Worst-Case to Average-Case Reductions for the SIS Problem: Tightness and Security

by

Elena Colette Bakos Lang

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

Waterloo, Ontario, Canada, 2019

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

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

Abstract

We present a framework for evaluating the concrete security assurances of cryptographic constructions given by the worst-case SIVP_γ to average-case $\mathrm{SIS}_{n,m,q,\beta}$ reductions. As part of this analysis, we present the tightness gaps for three worst-case SIVP_γ to average-case $\mathrm{SIS}_{n,m,q,\beta}$ reductions. We also analyze the hardness of worst-case SIVP_γ instances.

We apply our methodology to two SIS-based signature schemes, and compute the security guarantees that these systems get from reductions to worst-case SIVP_γ. We find that most of the presented reductions do not apply to the chosen parameter sets for the signature schemes. We propose modifications to the schemes to make the reductions applicable, and find that the worst-case security assurances of the (modified) signature schemes are, for both signature schemes, significantly lower than the amount of security previously claimed.

Acknowledgements

I would like to thank my supervisor, Alfred Menezes, for being a great mentor throughout my Master's program and for his guidance and comments during the writing of this thesis. I would also like to thank my other readers, David Jao and Douglas Stebila, for all of their helpful comments and advice about the thesis.

I must also express my gratitude to the Department of Combinatorics and Optimization and the CryptoWorks21 program for their financial support.

Finally, I would like to thank my friends and family for their support.

Table of Contents

Chapter 1

Introduction

In recent years, there has been a surge of research with the aim of building a quantum computer. Amongst many potential applications, quantum computers are known to be able to solve the hard problems on which traditional cryptographic primitives are based, such as factoring or the discrete log problem, in polynomial time [49]. Thus the invention of a large-scale quantum computer would break public-key cryptography as we know it. In order to keep (classical) computers secure, we need to design cryptographic primitives that would be safe against an adversary equipped with quantum computers.

This looming quantum threat has prompted the National Institute of Standards and Technology (NIST) to announce a Post-Quantum Cryptography Standardization contest, with the goal of standardizing some quantum-safe encryption and signature schemes. The candidates that made it to the second round were announced in January 2019. The round two candidates include 17 encryption schemes that are based on supersingular elliptic curve isogenies [28], error correcting codes [9, 13, 3, 11, 52, 5, 10, 2] and lattices [41, 34, 43, 14, 22, 26, 48, 24], and 9 signature schemes that are based on lattices [36, 44, 17], multivariate polynomial equations [18, 16, 47, 23], and hash functions [51, 27].

Lattice-based cryptosystems are among the top contenders for both signature and encryption quantum-safe standards. The security of lattice-based schemes is usually based on one of the variants of either the Learning With Errors (LWE) problem, or the Short Integer Solution (SIS) problem. One of the main arguments in favour of lattice-based primitives is the existence of so-called worst-case to average-case security proofs for the lattice primitives.

In theoretical computer science, hardness is usually measured in terms of worst-case hardness, that is, the amount of effort required to solve the hardest instance of a particular

problem. On the other hand, cryptographic applications require average-case hardness, as we want a randomly selected instance of our problem to be hard to solve.

These two concepts are not necessarily equivalent: some problems can be hard in the worst case, but easy to solve in the average case. As an example, consider the problem of deciding whether a graph is three-colourable. The problem is NP-complete. However, the average number of nodes in the depth-first search tree (over all graphs of all sizes) is only 197 [50], so the expected runtime is constant, on average.

To solve this disparity, some cryptographers like to base their cryptographic primitives on problems that have a worst-case to average-case reduction. That is, they base their primitive on a problem such that a randomly selected instance is provably as hard as solving any instance of the other problem (in particular, the hardest one). Thus, if you had an oracle that could efficiently solve the first problem on average (without necessarily being able to solve all instances), then we could use it to efficiently solve the hardest instance of the second problem. Such a reduction shows that solving the first problem on average is at least as hard as solving the second problem in the worst-case (hence the name) and thus allows us to make statements about the average-case hardness of a problem, based on the worst-case hardness of a different one – which has often been studied more, and thus is better understood.

Many lattice-based cryptographic primitives have worst-case to average-case proofs, that links the average-case hardness of a cryptosystem to the worst-case hardness of a hard lattice problem such as the approximate Shortest Independent Vector Problem (SIVP_γ). The existence of these proofs makes the lattice-based candidates favoured in the post-quantum cryptographic community, as they offer evidence to support the difficulty of breaking lattice-based cryptosystems.

However, it is important to consider the limitations of worst-case to average-case reductions in the context of a cryptosystem. Worst-case to average-case proofs exhibit a polynomial time reduction between the worst-case and average-case problems. While this gives great insight into the asymptotic hardness of the average-case problem, concrete security guarantees depend on the polynomial itself, as well as on the hardness of the worst-case problem instance. We call the size of the polynomial (for a reduction that preserves the success probability) the *tightness gap*. Increasing the tightness gap implies decreasing the hardness guarantees that the average-case problem gets from the reduction by a proportional amount, thus leaving a gap between the hardness of the worst-case problem and the hardness guarantees it gives to the average-case problem.

Two papers by Chatterjee et al. [20, 19] studied the security implications for protocols with large tightness gaps in their security proofs. In particular, in [19] the authors studied

the tightness of the SIVP_γ to LWE worst-case to average-case reduction, finding a very large tightness gap and showing that the worst-case to average-case reduction gives no security guarantees for the parameters that have been proposed for implementing the LWE cryptosystems.

