this case G_1 accepts, while G_2 rejects. This is because $W^* = W_{M^*}$ would mean $c^* \neq H'(W^* \parallel M^*)$ causing G_2 to reject in line 6. Further considering that $M^* \notin \mathit{MSet}$, the random variable W_{M^*} would remain completely hidden from the view of the adversary as it has not been revealed as part of a valid signature in any of the signing queries Q_S. The commitment W_{M^*} thus have α bits of min-entropy. Using Definition 2.3.1 we get $Pr[W_{M^*} = W^*] \leq 2^{-\alpha}$ and obtain the inequality:

$$|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]| \leq 2^{-\alpha+1}.$$

Now, we wish to consider the adversary $\mathcal{B}$ in Figure 5.7 playing the UF-NMA game having quantum access to the random oracle H'. Adversary $\mathcal{B}$ then perfectly simulates $\mathcal{A}$'s view from G_2, using its own oracle H' to simulate H' as well as a $2\kappa_m Q_H$-wise independent hash function to perfectly simulating the random function RF.

Assume $\mathcal{A}$ produced a valid forgery in G_2. This means that $(M^*, \sigma^*) \leftarrow \mathcal{A}^{|H\rangle, \mathit{SIGN}_1}$ such that $M^* \notin \mathit{MSet}$ and the parsed signature would give $\mathit{Ver}(pk, W^*, c^*, Z^*) = 1$. To win G_2 we notice that $\mathcal{A}$ must have produced a signature such that $c^* = H(W^* \parallel M^*)$, which means $H(W^* \parallel M^*) = H'(W^* \parallel M^*)$. This means the forgery (M^*, σ^*) would also be valid in the UF-NMA game where it would yield $\mathit{Ver}(pk, W^*, c^*, Z^*) = 1$, where $c^* = H'(W^* \parallel M^*)$, giving us:

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \mathit{Adv}_{\mathit{SIG}}^{\mathit{UF-NMA}}(\mathcal{B}).$$

Adversary $\mathcal{B}^{|H'\rangle}(pk)$ //quantum access

1 : $(M^*, \sigma^*) \leftarrow \mathcal{A}^{|H\rangle, SIGN_1}(pk)$
2 : Parse $\sigma^* = (W^*, Z^*)$
3 : $c^* := H'(W^* \parallel M^*)$
4 : **if** $c^* \neq H'(W^* \parallel M^*)$, **then**
5 : **Abort**
6 : **if** $[\![M^* \notin MSet]\!] \wedge [\![V(pk, W^*, c^*, Z^*)]\!]$, **then**
7 : **return** (M^*, σ^*)
8 : **Abort**

Figure 5.7: Adversary $\mathcal{B}$ with quantum access to H' playing the UF-NMA game against signature *SIG* which has quantum access to H'. The other oracles are defined as in Figure 5.2.3.

Collecting the probabilities we have seen throughout the games, we get:

$$Adv_{SIG}^{UF-CMA_1}(\mathcal{A}) \leq Adv_{SIG}^{UF-NMA}(\mathcal{B}) + 2^{-\alpha+1} + \kappa_m Q_S \cdot \epsilon_{zk},$$

which we recognize as (5.4). To get the running time of $\mathcal{B}$ and complete the proof, we take advantage of the identical views of $\mathcal{A}$ in G_2 and $\mathcal{B}$ against UF-NMA. We see that $Time(\mathcal{B})$ is the time it takes to run $\mathcal{A}$ as a blackbox in G_2 where for every Q_H oracle query and Q_S signing query, no more than κ_m computations are performed. We have $Time(\mathcal{B}) = Time(\mathcal{A}) + \kappa_m(Q_H + Q_S) \approx Time(\mathcal{A})$, and thus see that Theorem 5.2.3 holds. □

LOSSY to UF-NMA

Theorem 5.2.4. *Let the ID-scheme ID be lossy and ϵ_{ls}-lossy sound. Then for any quantum adversary $\mathcal{A}$ playing the UF-NMA game issuing at most Q_H queries to the quantum random oracle H, there exists a quantum adversary $\mathcal{B}$ playing against LOSS, such that:*

$$Adv_{SIG}^{UF-NMA}(\mathcal{A}) \leq Adv_{ID}^{LOSS}(\mathcal{B}) + 8(Q_H + 1)^2 \cdot \epsilon_{ls}, \tag{5.5}$$

where: $Time(\mathcal{B}) = Time(\mathcal{A}) + Q_H \approx Time(\mathcal{A})$

Proof. We will now prove that a signature scheme $SIG := FS[ID, H, \kappa_m]$, where *ID* is a lossy ID-scheme, is UF-NMA secure. The reduction of games are described in Figure 5.8 and the adversaries and oracles are described in Figure 5.9. We let $\mathcal{A}$ be a quantum adversary playing UF-NMA against signature scheme *SIG*, where $\mathcal{A}$ makes at most Q_H quantum queries to the random oracle H.

Game G_0 : Looking at G_0 we notice that adversary $\mathcal{A}$ does not even have access to a signing oracle and to win the game $\mathcal{A}$ would have produce a valid message/signature making no signing

Game G_0 / $\boxed{G_1}$:

1 : $(pk, sk) \leftarrow IGen$

2 : $\boxed{pk \leftarrow LossyIGen}$

3 : $(M^*, \sigma^*) \leftarrow \mathcal{A}^{|H\rangle}(pk)$

4 : **parse :** $\sigma^* = (W^*, Z^*)$

5 : $c^* := H(W^* \parallel M^*)$

6 : **return** $Ver(pk, W^*, c^*, Z^*)$

All commands in regular print are executed from G_0 and throughout the games. Commands in print like this are executed only in G_0 and commands $\boxed{\text{printed like this}}$ are executed only in $\boxed{G_1}$.

Figure 5.8: G_0 and G_1 of Theorem 5.2.4.

queries and only having access to the random oracle H. It is apparent that G_0 is the UF-NMA game:

$$Adv_{SIG}^{UF-NMA}(\mathcal{A}) = \Pr[G_0^{\mathcal{A}} \Rightarrow 1]. \tag{5.6}$$

Game G_1 : It is easy to see that the only difference from G_0 to G_1 is that we now use the lossy key-generator algorithm *LossyIGen* instead of the real key-generator algorithm *IGen*. That is, in G_1 we have a lossy pk_{ls} with no corresponding sk. We consider adversary $\mathcal{B}$ simulating the oracle H using a $2Q_H$-wise hash function and thus adversary $\mathcal{A}$'s view in both games. We recall the *LOSS* advantage function (5.1) and see that:

$$|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_0^{\mathcal{A}} \Rightarrow 1]| = Adv_{ID}^{LOSS}(\mathcal{B}). \tag{5.7}$$

Before continuing, we note that our goal now is to reduce $\mathcal{A}$'s chance of winning in G_1 to the *GSPB*, such that:

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] \leq 8(Q_H + 1)^2 \cdot \epsilon_{ls}.$$

We also recall that for finite set S, the probabilistic algorithm $Uni(S)$ returns $x \leftarrow S$ uniformly over S, where the deterministic execution of the same algorithm is $x := Uni(S; r)$ given random tape r. We define the set of good challenges:

$$ChGOOD_{pk}(W) := \{c \in ChSet \mid \exists\, Z \in ZSet : Ver(pk, W, c, Z) = 1\},$$

containing all challenges c for which there exists a response Z such that (W, c, Z) is a valid transcript with respect to pk.

We now continue the proof. Just like we used adversary $\mathcal{B}$ to simulate $\mathcal{A}$'s view in order to deduct the *LOSS* advantage function, we will now use adversary $\mathcal{C}$ to simulate $\mathcal{A}$'s view in order to apply Lemma 2.1 and equate $\mathcal{A}$'s chance of winning G_1 to $\mathcal{C}$'s chance of winning the $GSPB_\lambda$ game. We let $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$, defined in Figure 5.9, be an unbounded quantum adversary and recall the $GSPB_\lambda$ game in Figure 2.1. The adversary $\mathcal{C}$ playing the $GSPB_\lambda$ game is allowed at most Q_H quantum queries to the quantum accessible oracle $g(\cdot)$.

Adversary $\mathcal{C}_1$ begins with fixing public key pk_{ls} using the *LossyIGen* key-generator and picking a $2Q_H$-wise hash function to simulate the random oracle H. Next, $\mathcal{C}_1$ computes the set of good

challenges for all possible $W \in \mathit{WSet}$ and defines the parameter $\lambda_{pk}(W)$, before defining the parameter $\lambda_{pk}(W \parallel W) := \lambda_{pk}(W)$ for every $M \in \mathit{MSet}$. Adversary $\mathcal{C}_1$ finishes with outputting the set $(\lambda_{pk}(W \parallel W))_{W \in \mathit{WSet}, M \in \mathit{MSet}}$. This process is very consuming and could take exponential time. As adversary $\mathcal{C}$ is unbounded we simply note what this process would require great computational power and move on to look at the probability that $\mathcal{C}$ succeeds when playing the GSPB_λ game.

With the set output by $\mathcal{C}_1$, the GSPB_λ game draws $g(W \parallel M) \leftarrow \mathfrak{B}_{\lambda_{pk}(W \parallel M)}$ for all possible $W \in \mathit{WSet}$ and $M \in \mathit{MSet}$. The second part of adversary $\mathcal{C}$, then uses adversary $\mathcal{A}$'s forgery in G_1. We see that $\mathcal{C}_2$ retrieves c^* by parsing the signature σ^* from the forgery and querying the oracle H with $H(W^* \parallel M^*)$. By GSPB, we know oracle $g(\cdot)$ outputs $y = 1$ with probability $\lambda_{pk}(W \parallel M) = \frac{|\mathit{ChGOOD}_{pk}(W)|}{|\mathit{ChSet}|}$. This ensures the output distribution of $H(W \parallel M)$, sampled in lines 2 and 3, being uniform over ChSet, just as in G_1. Further, using fixed random coins $f_{2Q_H}(W \parallel M)$, derived from the $2Q_H$-wise independent hash function f_{2Q_H}, when sampling c in the case $y = 0$, we assure consistency of H. We note that f_{2Q_H} looks like a perfectly random function to adversary $\mathcal{A}$.

Adversary $\mathcal{C}_1$

1 : $pk \leftarrow \mathit{LossyIGen}$
2 : pick $2Q_H$-wise independent hash function f_{2Q_H}
3 : for each $W \in \mathit{WSet}$:
4 : compute set $\mathit{ChGOOD}_{pk}(W) \subseteq \mathit{ChSet}$
5 : $\lambda_{pk}(W) := \dfrac{|\mathit{ChGOOD}_{pk}(W)|}{|\mathit{ChSet}|}$
6 : for each $M \in \mathit{MSet}$ set $\lambda_{pk}(W \parallel M) := \lambda_{pk}(W)$
7 : **return** $(\lambda_{pk}(W \parallel M))_{W \in \mathit{WSet}, M \in \mathit{MSet}}$

Adversary $\mathcal{C}_2^{|g\rangle}$

1 : $(M^*, \sigma^*) \leftarrow \mathcal{A}^{|H\rangle}(pk)$
2 : **Parse** $\sigma^* = (W^*, Z^*)$
3 : $c^* := H(W^* \parallel M^*)$
4 : **if** $\mathit{Ver}(pk, W^*, c^*, Z^*) = 1$, **then**
5 : **return** $(W^* \parallel M^*)$
6 : **else**
7 : **return** $\perp$

$H(W \parallel M)$ //quantum access

1 : $y := g(W \parallel M)$
2 : **if** $y = 1 : c := \mathit{Uni}\big(\mathit{ChGOOD}_{pk}(W); f_{2Q_H}(W \parallel M)\big)$
3 : **if** $y = 0 : c := \mathit{Uni}\left(\dfrac{\mathit{ChSet}}{\mathit{ChGOOD}_{pk}(W)}; f_{2Q_H}(W \parallel M)\right)$
4 : **return** c

Figure 5.9: Unbounded quantum adversary $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$ and quantum oracle H.

Looking closer at adversary $\mathcal{A}$'s forgery (M^*, σ^*) in G_1, we see that parsing σ^* into (W^*, Z^*) we can query the random oracle H to receive $c^* := H(W^*, M^*)$. If the verification algorithm accepts, namely $\mathit{Ver}(pk, W^*, c^*, Z^*) = 1$, then we know c^* is a "good" challenge, we have $c^* \in \mathit{ChGOOD}_{pk}(W^*)$. This implies $g(W^* \parallel M^*) = 1$ and we see that:

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid pk] = \Pr[\mathit{GSPB}_{\lambda_{pk}}^{\mathcal{C}} \Rightarrow 1] \leq 8(Q_H + 1)^2 \lambda_{pk}, \tag{5.8}$$

where

$$\lambda_{pk} = \max_{W \in WSet, M \in MSet} \lambda_{pk}(W \parallel M).$$

We observe that the probability of $\mathcal{A}$ creating a valid forgery with the lossy key pk_{ls} is the same as adversary $\mathcal{C}$ winning the $GSPB_\lambda$ game.

Averaging (5.8) over $pk \leftarrow LossyIGen$, and using the optimal adversary on (5.2), we get:

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] \leq 8(Q_H + 1)^2 \cdot \mathbf{E}_{pk}[\lambda_{pk}] \leq 8(Q_H + 1)^2 \epsilon_{ls}. \tag{5.9}$$

Now combining the results we have seen through (5.6), (5.7) and lastly (5.9), we get:

$$Adv_{SIG}^{UF-NMA}(\mathcal{A}) \leq Adv_{ID}^{LOSS}(\mathcal{B}) + 8(Q_H + 1)^2 \cdot \epsilon_{ls}.$$

We recognize this as (5.5) from Theorem 5.2.4 and consider the Theorem to be proved. □

Further, presenting the results from proving 5.2.3 and 5.2.4, namely (5.4) and (5.5), we get:

$$Adv_{SIG}^{UF-CMA_1}(\mathcal{A}) \leq Adv_{ID}^{LOSS}(\mathcal{B}) + 8(Q_H + 1)^2 \cdot \epsilon_{ls} + \kappa_m Q_S \cdot \epsilon_{zk} + 2^{-\alpha+1}.$$

Which, as no surprise, we recognize as (5.3) from Theorem 5.2.2. Thus, we see that Theorem 5.2.2 holds.

Strong Unforgeability

In proving the UF-CMA$_1$ security from UF-NMA security we notice that this can be extended to proving sUF-CMA$_1$ security. To do so requires changing the winning condition of the games in Figure 5.6, in order to account for the strong unforgeability requirement. For the sUF-CMA$_1$ proof the return line of the games, line 7, would be $[\![(M^*, \sigma^*) \notin MSet]\!] \wedge [\![Ver(pk, M^*, \sigma^*)]\!]$, recording all previous message/signature queries.

The proof conducts in the same fashion as we explored in detail for UF-CMA$_1$ security, and we therefore refer to Kiltz et al. [7] for the full explanation. We here note that for a UF-CMA$_1$ signature to also be sUF-CMA$_1$ secure, the underlying ID-scheme has to have Computational Unique Response (CUR).

Definition 5.2.5. If ID-scheme ID has *computational unique response*, we associate the advantage function for an adversary $\mathcal{A}$:

$$Adv_{ID}^{CUR}(\mathcal{A}) := \Pr\left[\begin{matrix} Ver(pk, W, c, Z) = 1 \wedge \\ Ver(pk, c, W, Z') = 1 \wedge Z \neq Z' \end{matrix} \;\middle|\; \begin{matrix} (pk, sk) \leftarrow IGen; \\ (W, c, Z, Z') \leftarrow \mathcal{A}(pk) \end{matrix}\right] \tag{5.10}$$

We note that for an adversary $\mathcal{A}$ playing the sUF-CMA$_1$ game, this would be equivalent to finding a valid signature on an already queried message. In other words, having received signature

$\sigma_{M^*} = (W_{M^*}, Z_{M^*})$ on message M, $\mathcal{A}$ would have to produce another forgery on $\sigma^* = (W^*, Z^*)$ on M, such that transcripts (W^*, c^*, Z^*) and $(W_{M^*}, c_{M^*}, Z_{M^*})$, where $Z^* \neq Z_{M^*}$ both are valid. We introduce Theorem 5.2.6 for sUF-CMA$_1$ security.

Theorem 5.2.6. *Let the ID-scheme ID be ϵ_{zk}-perfect naHVZK, ϵ_{ls}-lossy sound and having α bits of min-entropy. Then for any quantum adversary $\mathcal{A}$ playing the sUF-CMA$_1$ game issuing at most Q_H queries to the quantum random oracle H and Q_S classical queries to the signing oracle SIGN$_1$, there exists a quantum adversary $\mathcal{B}$ and a quantum adversary $\mathcal{C}$, such that:*

$$Adv_{SIG}^{sUF-CMA_1}(\mathcal{A}) \leq Adv_{ID}^{LOSS}(\mathcal{B}) + 8(Q_H+1)^2 \cdot \epsilon_{ls} + \kappa_m Q_S \cdot \epsilon_{zk} + 2^{-\alpha+1} + Adv(\mathcal{C})_{ID}^{CUR},$$

where $Time(\mathcal{C}) = Time(\mathcal{B}) = Time(\mathcal{A}) + \kappa_m Q_H \approx Time(\mathcal{A})$.

UF-CMA$_1$ to UF-CMA

We also want to note the results of Bellare et al. [23]. Which is that any UF-CMA$_1$ signature can be proven UF-CMA with the use of a pseudorandom function and by defining $Sign'((sk, K), M) := Sign(sk, M; PRF_K(M))$. What we are describing are signatures built through the deterministic Fiat-Shamir with aborts transformation.

$Sign(sk, M)$	$DSign((sk, K), M)$
$\kappa := 0$	$\kappa := 0$
while $Z = \perp$ **and** $\kappa \leq \kappa_m$ **do**	**while** $Z = \perp$ **and** $\kappa \leq \kappa_m$ **do**
$\quad \kappa := \kappa + 1$	$\quad \kappa := \kappa + 1$
$\quad (W, St) := P_1(sk)$	$\quad (W, St) := P_1(sk; PRF_K(0 \parallel m \parallel \kappa)$
$\quad c = H(W \parallel M)$	$\quad c = H(W \parallel M)$
$\quad Z := P_2(sk, W, c, St)$	$\quad Z := P_2(sk, W, c, St; PRF_K(1 \parallel m \parallel \kappa)$
if $Z = \perp$, **then**	**if** $Z = \perp$, **then**
$\quad$ **return** $\sigma = \perp$	$\quad$ **return** $\sigma = \perp$
return $\sigma = (W, Z)$	**return** $\sigma = (W, Z)$

Figure 5.10: The signing algorithm *Sign* for signature $SIG := FS[ID, H, \kappa_m]$ and deterministic signing algorithm for signature $DSIG := DFS[ID, H, PRF, \kappa_m]$.

Definition 5.2.7. Let $ID := (IGen, P, V)$. Using the *deterministic Fiat-Shamir with aborts transformation* on *ID* gives us the signature scheme $DSIG := (IGen, DSign, Ver)$, denoted $DFS[ID, H, PRF, \kappa_m]$. Here *ID* is the underlying ID-scheme, $H : \{0,1\}^* \to ChSet$ is the hash function used for challenges, κ_m is the maximum amount of times the scheme has to run in order to produce a valid transcript and *DSign*, described in Figure 5.10, is the signing algorithm derandomized using the pseudorandom function *PRF* with random key K.

With this definition we go back to the flowchart in Figure 5.2 and notice that proving UF-CMA using the deterministic Fiat-Shamir variant offers better tightness. To use this advantage we

extend Theorem 5.2.2 using pseudorandom function PRF and get:

$$Adv_{DSIG}^{UF-CMA}(\mathcal{A}) \leq Adv_{ID}^{LOSS}(\mathcal{B}) + 8(Q_H + 1)^2 \cdot \epsilon_{ls} + Adv_{PRF}^{PR}(\mathcal{D}) + \kappa_m Q_S \cdot \epsilon_{zk} + 2^{-\alpha+1},$$

where $\mathcal{D}$ is a quantum adversary with classical access to PRF. Bellare et al. [23] conducted this prof in the classical ROM, however, because of $\mathcal{D}$'s limited access to the PRF this proof also holds in a quantum setting. We note that, as described in the previous section, this can of course be extended to sUF-CMA.

Chapter 6

Two Signatures

Having introduced both ID-schemes and signature schemes, as well as how one proves security in the ROM and QROM we will in this chapter look at two concrete signature schemes. We shall look at Lyubashevsky's signature and the CRYSTAL's Dilithium signature. Both these are Fiat-Shamir derived signatures and considered to be safe in a post-quantum setting. Here we shall look at Lyubashevsky's security in the ROM and Dilithium's security in the QROM.

A significant, yet subtle difference of the signatures is how we consider them to be safe against quantum attackers. Lyubashevsky is considered safe against quantum attackers due to the fact that the underlying problem, SVP_γ, is quantum safe. The proof is conducted in the ROM, with no quantum adversaries, but the logic is that as the signature is safe and the underlying problem is quantum safe, then the signature is quantum safe. Dilithium is considered safe against quantum attackers due to the fact that the proof is conducted in the QROM. The QROM considers quantum adversaries and thus signatures with a QROM proof are safe against quantum attackers.

6.1 Lyubashevsky

Vadim Lyubashevsky's signature scheme [26] is a lattice based signature scheme that is considered to be safe in both a classical and quantum setting. The signature is often referred to as the *Fiat-Shamir with Aborts* signature, but to not confuse it with the transformation we will refer to this signature simply as the *Lyubashevsky* signature. We will first present the ID-Scheme and from there construct the Signature Scheme. We shall look at the scheme in the ROM here. As the main focus of this thesis is the QROM we will look at the big picture of the signature and refer the reader to the article cited above for details.

6.1.1 ID-Scheme

The ID-Scheme is initiated by the key-generation algorithm giving us secret key sk and public key pk . The secret key is a set of m polynomials drawn uniformly random from the set D_s, where $D \subseteq R_q$. The corresponding pk consists of the hash from our hash family $\mathcal{H}(R_q, D, m)$ and the hash of the random value used in the secret key, namely $h(\vec{s})$. We get the key pair $sk := \vec{s}$ and $pk := (h, h(\vec{s}))$. We further define the parameters of the scheme in figure 7.1, where we also offer two instantiations to see the security.

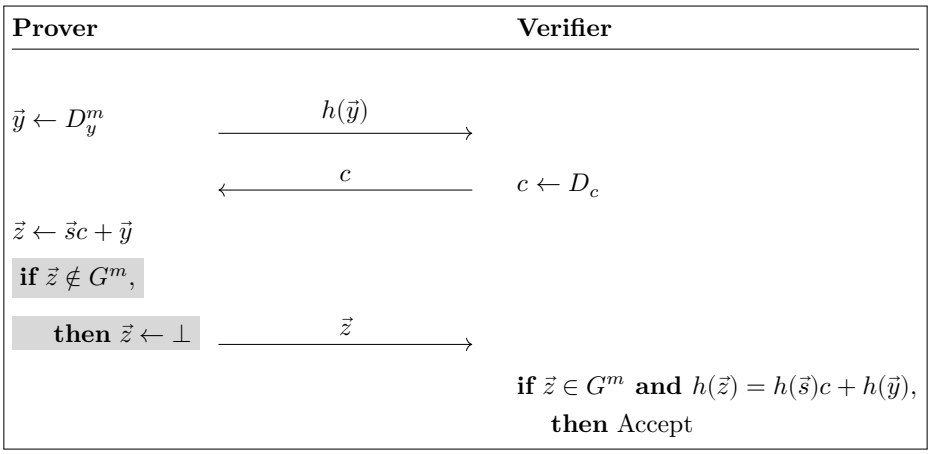

Figure 6.1: An interactive execution of the Lyubashevsky ID-scheme.

The rest of the scheme then runs in the familiar sigma fashion. The prover picks a random value $\vec{y} \leftarrow D_y^m$ and commits it to the verifier as $h(\vec{y})$. The verifier returns challenge $c \leftarrow D_c$. The prover computes $\vec{z} = \vec{s}c + \vec{y}$ and checks whether it falls in the correct range, namely whether $\vec{z} \in G^m$. If in the desired range the prover sends $\vec{z}$ to the verifier and if not the prover aborts. The verifier is now at the final step of the protocol and checks if $\vec{z} \in G^m$ and if $h(\vec{z}) = h(\vec{s})c + h(\vec{y})$. If both statements are true, the verifier will accept and output 1, and if one or both are false the verifier will reject and output 0.

A diagram of the ID-scheme executed between a prover and a verifier is given in Figure 6.1. The key-generation algorithm initiating the protocol is the same as the one we will use for the signature scheme and can be seen in Figure 6.2.

Correctness

Considering the homomorphic properties of the hashes in our hash function family, (3.3) and (3.4), it becomes apparent that a transcript produced by a honest prover, that does not abort,

will be accepted. We have correctness:

$$h(\vec{z}) = h(\vec{s}c + \vec{y}) = h(\vec{s}c) + h(\vec{y}) = h(\vec{s})c + h(\vec{y}).$$

Important to notice is that we can have correctness even if the prover aborts. For Lyubashevsky it is expected that the scheme is executed less than three times before a transcript is produced.

Witness Indistinguishability

To see that we have WI we consider secret keys $sk = \vec{s}$ and $sk' = \vec{s}'$, where $h(\vec{s}) = h(\vec{s}')$. We note that the parameter are set such that for every $\vec{s} \in D_s^m$, every $c \in D_c$ and every $\vec{z} \in G^m$, the value $\vec{z} - \vec{s}c$ is contained in D_y. This means that even an adversary having seen the history $(h(\vec{y}), c, \vec{z})$, is unable to tell whether $\vec{z}$ is computed using $sk = \vec{s}$ and masking parameter $\vec{y}$, or computed using $sk = \vec{s}'$ and masking parameter $\vec{y}' = \vec{z} - \vec{s}'c = \vec{y} + \vec{s}c - \vec{s}'c = \vec{y} + (\vec{s} - \vec{s}')c$. This is because both $h(\vec{s}) = h(\vec{s}')$ and $h(\vec{y}) = h(\vec{y}')$.

6.1.2 Signature Scheme

The Fiat-Shamir with aborts transformation was developed by Lyubashevsky and introduced through the article we are working with here. It is clear that the Lyubashevsky signature is a $FS[ID, H, \kappa_m]$ signature — it is actually the very first. The signature is initiated the same way as the ID-scheme using the key-generation algorithm *IGen*, while the interactive protocol gets split between the signing and verification algorithms *Sign* and *Ver*. All the procedures of the signature scheme can be seen in Figure 6.2.

IGen		*Sign*$(\mu, h, \vec{s})$	
1 :	$\vec{s} \leftarrow D_s^m$	1 :	$\vec{y} \leftarrow D_y^m$
	$sk := \vec{s}$	2 :	$c \leftarrow H(h(\vec{y}), \mu)$
2 :	$h \leftarrow H(R, D, m)$	3 :	$\vec{z} \leftarrow \vec{s}c + \vec{y}$
	$pk := (h, h(\vec{s}))$	4 :	**if** $\vec{z} \notin G^m$ **then**
3 :	**return** (sk, pk)		go to step 1
		5 :	**return** $(\vec{z}, c)$

Ver$(\mu, \vec{z}, c, h, h(\vec{s}))$

1 : **return** $[\![\vec{z} \in G^m]\!] \wedge [\![c = H(h(\vec{z}) - h(\vec{s})c, \mu)]\!]$

Figure 6.2: The Lyubashevsky signature scheme. The key-generation algorithm *IGen* is the same for both the ID and signature scheme.

Looking at the signature scheme, we see that the prover now receives the challenge c from the

hash function $H : \{0,1\}^* \to D_c$ modeled as a RO. The prover queries the RO with the hash of the commitment $h(\vec{y})$ and the message μ. The RO outputs $c := H(h(\vec{y}), \mu)$ to the query. With the challenge from the RO, the prover now computes $\vec{z} = \vec{s}c + \vec{y}$. If $\vec{z} \in G^m$ the prover sends the pair $(\vec{z}, c)$ to the verifier, if $\vec{z} \notin G^m$ the prover aborts. Upon receiving the pair $(\vec{z}, c)$, the verifier checks if $\vec{z} \in G^m$ and if $c = H(h(\vec{z}) - h(\vec{s})c, \mu)$. If both statements are true the verifier accepts and outputs 1, and otherwise rejects and outputs 0.

6.1.3 Security

As we mentioned in the beginning of this section, we will not go into great detail on the security proof. However, we still wish to present an outline of what the security proof looks like, and refer the reader to Lyubashevsky [26].

Sketch of Security Proof

To show the security of Lyubashevsky, we begin with showing that adversary $\mathcal{A}$ capable of breaking the ID-scheme can be used to solve the collision problem, Definition 3.3.2.

Given random $h \leftarrow \mathcal{H}(R_q, D, m)$, we generate keys $(sk, pk) \leftarrow IGen$. The proof continues in a manner allowing us to extract two challenge/response pairs $(c, \vec{z})$ and $(c', \vec{z}')$ such that along with secret key sk we have $h(\vec{z} - \vec{s}c) = h(\vec{z}' - \vec{s}c')$. We have set the parameters such that both $(\vec{z} - \vec{s}c)$ and $(\vec{z}' - \vec{s}c')$ are contained in D. Having WI, we know adversary $\mathcal{A}$ will not be able to distinguish which secret key is being used. The probability of $\vec{z} - \vec{s}c$ and $\vec{z}' - \vec{s}c'$ being distinct is at least $1/2$ and we have a collision for h. It becomes apparent that an adversary like $\mathcal{A}$ breaking the ID-scheme can be used to solve the $Col(h, D)$ problem for random $h \leftarrow \mathcal{H}(R_q, D, m)$. Thus, by Theorem 6.1.1, $\mathcal{A}$ can solve SVP_γ in all $(\vec{x}^n + 1)$-cyclic lattices.

Theorem 6.1.1. *Let $D = \{y \in R_q : \|y\|_\infty \leq d\}$, where $d \in \mathbb{Z}$. Let $\mathcal{H}(R_q, D, m)$ be our hash function family where $m > \frac{\log p}{\log 2d}$ and $p \geq 4dmn^{1.5} \log 2d$. If there exists a polynomial time algorithm solving the $Col(h, D)$ problem for random $h \in \mathcal{H}(R_q, D, m)$ with non-negligible probability, then there exists an algorithm solving the SVP_γ problem for every $(x^n + 1)$-cyclic lattice Λ , where $\gamma = 16dmn \log 2d$.*

For the singature scheme we know that the WI follows directly from the ID-scheme, and the security proof is similar to that of the ID-scheme. Using the *Forking Lemma* we obtain two signatures on the same RO query, we refer to Pointcheval and Stern [21] for the Lemma and it's proof. The sUF-CMA security of the Lyubashevsky signature is then shown through Theorem 6.1.2, essentially saying that the security comes down to the hardness of the SVP_γ.

Theorem 6.1.2. *If the signature scheme in Figure 6.2 is not sUF-CMA secure, then there exists a polynomial time algorithm $\mathcal{A}$ that solves $SVP_\gamma(\Lambda)$ for $\gamma = \tilde{O}(n^2)$ for every lattice Λ corresponding to an ideal in the ring $\mathbb{Z}[X]/\langle X^n + 1\rangle$.*

6.2 CRYSTAL's Dilithium

CRYSTAL's Dilithium is a signature scheme that is based on the learning with errors (LWE) problem [27]. The signature is designed to be fast and efficient, as well as secure against both classical and quantum attacks. It is one of the signature schemes that have been recommended by the National Institute of Standards and Technology (NIST) for use in post-quantum cryptography (PQC).

Similarly to Lyubashevsky the Dilithium signature is constructed from an ID-Scheme using Fiat-Shamir with Aborts. The version we will introduce here differs slightly to that of Ducas et al. [27]. As we wish to prove it safe in the QROM we need the underlying ID-scheme to be lossy and thus follow the construction of Dilithium presented by Kiltz et al. [7]. This approach results in an increase in the total size of the public key and signature by a factor of a little more than 3 to that of the original scheme.

We begin by introducing the ID-scheme, before we look closer at the supporting algorithms. We then look at some significant properties of the ID-scheme, before we move on to the signature scheme and its security proof. We shall prove Dilithium safe in the QROM, following the approach of Kiltz et al. [7] and the proof we presented in Chapter 5.

6.2.1 ID-Scheme

The ID-scheme consists of algorithms $ID := (IGen, P = (P_1, P_2), V)$ described in Figure 6.3. We define the spaces for the scheme as:

The set of commitments: $WSet := \{\vec{w}_1 : \exists\, \vec{y} \in S^l_{\gamma'-1} \text{ s.t. } \vec{w}_1 = HighBits_q(\mathbf{A}\vec{y}, 2\gamma)\}$

The set of responses: $ZSet := S^l_{\gamma'-\beta-1} \times \{0,1\}^k$.

The set of challenges: $ChSet := \{c \in R \mid \|c\|_\infty = 1 \text{ and } \|c\| = \sqrt{46}\}$.

Noting that the challenge space consists of elements in the ring $R = \mathbb{Z}[X]/\langle X^n + 1\rangle$, with coefficients -1, 0 or 1, where exactly 46 are non-zero. We observe that the size of this set then becomes $\binom{n}{46} \cdot 2^{46}$. We further define the parameters of the scheme in figure 7.1, where we also offer two instantiations to see the security.

The key-generation algorithm *IGen* initiates the scheme by generating sk and pk. Random seed ρ is drawn and expanded to the matrix $\mathbf{A} \in R_q^{k\times l}$ using the XOF *Sam* modeled as a RO. The secret keys $(\vec{s_1}, \vec{s_2}) \leftarrow S^l_\eta \times S^k_\eta$ have uniformly random coefficients ranging from $-\gamma$ to γ. Further, $\vec{t} := \mathbf{A}\vec{s_1} + \vec{s_2}$ is computed, we note that $\vec{t} = \vec{t_1} \cdot 2^d + \vec{t_0}$ for some small $\vec{t_0}$. It is assumed that $\vec{t_0}$ is known to the adversary, however as it is not necessary for verification only $\vec{t_1}$ is included in the public key. *IGen* outputs $sk := (\rho, \vec{s_1}, \vec{s_2}, \vec{t_0})$ and $pk := (\rho, \vec{t_1})$.

With access to the keys, the prover and the verifier begin their communication. The prover

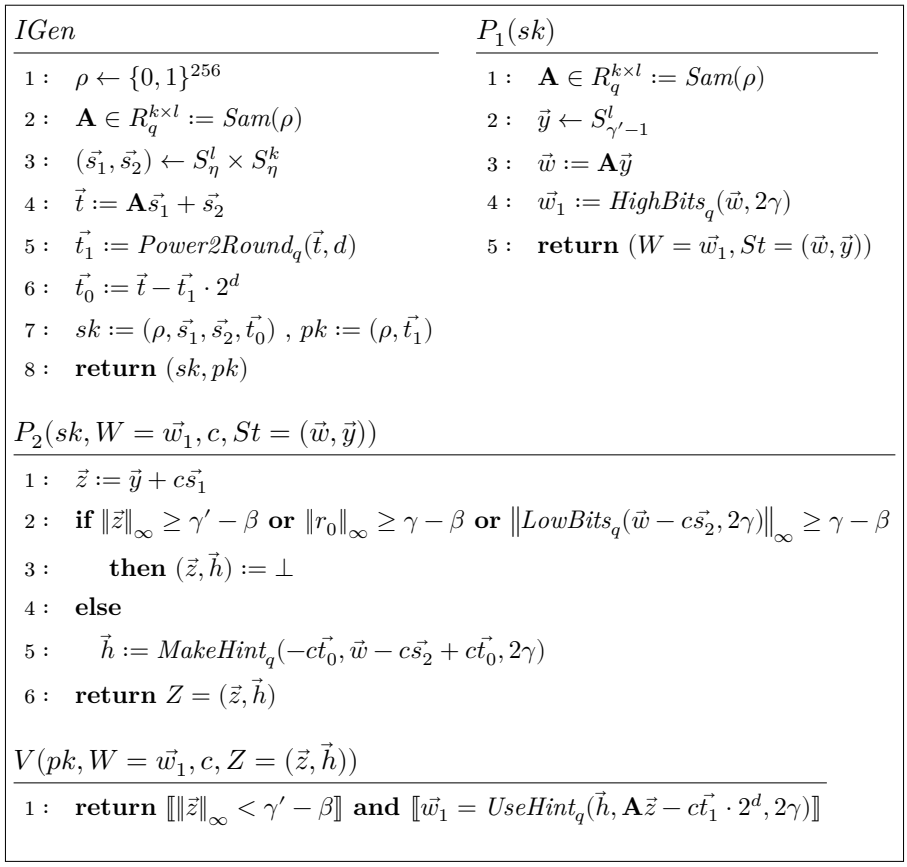

Figure 6.3: The Dilithium ID-Scheme

executes P_1 and reconstructs $\mathbf{A} \in R_q^{k\times l}$ by querying the RO Sam with random seed ρ, before continuing by sampling $\vec{y} \leftarrow S^l_{\gamma'-1}$ to compute $\vec{w} = \mathbf{A}\vec{y}$. The prover sets $\vec{w} = 2\gamma \cdot \vec{w_1} + \vec{w_0}$ (looking at the P_1 algorithm and following the supporting algorithms, it can be seen that this holds), where $\vec{w_0}$ ranges from $-\gamma$ to γ. The prover then sends $\vec{w_1}$ to the verifier.

The verifier samples challenge $c \leftarrow ChSet$ and returns this. The prover initiates P_2 and computes $\vec{z} := \vec{y} + c\vec{s}$ and checks whether the computed vector is in the desired range. If either $\vec{z} \notin S^l_{\gamma'-\beta-1}$ or $LowBits_q(\vec{w} - c\vec{s_2}, 2\gamma) \notin S^l_{\gamma'-\beta-1}$ the prover sets $(\vec{z}, \vec{h}) = \perp$, before sending the response $Z := (\vec{z}, \vec{h})$ to the verifier. Checking the lower bits is directly necessary for the security as we check whether $\vec{z}$ could reveal anything about the secret key.

We will show correction in Section 6.2.3, but we point out that if the prover does not set $Z = \perp$ and actually computes the hint, then $\vec{w_1} = HighBits_q(\mathbf{A}\vec{z} - c\vec{t}, 2\gamma)$. The verifier checks whether $\|\vec{z}\|_\infty \leq \gamma' - \beta$ and equipped with the hint $\vec{h}$ from the prover reconstructs and checks $\vec{w_1}$. Reconstructing $\vec{w_1}$ is not necessary for the security of the scheme, as the verifier could have simply checked that $\mathbf{A}\vec{z} - c\vec{t_1} \cdot 2^d \approx \vec{w_1}$. However, it reduces the communication size in the

Fiat-Shamir transformation if the ID-scheme is commitment recoverable.

6.2.2 Supporting Algorithms

One of the most attractive aspects of the Dilithium scheme is the small size of the public key. This comes from the supporting algorithms $HighBits_q$ and $LowBits_q$, allowing us to extract the higher and lower ordered bits from elements in $\mathbb{Z}_q$. More exactly we can recover the higher order bits of $r + z$ for $r, z \in \mathbb{Z}_q$ where z is small, without having to store z. Two useful Lemmas are presented below and all supporting algorithms are presented in Figure 6.4. Before we dive deeper, note that we consider the polynomials coefficient-wise when working with the algorithms and thus conduct the proofs for integers.

Lemma 6.2.1. *Suppose q and α are positive integers, such that $q > 2\alpha$, $q \equiv 1$ mod α, where α is even. Let $\vec{r}$ and $\vec{z}$ be vectors of elements in the ring R_q, where $\|z\|_\infty \leq \alpha/2$, and let $\vec{h}, \vec{h'}$ be vectors of bits. Then the algorithms $HighBits_q$, $MakeHint_q$ and $UseHint_q$, satisfy the following properties:*

i $UseHint_q(MakeHint_q(\vec{z}, \vec{r}, \alpha), \vec{r}, \alpha)) = HighBits_q(\vec{r} + \vec{z}, \alpha)$.
ii *Let* $\vec{v_1} = UseHint_q(\vec{h}, \vec{r}, \alpha)$, *then* $\|\vec{r} - \vec{v_1} \cdot \alpha\|_\infty \leq \alpha + 1$.
iii *For any* $\vec{h}, \vec{h'}$, *if* $UseHint_q(\vec{h}, \vec{r}, \alpha) = UseHint_q(\vec{h'}, \vec{r}, \alpha)$ *then* $\vec{h} = \vec{h'}$.

Proof. i) Let $r, z \in \mathbb{Z}_q$ and $\|z\|_\infty \leq \alpha/2$ and define $m := (q-1)/\alpha$. We recall that the purpose of the $MakeHint_q$ and $UseHint_q$ procedures is to provide a hint on the higher bits of $r + z$ without storing z.

We begin looking at the $MakeHint_q(\vec{z}, \vec{r}, \alpha)$ on the left hand side of the equation. This procedure calls the $HighBits_q$ procedure and defines $r_1 := HighBits_q(r, \alpha)$ and $v_1 := HighBits(r + z, \alpha)$ and outputs $h = 0$ if $r_1 = v_1$ and $h = 1$ if $r_1 \neq v_1$. The $UseHint_q$ procedure calls the $Decompose_q$ procedure and retrieves (r_1, r_0), we get three possible scenarios for the left hand side:

Case 1: If $h = 0$, then the left hand side simply outputs the r_1 retrieved from $Decompose_q$.

Case 2: If $h = 1$, the $UseHint_q$ checks the r_0 retrieved from $Decompose_q$. If $r_0 > 0$ then $UseHint_q$ outputs $r_1 := (r_1 + 1) \bmod {}^+ m$.

Case 3: If $h = 1$, the $UseHint_q$ checks the r_0 retrieved from $Decompose_q$. If $r_0 \leq 0$ it outputs $r_1 := (r_1 - 1) \bmod {}^+ m$.

Now, looking at the right hand side of the equation, where $HighBits_q(r+z, \alpha)$ calls $Decompose_q(r+z, \alpha)$ which outputs (r_0, r_1), where $0 \leq r_1 < m$ and $\|r_0\|_\infty \leq \alpha/2$. As $\|z\|_\infty \leq \alpha/2$ we see that $HighBits_q$ calls $Decompose_q$ to retrieve r_1 which it then outputs. Looking at the definition of v_1 from calling $MakeHint_q(\vec{z}, \vec{r}, \alpha)$ on the left hand side, we see that the r_1 output by $HighBits_q(r + z, \alpha)$ relates to this v_1 in two possible ways:

Case 1: If $r_0 > 0$, then $-\alpha/2 < r_0 + z \leq \alpha$ which means $v_1 = r_1$ or $v_1 = (r_1 + 1) \bmod m$.

Case 2: If $r_0 \leq 0$, then $-\alpha < r_0 + z \leq \alpha/2$ which means $v_1 = r_1$ or $v_1 = (r_1 - 1) \bmod m$.

It is clear that in the case $v_1 \neq r_1$ the first case on the left hand side will output the same as the two cases on the right hand side depending on r_0. If $v_1 = r_1$ it is clear that the last two cases on the left hand side also equals the two cases on the right hand side. It is apparent that we have equality and have thus proved *i)*.

ii) Let $(h, r) \in \{0, 1\} \times \mathbb{Z}_q$ and $v_1 := \mathit{UseHint}_q(h, r, \alpha)$. We consider the three possible cases for $\mathit{UseHint}_q(h, r, \alpha)$:

Case 1: If $h = 0$, then $\mathit{UseHint}_q(0, r, \alpha)$ outputs $r_1 = v_1$ and we see that $\mathit{Decompose}_q$ computes:

$$r - v_1 \cdot \alpha = r_1 \cdot \alpha + r_0 - r_1 \cdot \alpha = r_0,$$

which by definition has absolute value $\alpha/2$.

Case 2: If $h = 1$ we go on to check r_0. If and $r_0 > 0$, then $v_1 = r_1 + 1 - \kappa \cdot (q-1)/\alpha$, where $\kappa = 0$ or 1. We see that:

$$\begin{aligned} r - v_1 \cdot \alpha &= r_1 \cdot \alpha + r_0 - (r_1 + 1 - \kappa \cdot (q-1)/\alpha) \cdot \alpha \\ &= \alpha + r_0 - \kappa \cdot (q-1). \end{aligned}$$

Using centered reduction modulo we see that the absolute value is no greater than α.

Case 3 If $r_0 \leq 0$, then giving us $v_1 = r_1 - 1 + \kappa \cdot (q-1)/\alpha$, where $\kappa = 0$ or $\kappa = 1$. We see that:

$$\begin{aligned} r - v_1 \cdot \alpha &= r_1 \cdot \alpha + r_0 - (r_1 + 1\kappa \cdot (q-1)/\alpha) \cdot \alpha \\ &= -\alpha + r_0 + \kappa \cdot (q-1). \end{aligned}$$

Using centered reduction modulo we see that the absolute value is no greater than $\alpha + 1$.

We see that for both $h = 0$ and $h = 1$ and for all values of r_0, the absolute value of $r - v_1 \cdot \alpha$ is less than $\alpha + 1$. We have proved *ii)*.

iii) Let $r \in \mathbb{Z}_q$ and $h, h' \in \{0, 1\}$. We consider the two possible values for h, namely 0 and 1.

We see that $\mathit{UseHint}_q(0, r, \alpha) = r_1$ and $\mathit{UseHint}_q(1, r, \alpha) = (r_1 \pm 1) \bmod {}^+((q-1)/\alpha)$. As $(q-1)/\alpha \geq 2$, it is clear that $r_1 \neq (r_1 \pm 1) \bmod {}^+((q-1)/\alpha)$. We see that $\mathit{UseHint}_q(h, r, \alpha) = \mathit{UseHint}_q(h', r, \alpha)$ implies $h = h'$, thus proving *iii)*.

Having proved *i)*, *ii)* and *iii)* we see that Lemma 6.2.1 holds. □

Lemma 6.2.2. *If* $\|\vec{s}\|_\infty \leq \beta$ *and* $\left\|\mathit{LowBits}_q(\vec{r}, \alpha)\right\|_\infty < \frac{\alpha}{2} - \beta$ *, then:*

$$\mathit{HighBits}_q(\vec{r}, \alpha) = \mathit{HighBits}_q(\vec{r} + \vec{s}, \alpha).$$

Proof. Assume $\|s\|_\infty \leq \beta$ and $\left\|LowBits_q(r,\alpha)\right\|_\infty < \frac{\alpha}{2} - \beta$. As $LowBits_q(r,\alpha)$ calls on the decompose procedure we see that for $Decompose_q(r,\alpha)$ we have $r = r_1 \cdot \alpha + r_0$ where $-\alpha/2 + \beta < r_0 \leq \alpha/2 + \beta$. For $Decompose_q(r+s,\alpha)$ we have $r = r_1 \cdot \alpha + r_0$ where $-\alpha/2 + \beta < r_0 \leq \alpha/2 + \beta$. Thus; $r + s = r_1 \cdot \alpha + r_0 + s$ and:

$$(r+s) - ((r+s) \bmod^{\pm} \alpha) = r_1 \cdot \alpha = r - (r \bmod^{\pm} \alpha),$$

which is the claim of the Lemma, and we see that it holds. □

$\underline{UseHint_q(h,r,\alpha)}$
$m := (q-1)/\alpha$
$(r_1, r_0) := Decompose_q(r,\alpha)$
if $h = 1$ **and** $r_0 > 0$, **then**
  $r_1 := (r_1 + 1) \bmod^{+} m$
if $h = 1$ **and** $r_0 \leq 0$, **then**
  $r_1 := (r_1 - 1) \bmod^{+} m$
return r_1

$\underline{HighBits_q(r,\alpha)}$
$(r_1, r_0) := Decompose_q(r,\alpha)$
return r_1

$\underline{Decompose_q(r,\alpha)}$
$r := r \bmod^{+} q$
$r_0 := r \bmod^{\pm} \alpha$
if $r - r_0 = q - 1$, **then**
  $r_1 := 0, r_0 := r_0 - 1$
else $r_1 := (r - r_0)/\alpha$
return (r_1, r_0)

$\underline{LowBits_q(r,\alpha)}$
$(r_1, r_0) := Decompose_q(r,\alpha)$
return r_0

$\underline{Power2Round_q(r,d)}$
$r := r \bmod^{+} q$
$r_0 := r \bmod^{\pm} q$
return $((r - r_0)/2^d, r_0)$

$\underline{MakeHint_q(z,r,\alpha)}$
$r_1 := HighBits(r,\alpha)$
$v_1 := HighBits(r+z,\alpha)$
return $[\![r_1 \neq v_1]\!]$

Figure 6.4: The supporting algorithms of Dilithium.

6.2.3 Significant Properties

In this section we will look at some significant properties of the Dilithium ID-scheme. This is an important step of proving Dilithium's unforgeability, which we will see in Chapter 6.2.5.

Correctness

To see the correctness of the scheme we will show that the verification algorithm always accepts if a prover sends a transcript such that $Z \neq \perp$, as well as look at the probability that the prover returns $\perp$, namely the correctness error δ.

We begin by looking at the probability that $\|\vec{z}\|_\infty < \gamma' - \beta$. We will do this by looking at the coefficients separately. For each coefficient of $\vec{cs_1}$, the corresponding coefficient of $\vec{z}$ will be (inclusively) between $-\gamma' + \beta + 1$ and $\gamma' - \beta - 1$. We get a range with size $2(\gamma' - \beta) - 1$, giving

the coefficients of $\vec{y}$ exactly $2\gamma' - \beta$ possibilities. We get the probability:

$$\left(\frac{2(\gamma'-\beta)-1}{2\gamma'-1}\right)^{nl} = \left(1-\frac{\beta}{\gamma'}\right)^{nl} \approx e^{-\beta nl/\gamma'}, \tag{6.1}$$

where we have approximate equality as $\beta \ll \gamma$. Next, we want to find the probability that $\left\|LowBits_q(\vec{w} - c\vec{s_2}, 2\gamma)\right\|_\infty < \gamma - \beta$. We assume the low order bits to be uniformly distributed modulo 2γ. We have:

$$\left(\frac{2(\gamma-\beta)-1}{2\gamma}\right)^{nk} \approx e^{-\beta nk/\gamma}, \tag{6.2}$$

using the fact that our β values are large compared to $1/2$. It is now easy to find the total probability the prover returns $\vec{z} \neq \perp$, we multiply (6.1) and (6.2) and see that:

$$\left(e^{-\beta\frac{nl}{\gamma'}}\right) \cdot \left(e^{-\beta\frac{nk}{\gamma}}\right) = e^{-\beta n\cdot(k/\gamma + l/\gamma')}.$$

Thus, the correctness error is $\delta = 1 - e^{-\beta n\cdot(k/\gamma+l/\gamma')}$. It now only remains to show that $\vec{z} \neq \perp$ means the verifier always accept. We assume $\vec{z} \neq \perp$. Clearly the response will always pass the verifier's check on $\|\vec{z}\|_\infty < \gamma' - \beta$. It remains to check $\vec{w_1} = UseHint_q(\vec{h}, \mathbf{A}\vec{z} - c\vec{t}\cdot 2^d, 2\gamma)$, and begin by noticing:

$$\vec{w} - c\vec{s_2} = \mathbf{A}\vec{y} - c\vec{s_2} = \mathbf{A}(\vec{z} - c\vec{s_1}) - c\vec{s_2} = \mathbf{A}\vec{z} - c\vec{t} = \mathbf{A}\vec{z} - c\vec{t_0} - c\vec{t_1}\cdot 2^d.$$

Now, by Lemma 6.2.1 and knowing $\left\|c\vec{t_0}\right\|_\infty < \gamma$, we see that:

$$\begin{aligned} UseHint_q(\vec{h}, \mathbf{A}\vec{z} - c\vec{t_1}\cdot 2^d, 2\gamma) &= HighBits_q(\mathbf{A}\vec{z} - c\vec{t_1}\cdot 2^d, 2\gamma) \\ &= HighBits_q(\vec{w} - c\vec{s_2}, 2\gamma). \end{aligned}$$

We also check that the verifier will compute the same $\vec{w_1}$ as the signer. Using Lemma 6.2.2 and that $\left\|LowBits_q(\vec{w} - c\vec{s_2}, 2\gamma)\right\|_\infty < \gamma - \beta$, we have:

$$HighBits_q(\vec{w} - c\vec{s_2}, 2\gamma) = HighBits_q(\vec{w}, 2\gamma) = \vec{w_1}.$$

It becomes apparent that we have correctness.

No-Abort Honest Verifier Zero-Knowledge

In Figure 6.5 we see the honestly generated transcript produced by a prover using the sk and a simulated transcript produced by a simulator knowing no more than the pk.

Lemma 6.2.3. *The ID-scheme ID is perfectly naHVZK, if* $\beta \geq \max_{\vec{s}\in S_\eta, c\in ChSet}\|c\vec{s}\|_\infty$.

Trans(sk):

1 : $\mathbf{A} \in R_q^{k \times l} := Sam(\rho)$
2 : $\vec{y} \leftarrow S^l_{\gamma'-1}$
3 : $\vec{w} := \mathbf{A}\vec{y}$
4 : $\vec{w_1} := HighBits_q(\vec{w}, 2\gamma)$
5 : $c \leftarrow ChSet$
6 : $\vec{z} := \vec{y} + c\vec{s_1}$
7 : **if** $\|\vec{z}\|_\infty \geq \gamma' - \beta$
8 : **then return** $\perp$
9 : **if** $\|LowBits_q(\vec{w} - c\vec{s_2}, 2\gamma)\|_\infty \geq \gamma - \beta$
10 : **then return** $\perp$
11 : $\vec{h} := MakeHint_q(-c\vec{t_0}, \vec{w} - c\vec{s_2} + c\vec{t_0}, 2\gamma)$
12 : **return** $(c, (\vec{z}, \vec{h}))$

SimTrans(pk):

1 : $\mathbf{A} \in R_q^{k \times l} := Sam(\rho)$
2 : with probability $1 - \frac{|S^l_{\gamma'-\beta-1}|}{|S^l_{\gamma'-1}|}$ **return** $\perp$
3 : $\vec{z} \leftarrow S^l_{\gamma'-1}$
4 : $c \leftarrow ChSet$
5 : **if** $\|LowBits_q(\mathbf{A}\vec{z} - c\vec{t} + c\vec{t_0}, 2\gamma)\|_\infty \geq \gamma - \beta$
6 : **then return** $\perp$
7 : $\vec{h} := MakeHint_q(-c\vec{t_0}, \mathbf{A}\vec{z} - c\vec{t}, 2\gamma)$
8 : **return** $(c, (\vec{z}, \vec{h}))$

Figure 6.5: Algorithm *Trans*(pk) for transcripts honestly generated by the prover and *SimTrans*(pk) for simulated transcripts.

Proof. Wanting to show that the output distributions of *Trans*(sk) and *SimTrans*(pk) in Figure 6.5, are identical, we let $(\vec{s_1}, \vec{s_2}) \in S^l_\eta \times S^k_\eta$ be any polynomials s.t. $\mathbf{A}\vec{s_1} + \vec{s_2} = \vec{t}$.

We want to compute the probability that some particular $\vec{z} \in S^k_{\gamma'-\beta-1}$ is computed in line 6 of the *Trans*(sk) procedure. For any $c \in ChSet$, we then get:

$$\Pr_{\vec{y} \leftarrow S^l_{\gamma'-1}}[\vec{y} + c\vec{s_1} = \vec{z}] \;=\; \Pr_{\vec{y} \leftarrow S^l_{\gamma'-1}}[\vec{y} = \vec{z} - c\vec{s_1}] \;=\; \frac{1}{|S^l_{\gamma'-1}|}.$$

We get this from noticing that $\|c\vec{s_1}\|_\infty \leq \beta$, and thus $\vec{z} - c\vec{s_1} \in S^l_{\gamma'-1}$. We see that every $\vec{z} \in S^l_{\gamma'-\beta-1}$ is equally likely to be generated. Further, the probability of producing a $\vec{z} \in S^l_{\gamma'-\beta-1}$, and thus not outputting $\perp$ becomes exactly $\frac{|S^l_{\gamma'-\beta-1}|}{|S^l_{\gamma'-1}|}$. We see that the output distribution of $(c, \vec{z})$ is uniform in $ChSet \times S^l_{\gamma'-\beta-1}$ or *Trans*(sk) outputs $\perp$ with probability $1 - \frac{|S^l_{\gamma'-\beta-1}|}{|S^l_{\gamma'-1}|}$ in line 7. We compare *Trans*(sk) and *SimTrans*(pk) and see that they have the exact same distribution. Further to see that the rejection sampling in line 9 of *Trans*(sk) and line 5 of *SimTrans*(sk) is equal we note that:

$$\vec{w} - c\vec{s_2} = \mathbf{A}\vec{y} - c\vec{s_2} = \mathbf{A}(\vec{z} - c\vec{s_1}) - c\vec{s_2} = \mathbf{A}\vec{z} - \mathbf{A}c\vec{s_1} - c\vec{s_2} = \mathbf{A}\vec{z} - c\vec{t}.$$

Which completes the proof. □

Lossyness

Wanting to show lossy soundness, we consider the ID-scheme running for a lossy public key $pk_{ls} \leftarrow LossyIGen$. We wish to see that even an unbounded prover using this lossy key only has approximately $\frac{1}{|ChSet|}$ probability of producing a transcript the verifier accepts, in each run of the ID-scheme. This will give us an upper-bound for (5.2) considering an unbounded adversary against the $LOSSY-IMP$ game.

LossyIGen:

1 : $\rho \leftarrow \{0,1\}^{256}, \mathbf{A} \leftarrow R_q^{k\times l} := Sam(\rho)$
2 : $\vec{t} \leftarrow R_q^k$
3 : $\vec{t_1} := Power2Round_q(\vec{t}, d)$
4 : $\vec{t_0} := \vec{t} - \vec{t_1} \cdot 2^d$
5 : **return** $pk := (\rho, \vec{t_1}, \vec{t_0})$

Figure 6.6: The lossy key-generator algorithm for Dilithium.

Looking at the *LossyIGen* algorithm for Dilithium in Figure 6.6 and comparing it to the *IGen* algorithm for Dilithium in Figure 6.3, we see that:

$$(\mathbf{A}, \vec{t}) \leftarrow LossyIGen, \text{ where } (\mathbf{A}, \vec{t}) \leftarrow R_q^{k\times l} \times R_q^k$$
$$(\mathbf{A}, \mathbf{A}\vec{s_1} + \vec{s_2}) \leftarrow IGen, \text{ where } \mathbf{A} \leftarrow R_q^{k\times l} \text{ and } (\vec{s_1}, \vec{s_2}) \leftarrow S_\eta^l \times S_\eta^k$$

It is apparent that:

$$Adv_{ID}^{LOSS}(\mathcal{A}) = Adv_{k,l,D}^{MLWE}(\mathcal{A}),$$

for the uniform distribution D over S_η.

In order to show Dilithium to be lossy sound we need Lemma 6.2.4, showing that if $(\mathbf{A}, \vec{t})$ is drawn at random then a particular linear equation is unlikely to have any solutions. The proof can at times seem rather far fetched, remembering that it is mostly about counting the possible outcomes will help us to see the intuition.

Lemma 6.2.4. *Let $\alpha_1, \alpha_2 \leq \sqrt{q/2}$ be positive integers, let D be a set of elements in $R\backslash\{0\}$ with coefficients less than $\sqrt{q/2}$ and let d be such that $2^d < 2\alpha_1$, then:*

$$\Pr_{\boldsymbol{A}\leftarrow R_q^{k\times l}, \vec{t}\leftarrow R_q^k}[\exists\, (\vec{z_1}, \vec{z_2}, c) \in S_{\alpha_1}^l \times S_{\alpha_2}^k \times D \ s.t\ \boldsymbol{A}\vec{z_1} + \vec{z_2} = c\vec{t_1} \cdot 2^d]$$
$$\leq 2 \cdot |D| \left(\frac{(2\alpha_1 + 1)^l \cdot (2\alpha_2 + 1)^k}{q^k} \right)^n, \tag{6.3}$$

where $\vec{t_1} := Power2Round_q(\vec{t}, d)$.

Proof. There are two cases to consider when we prove this Lemma, namely $\vec{z_1} = 0$ and $\vec{z_1} \neq 0$.

Case 1: ($\vec{z_1} = 0$) We rewrite (6.3) for this case and get:

$$\Pr_{\vec{t} \leftarrow R_q^k}[\exists\, (\vec{z_2}, c) \in S_{\alpha_2}^k \times D \text{ s.t } \vec{z_2} = c\vec{t_1} \cdot 2^d].$$

As $0 < \|c\|_\infty < \sqrt{q/2}$ and $q = 5 \bmod 8$, we know that c is invertible in R_q . This allows us to consider the above probability as:

$$\begin{aligned}
&\Pr_{\vec{t} \leftarrow R_q^k}[\exists\, (\vec{z_2}, c) \in S_{\alpha_2}^k \times D \text{ s.t } \vec{z_2} \cdot (2^d c)^{-1} = \vec{t_1}] \\
&\quad \leq \sum_{\vec{z_2} \in S_{\alpha_2}^k, c \in D} \Pr_{\vec{t} \leftarrow R_q^k}[\vec{t_1} = \vec{z_2} \cdot (2^d c)^{-1}] \\
&\quad \leq \sum_{\vec{z_2} \in S_{\alpha_2}^k, c \in D} \left(\frac{2^d}{q}\right)^{nk} = \left(\frac{(2\alpha_2 + 1) \cdot 2^d}{q}\right)^{nk} \cdot |D|. \qquad (6.4)
\end{aligned}$$

We obtain the last line of (6.4) by noticing that the most frequent value of each coefficient of $\vec{t_1}$ occurs no more than 2^d times, for $\vec{t} \in R_q^k$.

Case 2: ($\vec{z_1} \neq 0$) We assume, without loss of generality, that the first polynomial in $\vec{z_1}$ is non-zero. For the tuple $(\vec{z_1} \neq 0, \vec{z_2}, c)$ we then get the probability:

$$\Pr_{\mathbf{A} \leftarrow R_q^{k \times l}, \vec{t} \leftarrow R_q^k}[\mathbf{A}\vec{z_1} + \vec{z_2} = c\vec{t_1} \cdot 2^d].$$

By defining $\vec{z_1} := \begin{bmatrix} z \\ \vec{z_1'} \end{bmatrix}$, we are able to see that the above probability equals:

$$\Pr_{\vec{a} \leftarrow R_q^{k \times l}, \mathbf{A'} \leftarrow R_q^{(k-1) \times l}, \vec{t} \leftarrow R_q^k}[\vec{a}z + \mathbf{A'}\vec{z_1'} + \vec{z_2} = c\vec{t_1} \cdot 2^d] = \Pr_{\vec{a} \leftarrow R_q^k}[\vec{a}z = -\mathbf{A'}\vec{z_1'} - \vec{z_2} + c\vec{t_1} \cdot 2^d].$$

Using the fact that $\|\vec{z}\|_\infty < \sqrt{q/2}$ and $q = 5 \bmod 8$, we see that $\vec{z}$ has an inverse in R_q. The above probability becomes:

$$\Pr_{\vec{a} \leftarrow R_q^k}[\vec{a} = z^{-1} \cdot (\mathbf{A'}\vec{z_1'} - \vec{z_2} + c\vec{t_1} \cdot 2^d)] = \left(\frac{1}{q}\right)^{nk}.$$

Thus, for $\vec{z} \neq 0$, we can upper bound (6.3) using the union bound:

$$\sum_{\vec{z_1} \in S_{\alpha_1}^l \setminus \{0\}, \vec{z_2} \in S_{\alpha_2}^k, c \in D} \left(\frac{1}{q}\right)^{nk} < \left(\frac{(2\alpha_1 + 1)^l \cdot (2\alpha_2 + 1)^k}{q^k}\right)^n \cdot |D|. \qquad (6.5)$$

We recall our assumption $2^d < 2\alpha$ and add the probabilities for *Case 1* and *Case 2*, namely (6.4) and (6.5), and get (6.3). We see that the Lemma holds. □

With Lemma 6.2.4 we move on to prove Lemma 6.2.5. Here we want to show that if an adversary $\mathcal{C}$ outputting $(\vec{w}_1, St)$ is able to create a valid response to more than one challenge c, then the linear equation Lemma 6.2.4 refers to has a solution. We conclude that all $\mathbf{A}, \vec{t}$ produced by *LossyIGen* only has one challenge the prover can respond to, giving the prover a sucess probability of $\frac{1}{|ChSet|}$.

Lemma 6.2.5. *If $4\gamma + 2, 2\gamma' < \sqrt{q/2}$ and $\gamma' < \gamma\beta$, and $l \leq k$, then ID has ϵ_{ls}-lossy soundness such that:*

$$\epsilon_{ls} \leq \frac{1}{|ChSet|} + 2 \cdot |ChSet|^2 \cdot \left(\frac{32\gamma\gamma'}{q}\right)^{nk}.$$

Proof. We consider an unbound adversary $\mathcal{C}$ playing the $LOSSY-IMP$ game in Figure 5.5, instantiated with the *LossyIGen* algorithm in Figure 6.6 and the verifier algorithm in Figure 6.3. Suppose, for the commitment $\vec{w}_1$, there exists two tuples $(\vec{z}, \vec{h})$ and $(\vec{z}', \vec{h}')$ for which $\mathcal{C}$ wins. That is, both $\|\vec{z}\|_\infty < \gamma' - \beta$ and $\|\vec{z}'\|_\infty < \gamma' - \beta$, and:

$$\vec{w}_1 = UseHint_q(\vec{h}, \mathbf{A}\vec{z} - \vec{t_1}c \cdot 2^d, 2\gamma) = UseHint_q(\vec{h}', \mathbf{A}\vec{z}' - \vec{t_1}c' \cdot 2^d, 2\gamma).$$

Using Lemma 6.2.1 it is clear to see that:

$$\left\|\mathbf{A}\vec{z} - \vec{t_1}c \cdot 2^d - \vec{w}_1 \cdot 2\gamma\right\|_\infty \leq 2\gamma + 1,$$
$$\left\|\mathbf{A}\vec{z}' - \vec{t_1}c' \cdot 2^d - \vec{w}_1 \cdot 2\gamma\right\|_\infty \leq 2\gamma + 1,$$

where using the triangular inequality, further give us:

$$\left\|\mathbf{A}(\vec{z} - \vec{z}') - \vec{t_1}(c - c') \cdot 2^d\right\|_\infty \leq 4\gamma + 2.$$

For $\|\vec{z} - \vec{z}'\|_\infty \leq 2(\gamma' - \beta - 1)$ and some $\vec{u}$ where $\|\vec{u}\|_\infty \leq 4\gamma + 2$ we rewrite this as:

$$\mathbf{A}(\vec{z} - \vec{z}') + \vec{u} = \vec{t_1} \cdot 2^d \cdot (c - c'). \tag{6.6}$$

Applying Lemma 6.2.4, for $\mathbf{A} \leftarrow R_q^{k \times l}$ and $\vec{t} \leftarrow R_q^k$, we see that (6.6) is satisfied with probability less than:

$$2 \cdot |ChSet|^2 \cdot \frac{(4(\gamma' - \beta))^{nl} \cdot (8\gamma + 5)^{nk}}{q^{nk}} < 2 \cdot |ChSet|^2 \cdot \left(\frac{32\gamma\gamma'}{q}\right)^{nk}.$$

We see the claim of the Lemma holds. □

Min-Entropy

To prove the min-entropy of our scheme, we will show that the $\vec{w}_1$ sent by an honest prover is extremely likely to be distinct each time we run the protocol. While proving this lemma we will use the same technique as we did for Lemma 6.2.4.

Lemma 6.2.6. *If $2\gamma, 2\gamma' < \sqrt{q/2}$ and $l \leq k$, the Dilithium ID-scheme in Figure 6.3 has α bits of min-entropy, where:*

$$\alpha > nl \cdot \log\left(\min\left\{\frac{q}{(4\gamma+1)(4\gamma'+1)}, 2\gamma'-1\right\}\right).$$

Proof. We make the claim that:

$$\begin{aligned}\Pr_{\mathbf{A}\leftarrow R_q^{k\times l}}[\exists\, \vec{z} \neq \vec{z}' \in S^l_{\gamma'-1} \text{ s.t } \mathit{HighBits}_q(\mathbf{A}\vec{y}, 2\gamma) = \mathit{HighBits}_q(\mathbf{A}\vec{y}', 2\gamma)] \\ < \left(\frac{(4\gamma+1)(4\gamma 1+1)}{q}\right)^{nk}. \end{aligned} \tag{6.7}$$

We see the probability of a collision for $\vec{z}$. If there are no collisions we know that every $\vec{w_1} = \mathit{HighBits}_q(\mathbf{A}\vec{y}', 2\gamma)$ will be drawn uniformly. In other words, with probability $1-\left(\frac{(4\gamma+1)(4\gamma 1+1)}{q}\right)^{nk}$ over $\mathbf{A} \leftarrow R_q^{k\times l}$ every $\vec{w_1} = \mathit{HighBits}_q(\mathbf{A}\vec{y}', 2\gamma)$ has a $\frac{1}{|\gamma'-1|} = (2\gamma'-1)^{-nl}$ chance of being output.

In order to see why we want to make this claim, we rewrite the bound for α from the Lemma:

$$\begin{aligned}\alpha &> nl \cdot \log\left(\min\left\{\left(\frac{q}{(4\gamma+1)(4\gamma'+1)}\right), (2\gamma'-1)\right\}\right) \\ &= \log\left(\min\left\{\left(\frac{q}{(4\gamma+1)(4\gamma'+1)}\right)^{nl}, (2\gamma'-1)^{nl}\right\}\right) \\ &= \log\left(\max\left\{\left(\frac{(4\gamma+1)(4\gamma'+1)}{q}\right)^{-nl}, (2\gamma'-1)^{nl}\right\}\right) \\ &= -\log\left(\max\left\{\left(\frac{(4\gamma+1)(4\gamma'+1)}{q}\right)^{nl}, (2\gamma'-1)^{nl}\right\}\right),\end{aligned}$$

and recall Definition 2.3.1. It becomes apparent that with the assumption that $k \geq l$, the claim of the Lemma follows and it only remains for us to prove (6.7). We define the vector pairs:

$$(\vec{w}_1, \vec{w}_0) = \mathit{Decompose}_q(\mathbf{A}\vec{y}, 2\gamma) \text{ and } (\vec{w}'_1, \vec{w}'_0) = \mathit{Decompose}_q(\mathbf{A}\vec{y}', 2\gamma),$$

and then we see that:

$$\mathit{HighBits}_q(\mathbf{A}\vec{y}, 2\gamma) = \mathit{HighBits}_q(\mathbf{A}\vec{y}', 2\gamma) \implies \vec{w}_1 = (\mathbf{A}\vec{y} - \vec{w}_0)/2\gamma \text{ and } \vec{w}'_1 = (\mathbf{A}\vec{y}' - \vec{w}'_0)/2\gamma$$

where $\vec{w}_1 = \vec{w}'_1$ and $\|\vec{w}_0\|_\infty, \|\vec{w}'_0\|_\infty \leq \gamma$. Because of this, we get:

$$\begin{aligned}(\mathbf{A}\vec{y} - \vec{w}_0)/2\gamma = (\mathbf{A}\vec{y}' - \vec{w}'_0)/2\gamma &\implies \mathbf{A}\vec{y} - \vec{w}_0 = \mathbf{A}\vec{y}' - \vec{w}'_0 \\ &\implies \mathbf{A}(\vec{y} - \vec{y}') - (\vec{w}_0 - \vec{w}'_0) = 0 \end{aligned} \tag{6.8}$$

where $\|\vec{y} - \vec{y}\|_\infty \leq 2\gamma'$ and $\|\vec{w}_0 - \vec{w}'_0\|_\infty \leq 2\gamma$.

Using the same argument as in *Case 2* of Lemma 6.2.4, as $2\gamma, 2\gamma' < \sqrt{q/2}$, over the choice of $\mathbf{A} \leftarrow R_q^{k \times l}$ the probability that there exists two non-zero elements, here $(\vec{y} - \vec{y})$ and $(\vec{w}_0 - \vec{w}_0')$ with norm less than 2γ and $2\gamma'$, respectively, satisfying (6.8) is at most:

$$\left(\frac{(4\gamma+1)^l(4\gamma'+1)^k}{q^k}\right)^n \leq \left(\frac{(4\gamma+1)(4\gamma 1+1)}{q}\right)^{nk}.$$

Which proves (6.7) and thus the Lemma. □

Computational Unique Response

In order to show the strong unforgeability of our scheme, we wish to prove that the Dilithium ID-scheme has *CUR*. To do so, we need to prove two lemmas.

Lemma 6.2.7. *If transcripts* $(\vec{w}_1, c, (\vec{z}, \vec{h}))$ *and* $(\vec{w}_1, c, (\vec{z'}, \vec{h'}))$ *are such that* $V(pk, \vec{w}_1, c, (\vec{z}, \vec{h})) = V(pk, \vec{w}_1, c, (\vec{z}, \vec{h})) = 1$*, where* $(\vec{z}, \vec{h}) \neq (\vec{z'}, \vec{h'})$*, then there exists vectors* $\vec{v}, \vec{u}$ *such that* $\|\vec{v}\|_\infty < 2(\gamma' - \beta)$*,* $\|\vec{u}\|_\infty < 4\gamma + 2$ *and* $\boldsymbol{A}\vec{v} + \vec{u} = 0$*.*

Proof. Following the conditions of the Lemma we see that they imply:

$$\vec{w}_1 = \mathit{UseHint}_q(\vec{h}, \mathbf{A}\vec{z} - c\vec{t_1} \cdot 2^d, 2\gamma) = \mathit{UseHint}_q(\vec{h'}, \mathbf{A}\vec{z'} - c\vec{t_1} \cdot 2^d, 2\gamma).$$

By *iii)* of Lemma 6.2.1 we notice that we must have $\vec{z} \neq \vec{z'}$, as $\vec{z} = \vec{z'}$ would imply $\vec{h} = \vec{h'}$ and thus $Z = (\vec{z}, \vec{h}) = (\vec{z'}, \vec{h'}) = Z'$. Now using *ii)* of the same lemma, we see that the above equation means:

$$\left\|\mathbf{A}\vec{z} - c\vec{t_1} \cdot 2^d - \vec{w}_1 \cdot 2\gamma\right\|_\infty \leq 2\gamma + 1,$$
$$\left\|\mathbf{A}\vec{z'} - c'\vec{t_1} \cdot 2^d - \vec{w}_1 \cdot 2\gamma\right\|_\infty \leq 2\gamma + 1.$$

Where the triangle inequality allows us to rewrite this as:

$$\mathbf{A}(\vec{z} - \vec{z'}) + (c - c')\vec{t} \cdot 2^d = 0,$$

Now, we see that for $\vec{v}$ and $\vec{u}$, where $\|\vec{v}\| < 2(\gamma' - \beta)$ and $\|\vec{u}\|_\infty < 4\gamma + 2$, we can write the above equation as $\mathbf{A}\vec{v} + \vec{u} = 0$ and thus prove the Lemma. □

With the results of Lemma 6.2.7, we have the results we need to prove the advantage of an adversary against the *CUR* of the scheme.

Lemma 6.2.8. *If* $4\gamma+2, 2\gamma' < \sqrt{q/2}$*,* $\gamma' < \gamma\beta$*, and* $l \leq k$*, then for every (unbounded) adversary* $\mathcal{A}$ *against CUR:*

$$\mathit{Adv}(\mathcal{A})_{ID}^{CUR} < \left(\frac{32\gamma\gamma'}{q}\right)^{nk}.$$

Proof. Let $(W, c, Z) = (\vec{w}_1, c, (\vec{z}, \vec{h}))$ be any valid transcript. If an adversary $\mathcal{A}$ is able to generate a response $Z' = (\vec{z'}, \vec{h'}) \neq Z$ such that $V(pk, \vec{w}_1, c, (\vec{z'}, \vec{h'})) = 1$ for $pk = (\mathbf{A}, \vec{t}_1)$, then by Lemma 6.2.7 there exists vectors $\vec{v}, \vec{u}$ such that $\mathbf{A}\vec{v} + \vec{u} = 0$ where $\|\vec{v}\|_\infty < 2(\gamma' - \beta)$ and $\|\vec{u}\|_\infty < 4\gamma + 2$.

Using the same argument as in *Case 2* of Lemma 6.2.4, as $4\gamma + 2, 2\gamma' < \sqrt{q/2}$, over the choice of $\mathbf{A} \leftarrow R_q^{k \times l}$ the probability that there exists two non-zero elements, $\vec{v}$ and $\vec{u}$, with norm less than $2(\gamma' - \beta)$ and $4\gamma + 2$, respectively, such that $\mathbf{A}\vec{v} + \vec{u} = 0$ is:

$$\frac{(4(\gamma' - \beta))^{nl} \cdot (8\gamma + 5)^{nk}}{q^{nk}} < 2 \cdot |ChSet|^2 \cdot \left(\frac{32\gamma\gamma'}{q}\right)^{nk}.$$

□

6.2.4 Signature Scheme

The Dilithium signature scheme we work with here differs slightly from the signature presented by Ducas et al. [27]. We present the $DFS[ID, H, PRF]$ derived Dilithium signature by Kiltz et al. [7] based on the lossy ID-scheme we presented in Section 6.2.1. We refer to the signature as Dilithium and will point out the difference between the Ducas et al. and Kiltz et al signature when distinguishing between the two is necessary.

The signature scheme is presented in Figure 6.7. In lines 5 we see the random key K as the signature is derived through the deterministic Fiat-Shamir with aborts transformation. We also note that the bound on κ in line 3 only is present to maintain consistency from the generic signing algorithms presented in Figure 5.10.

6.2.5 Security Proof

We now want to apply the theorems and results from Chapter 5 to the Dilithium signature we just introduced. We wish to show the signature to be unforgeable in the QROM. To do so we recall Theorem 5.2.2 and its proof. Applying this to Dilithium we get:

$$Adv^{UF-CMA_1}_{Dilithium_{SIG}}(\mathcal{A}) \leq Adv^{LOSS}_{Dilithium_{ID}}(\mathcal{B}) + 8(Q_H + 1)^2 \cdot \epsilon_{ls} + \kappa_m Q_S \cdot \epsilon_{zk} + 2^{-\alpha+1}.$$

In order to simplify the above equation we look at the properties we explored in Section 6.2.3 and instantiate these for the recommended security level in Table 7.1.

In Section 6.2.3 of this chapter, we find Lemma 6.2.3 and that the Dilithium ID-scheme is perfectly naHVZK. This means $\epsilon_{zk} = 0$, which allows us to remove the $\kappa_m Q_S \cdot \epsilon_{zk}$ term. Later in the same section, we see that the advantage of the $\mathcal{B}$ against $LOSS$ can be substituted with the advantage against $MLWE$. Further, for the lossy soundness we look back at Lemma 6.2.5 and see that $\epsilon_{ls} \leq \frac{1}{|ChSet|} + 2 \cdot |ChSet|^2 \cdot \left(\frac{32\gamma\gamma'}{q}\right)^{nk}$. The recommended value for the ring dimension is $n = 512$, which means the size of $ChSet$ is greater than 2^{265} allowing us to write

$Sign((sk, K), M)$

$\kappa := 0$
$\mathbf{A} \leftarrow R_q^{k \times l} := Sam(\rho)$
while $(\vec{z}, \vec{h}) = \perp$ **and** $\kappa \leq \frac{2000}{1-\delta}$ **do**
  $\kappa := \kappa + 1$
  $\vec{y} \leftarrow \mathbf{S}^l_{\gamma'-1} := Sam(K \parallel M \parallel \kappa)$
  $\vec{w} := \mathbf{A}\vec{y}$
  $\vec{w_1} := HighBits_q(\vec{w}, 2\gamma)$
  $c := H(\vec{w_1} \parallel M)$
  $\vec{z} := \vec{y} + c\vec{s_1}$
  if $\|\vec{z}\|_\infty \geq \gamma' - \beta$ **or** $\|LowBits_q(\vec{w} - c\vec{s_2}, 2\gamma)\|_\infty \geq \gamma - \beta$
    then $(\vec{z}, \vec{h}) := \perp$
  else $\vec{h} := MakeHint_q(-c\vec{t_0}, \vec{w} - c\vec{s_2} + c\vec{t_0}, 2\gamma_2)$
return $\sigma = (\vec{z}, \vec{h}, c)$

$Ver(pk, M, \sigma)$

$\mathbf{A} \leftarrow R_q^{k \times l} := Sam(\rho)$
$\vec{w_1'} := UseHint_q(\vec{h}, \mathbf{A}\vec{z} - c\vec{t_1} \cdot 2^d, \gamma_2)$
return $[\![\|\vec{z}\|_\infty < \gamma' - \beta]\!]$ **and** $[\![c = H(\vec{w_1'} \parallel M)]\!]$

Figure 6.7: The Dilithium signature scheme. The key-generation algorithm is the same as in Figure 6.3, where the secret key now also contains a random key K for the pseudorandom function *Sam*.

$\epsilon_{ls} \leq 2^{-265} + 2^{-334} \leq 2^{-264}$. Lastly, for the α bits of minimal entropy, using Lemma 6.2.6 for the recommended instance in Table 7.1, we will get $\alpha > 2900$ allowing us to ignore the $2^{-\alpha}$ term. The above equation can now be written as:

$$Adv^{UF-CMA_1}_{Dilithium_{SIG}}(\mathcal{A}) \leq Adv^{MLWE}_{k,l,D}(\mathcal{B}) + Q_H^2 \cdot 2^{-261},$$

where the advantage function for $\mathcal{B}$ against the decision-$MLWE$ problem is:

$$\begin{aligned} Adv(\mathcal{B})^{MLWE}_{m,k,D} :=& |\Pr[\mathcal{B}(\mathbf{A}, \vec{t}) \Rightarrow 1 | \mathbf{A} \leftarrow R_q^{m \times k}, \vec{t} \leftarrow R_q^m] \\ &- \Pr[\mathcal{A}(\mathbf{A}\vec{s_1}, \vec{s_2}) \Rightarrow 1 | \mathbf{A} \leftarrow R_q^{m \times k}, \vec{s_1} \leftarrow D^k \vec{s_2} \leftarrow D^m]|. \end{aligned}$$

In other words the UF-CMA$_1$ security of Dilithium comes down to the hardness of the $MLWE$ problem.

Further on Security

Just like we did for the general proof of UF-CMA$_1$ security of $DFS[LID, H, PRF, \kappa_m]$ signatures in Chapter 5, we can extend the UF-CMA$_1$ security for Dilithium to UF-CMA security. We use the fact that our Dilithium signature is derived through the deterministic Fiat-Shamir with aborts transformation and get:

$$Adv^{UF-CMA}_{Dilithium_{SIG}}(\mathcal{A}) \leq Adv^{MLWE}_{k,l,D}(\mathcal{B}) + Q_H^2 \cdot 2^{-261} + Adv^{PR}_{Sam}(\mathcal{C}).$$

Recalling Lemma 6.2.8 from Section 6.2.3, we know that Dilithium satisfies the CUR property. With this, we extend the UF-CMA security of the signature to sUF-CMA security and get:

$$Adv^{UF-CMA}_{Dilithium_{SIG}}(\mathcal{A}) \leq Adv^{MLWE}_{k,l,D}(\mathcal{B}) + Q_H^2 \cdot 2^{-261} + Adv^{PR}_{Sam}(\mathcal{C}) + Adv^{CUR}_{Dilithium_{ID}}(\mathcal{D}).$$

We see that Dilithium is strongly unforgeable in the QROM and we therefore consider it secure against a quantum attacker.

Chapter 7

Discussion

In this thesis we have studied what provable security means in PQC. We have looked at two signature schemes that are both considered quantum safe, but on different assumptions. We will compare the two schemes and the different approaches to quantum security.

In 2009 Lyubashevsky introduced the Fiat-Shamir with aborts technique, as he released the Lyubashevsky signature we have looked at. The signature scheme extended the original Fiat-Shamir transformation to allow the prover to abort. Lyubashevsky gave this technique of turning an interactive ID-schemes into a non-interactive signature scheme, a broader range of application.

The Lyubashevsky signature is proven unforgeable in the classical ROM using rewinding. As we know, rewinding is problematic against an adversary with quantum capabilities and thus the proof is not considered safe against a quantum attacker. However, the underlying problem, namely the SVP_γ, is considered hard for even an adversary with quantum capabilities — Lyubashevsky is built on a quantum safe problem. The signature has become a building block for PQC signatures to come, but is itself not the most efficient signature.

Dilithium is a signature that NIST have recommended for use in PQC. It is considered relatively simple to implement, it has strong theoretical security and high efficiency. It can be applied to a broad range of cryptographic applications and is NIST's primary choice for signatures in PQC. The efficiency of the scheme is one of the most desirable aspects of Dilithium. The "key" to this is the small size of the public key. The matrix $\mathbf{A}$ is not included in the key, rather retrieved using the XOF *Sam* on the seed ρ. The public key then consists of a seed ρ as well as the vector $\vec{t_1} := \mathit{Power2Round}_q(\vec{t}, d)$. A hint $\vec{h}$ is provided so that the verifier can still check the signature without knowing the entire $\vec{t}$. Along with $\mathit{Power2Round}_q$ Dilithium has five other supporting algorithms, which can be seen in Figure 6.4. These algorithms all work to ensure the efficinecy of the signature, while still preserving its security.

Dilithium is proven unforgeable in the QROM, which means it is proven safe against an adversary with quantum capabilities. Important to notice is that we are referring to Kiltz et al.'s [7]

Dilithium signature that is derived from a lossy ID-scheme. The original Dilithium signature by Ducas et al. [27] does not have a QROM proof and is more efficient than the lossy derived Dilithium. We point out that Kiltz et al. [7] do explore a reduction of a deterministic variant of the original Dilihtium signature that is not dependant on the underlying ID-scheme being lossy. This reduction is based on a combination of the MLWE and a special instantiation of MSIS called *SelfTargetMSIS*. We get a QROM proof reduced to MLWE and SelfTargetMSIS, but as SelfTargetMSIS is derived from MSIS using the forking lemma, we are left with the same argument as we have for Lyubashevsky. We have an underlying problem that is quantum safe, but we are not able to fully model a quantum attacker. We note that SelfTargetMSIS is the underlying lattice problem for the original Dilithium signature.

In Table 7.1 we see a comparison of Lyubashevsky, the original Dilithium and the lossy derived Dilithium, here denoted as Dilithium-QROM. More exactly we see the size difference of the public keys and signatures, measured in bytes, for two security levels. It is apparent that Dilithium is a far more efficient signature than Lyubashevsky. We also witness the increased size of the Dilithium-QROM signature to that of the original Dilithium — making it lossy increases the combined key and signature size with a factor of a little more than 3. It is interesting to compare all of these schemes to modern day standards like the RSA. The public key and the signature of RSA are both an impressive 256 bytes. Clearly, as far as efficiency goes this is much preferred over the quantum safe protocols we have looked at, when classical security is sufficient.

As both Lyubashevsky and Dilithium are considered to be quantum safe, but only Dilithium has a security proof in the QROM, it becomes natural to ask — do we need the QROM? The question is simple; the answer is less obvious.

Looking at the classical ROM we know that the model has been considered sufficient, even though it does not actually reflect the real world. However, the test of time has spoken in favour of the model. This is a good talking point for why we should continue using this model, but it does not take into consideration the capabilities of new quantum attackers, nor should it necessarily give us confidence that it will continue to be sufficient in the face of new adversarial powers. The cryptographic community has not agreed on the need of a new model. Yet, the fact that the QROM has been developed as a response to the quantum development suggests that it is worth considering. Having a model capable of testing a scheme towards its possible attackers is a useful tool giving us greater confidence in the security of quantum safe protocols. The fact that schemes like Lyubashevsky seems to be quantum secure with only a ROM proof is not necessarily a valid argument against the QROM, rather it is a good argument for the ROM. Looking at the size difference of the original Dilithium and the lossy derived Dilithium, it is easy to wish that the ROM will stand the test of time also in a quantum setting.

Signature	Parameter	Definition	Sample Instantiation	
			Recommended	Very High
Lyubashevsky [28]	n	integer that is power of 2	512	1024
	m	any integer	8	8
	σ	any integer	2047	2047
	κ	integer that is power of 2	512	1024
	p	integer $\approx$	$2^{95.8}$	$2^{95.9}$
	R_q	$\mathbb{Z}_q[X]/\langle X^n+1\rangle$		
	D	$\{g \in R \mid \|g\|_\infty \leq mn\sigma\kappa\}$		
	D_s	$\{g \in R \mid \|g\|_\infty \leq \sigma\}$		
	D_c	$\{g \in R \mid \|g\|_1 \leq \kappa\}$		
	D_y	$\{g \in R \mid \|g\|_\infty \leq mn\sigma\kappa\}$		
	G	$\{g \in R \mid \|g\|_\infty \leq mn\sigma\kappa - \sigma\kappa\}$		
	pk size in bytes		49000	9800
	signature size in bytes		119000	246000
Dilithium-QROM [7]	q	ring modulus	$2^{45} - 21283$	$2^{45} - 21283$
	n	ring dimension	512	512
	(k, l)	dimensions of matrix $\mathbf{A}$	$(4, 4)$	$(5, 5)$
	d	dropped bits from $\vec{t}$	15	15
		# of ± 1 in $c \in ChSet$	46	46
	γ	s.t. $\gamma \mid q-1$	905679	905679
	γ'	$\approx$ max. sig. coefficient	905679	905679
	η	max. coefficient of $\vec{s_1}, \vec{s_2}$	7	3
	β	$\eta \cdot$ (#of ± 1's in c)	322	138
	pk size in bytes		7712	9632
	signature size in bytes		5690	70987
Dilithium [27]	q	ring modulus	$2^{23} - 8191$	$2^{23} - 8191$
	n	ring dimension	256	256
	(k, l)	dimensions of matrix $\mathbf{A}$	$(5, 4)$	$(6, 5)$
	d	dropped bits from $\vec{t}$	14	14
		# of ± 1 in $c \in ChSet$	60	60
	γ	s.t. $\gamma \mid q-1$	261888	261888
	γ'	$\approx$ max. sig. coefficient	523776	523776
	η	max. coefficient of $\vec{s_1}, \vec{s_2}$	5	3
	β	$\eta \cdot$ (#of ± 1's in c)	275	175
	pk size in bytes		1472	1760
	signature size in bytes		2701	3366

Table 7.1: Description of the parameters for Lyubashevsky signature, the original Dilithium signature and the lossy derived Dilithium signature, here denoted as Dilithium-QROM. In the last two columns we see the parameters instantiated for two different security levels — We are able to see the size difference of the three signatures.

Bibliography

[1] P. W. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, 1994. DOI: 10.1109/SFCS.1994.365700.

[2] C. Gidney and M. Ekerå, *How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits*, Apr. 2021. DOI: 10.22331/q-2021-04-15-433.

[3] J. Chow, O. Dial and J. Gambetta, *Ibm quantum breaks the 100-qubit processor barrier*, IBM Research Blog, 2, https://research.ibm.com/blog/127-qubit-quantum-processor-eagle, 2021.

[4] J. Gambetta, *Quantum-centric supercomputing: The next wave of computing*, IBM Research Blog, 1, https://research.ibm.com/blog/next-wave-quantum-centric-supercomputing, 2022.

[5] A. Dash, D. Sarmah, B. K. Behera and P. K. Panigrahi, *Exact search algorithm to factorize large biprimes and a triprime on ibm quantum computer*, 2018. arXiv: 1805.10478.

[6] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson and D. Smith-Tone, *Status report on the third round of the nist post-quantum cryptography standardization process*, csrc.nist.gov/publications/, NISTIR 7896, https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413-upd1.pdf, 2022.

[7] E. Kiltz, V. Lyubashevsky and C. Schaffner, *A concrete treatment of fiat-shamir signatures in the quantum random-oracle model*, Cryptology ePrint Archive, Paper 2017/916, https://eprint.iacr.org/2017/916, 2017.

[8] V. Lyubashevsky and G. Neven, *One-shot verifiable encryption from lattices*, Cryptology ePrint Archive, Paper 2017/122, https://eprint.iacr.org/2017/122, 2017.

[9] S.-j. Chang, R. Perlner, W. E. Burr, M. S. Turan, J. M. Kelsey, S. Paul and L. E. Bassham, *Third-round report of the sha-3 cryptographic hash algorithm competition*, csrc.nist.gov/publications/, NISTIR 7896, https://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7896.pdf, 2012.

[10] A. Hülsing, J. Rijneveld and F. Song, *Mitigating multi-target attacks in hash-based signatures*, Cryptology ePrint Archive, Paper 2015/1256, https://eprint.iacr.org/2015/1256, 2015.

[11] M. Zhandry, *How to construct quantum random functions*, Cryptology ePrint Archive, Paper 2012/182, https://eprint.iacr.org/2012/182, 2012.

[12] M. Zhandry, 'Secure identity-based encryption in the quantum random oracle model,' in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds., ser. Lecture Notes in Computer Science, vol. 7417, Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2012, pp. 758–775. DOI: 10.1007/978-3-642-32009-5_44.

[13] S. Patranabis, D. Mukhopadhyay and S. C. Ramanna, *Function private predicate encryption for low min-entropy predicates*, Cryptology ePrint Archive, Paper 2018/1250, https://eprint.iacr.org/2018/1250, 2018.

[14] K. Gjøsteen, *Public key encryption*, wiki.math.ntnu.no/tma4160, Lecture Notes Fall 2019, https://wiki.math.ntnu.no/_media/tma4160/pke.pdf, 2019.

[15] T. Laarhoven, J. van de Pol and B. de Weger, *Solving hard lattice problems and the security of lattice-based cryptosystems*, Cryptology ePrint Archive, Paper 2012/533, https://eprint.iacr.org/2012/533, 2012.

[16] A. Langlois and D. Stehle, *Worst-case to average-case reductions for module lattices*, Cryptology ePrint Archive, Paper 2012/090, https://eprint.iacr.org/2012/090, 2012.

[17] Z. Brakerski, C. Gentry and V. Vaikuntanathan, *Fully homomorphic encryption without bootstrapping*, Cryptology ePrint Archive, Paper 2011/277, https://eprint.iacr.org/2011/277, 2011.

[18] O. Goldreich, D. Micciancio, S. Safra and J.-P. Seifert, *Approximating shortest lattice vectors is not harder than approximating closest lattice vectors∗*, cseweb.ucsd.edu, https://cseweb.ucsd.edu/~daniele/papers/GMSS.pdf, 1999.

[19] D. Stehlé, R. Steinfeld, K. Tanaka and K. Xagawa, *Efficient public key encryption based on ideal lattices*, Cryptology ePrint Archive, Paper 2009/285, https://eprint.iacr.org/2009/285, 2009.

[20] P. Q. Nguyen, *Hermite's constant and lattice algorithms*, P. Q. Nguyen and B. Vallée, Eds., 2010. DOI: 10.1007/978-3-642-02295-1_2.

[21] D. Pointcheval and J. Stern, 'Provably secure blind signature schemes,' in *Advances in Cryptology – ASIACRYPT'96*, K. Kim and T. Matsumoto, Eds., ser. Lecture Notes in Computer Science, vol. 1163, Kyongju, Korea: Springer, Heidelberg, Germany, Nov. 1996, pp. 252–265. DOI: 10.1007/BFb0034852.

[22] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner and M. Zhandry, 'Random oracles in a quantum world,' in *Advances in Cryptology – ASIACRYPT 2011*, D. H. Lee and X. Wang, Eds., ser. Lecture Notes in Computer Science, vol. 7073, Seoul, South Korea: Springer, Heidelberg, Germany, Dec. 2011, pp. 41–69. DOI: 10.1007/978-3-642-25385-0_3.

[23] M. Bellare, B. Poettering and D. Stebila, 'From identification to signatures, tightly: A framework and generic transforms,' in *Advances in Cryptology – ASIACRYPT 2016, Part II*, J. H. Cheon and T. Takagi, Eds., ser. Lecture Notes in Computer Science, vol. 10032, Hanoi, Vietnam: Springer, Heidelberg, Germany, Dec. 2016, pp. 435–464. DOI: 10.1007/978-3-662-53890-6_15.

[24] M. Abdalla, P.-A. Fouque, V. Lyubashevsky and M. Tibouchi, *Tightly-secure signatures from lossy identification schemes*, Cryptology ePrint Archive, Paper 2013/856, https://eprint.iacr.org/2013/856, 2013.

[25] E. Alkim, N. Bindel, J. Buchmann, Ö. Dagdelen, E. Eaton, G. Gutoski, J. Krämer and F. Pawlega, *Revisiting tesla in the quantum random oracle model*, Cryptology ePrint Archive, Paper 2015/755, https://eprint.iacr.org/2015/755, 2015.

[26] V. Lyubashevsky, 'Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures,' in *Advances in Cryptology – ASIACRYPT 2009*, M. Matsui, Ed., ser. Lecture Notes in Computer Science, vol. 5912, Tokyo, Japan: Springer, Heidelberg, Germany, Dec. 2009, pp. 598–616. DOI: 10.1007/978-3-642-10366-7_35.

[27] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler and D. Stehle, *Crystals – dilithium: Digital signatures from module lattices*, Cryptology ePrint Archive, Paper 2017/633, https://eprint.iacr.org/2017/633, 2017.

[28] V. Lyubashevsky, 'Digital signatures based on the hardness of ideal lattice problems in all rings,' in *Advances in Cryptology – ASIACRYPT 2016, Part II*, J. H. Cheon and T. Takagi, Eds., ser. Lecture Notes in Computer Science, vol. 10032, Hanoi, Vietnam: Springer, Heidelberg, Germany, Dec. 2016, pp. 196–214. DOI: 10.1007/978-3-662-53890-6_7.

NTNU

Norwegian University of
Science and Technology