The remainder of this thesis is organized as follows. Chapter 2 presents some basic notions related to lattices, and introduces lattice problems of interest in cryptography. In Chapter 3, we present a tightness analysis of the SIVP_γ to SIS worst-case to average-case reductions due to Micciancio and Regev [39], Gentry, Peikert, Vaikuntanathan [25], and Micciancio and Peikert [38]. Algorithms for solving lattice problems, including SVP, SIVP_γ and SIS are discussed in Chapter 4. In Chapter 5, we examine the security proof of a SIS-based signature scheme invented by Lyubashevsky, and determine the security guarantees that it gets from the worst-case to average-case reductions. In Chapter 6, we examine the tightness of the security proof of the signature scheme Dilithium [36], whose security (partially) depends on the SIS problem. We conclude by determining the concrete security guarantees that the worst-case to average-case reductions give Dilithium. Finally, Chapter 7 makes concluding remarks and offers directions for future research.

Chapter 2

Lattice Preliminaries

In this work, we will be using $\|x\| = (\sum_i x_i^2)^{1/2}$ to denote the ℓ_2-norm of the vector x. If the norms ℓ_p for $p \neq 2$ are used, we will refer to them explicitly as $\|\cdot\|_p$. The infinity norm of $x = (x_1, x_2, \ldots, x_n)$ is defined by $\|x\|_\infty = \max\{|x_i| : 1 \leq i \leq n\}$. Given a set of vectors $X = \{x_1, \ldots, x_k\}$, we say the norm of X is $\|X\| = \max_{1 \leq i \leq k} \|x_i\|$.

2.1 Lattices

Let $B = \{b_1, \ldots, b_n\}$ be a set of independent vectors in $\mathbb{R}^n$. We define the (full-rank) lattice with basis B as

$$\mathcal{L}(B) = \left\{ \sum_{i=1}^{n} a_i b_i : a_i \in \mathbb{Z} \right\};$$

that is, $\mathcal{L}(B)$ is the set of all integer linear combinations of the basis B[1]. If we're discussing general lattices, which may not be linked to a specific basis, we may refer to a lattice by a different name such as Λ.

For $i \in [1, n]$, we define $\lambda_i = \lambda_i(\Lambda)$ to be the smallest t such that there exists a set X of i linearly independent vectors in $\mathcal{L}(B)$ with $\|X\| = t$. In particular, λ_1 is the length of a shortest nonzero vector in $\mathcal{L}(B)$.

[1]It is also possible to define lattices that are not full rank, by considering a set of independent vectors $\{b_1, \ldots, b_n\}$ in $\mathbb{R}^m$, with $n < m$. In that case, we call n and m the rank and dimension of the lattice, respectively.

We also define the *fundamental parallelepiped*

$$\mathcal{P}(B) = \{Bx : x \in [0,1)^n\},$$

and the determinant $\det(B)$ as the *volume* of $\mathcal{P}(B)$.

Given a lattice Λ, there is an important related lattice known as the *dual lattice*.

Definition 2.1 (Dual Lattice)**.** *Let Λ be a lattice. We define the dual lattice*

$$\Lambda^* = \{x \in \mathbb{R}^n : \langle x, z \rangle \in \mathbb{Z} \ \forall z \in \Lambda\}.$$

If B is a basis for the lattice Λ, then $(B^T)^{-1}$ is a basis for the dual lattice.

The relationship between a lattice and its dual is at the core of many results about lattices, including properties of Gaussian distributions on lattices and the *smoothing parameter* (see Definition 2.4), which are fundamental to many worst-case to average-case reductions.

2.2 Gaussians and Statistical Distributions

The *statistical distance* is a measure of distance between two probability distributions X and Y that can be used to analyze probabilistic algorithms.

Definition 2.2 (Statistical Distance)**.** *If X, Y are discrete random variables over a countable set A, the statistical distance is defined as*

$$\Delta(X,Y) = \frac{1}{2} \sum_{a \in A} |Pr(X = a) - Pr(Y = a)|.$$

Similarly, if X, Y are continuous random variables over $\mathbb{R}^n$ with probability density functions T_X and T_Y respectively, the statistical distance is defined as

$$\Delta(X,Y) = \frac{1}{2} \int_{r \in \mathbb{R}^n} |T_X(r) - T_Y(r)| dr.$$

Two useful properties of the statistical distance are the fact that applying a (possibly randomized) function f cannot increase the statistical distance, that is

$$\Delta(f(X), f(Y)) \leq \Delta(X,Y)$$

and that if $X_1, \ldots, X_k$ and $Y_1, \ldots, Y_k$ are lists of independent random variables, then

$$\Delta((X_1, \ldots, X_k), (Y_1, \ldots, Y_k)) \leq \sum_{i=1}^{k} \Delta(X_i, Y_i).$$

The Gaussian family of distributions is very important in lattice theory. We first introduce the *Gaussian function* $\rho_{s,c} : \mathbb{R}^n \to \mathbb{R}$ of width s centered at c, which is defined as

$$\rho_{s,c}(x) = e^{-\pi \|(x-c)/s\|^2}.$$

If c or s are not specified, we assume that c is the origin and $s = 1$ (in which case $\rho_{1,c}$ is a probability distribution). We can extend this definition to any countable set A in the expected way:

$$\rho_{s,c}(A) = \sum_{x \in A} \rho_{s,c}(x).$$

Since $\int_{x \in \mathbb{R}^n} \rho_{s,c}(x)dx = s^n$, we can define the *Gaussian probability distribution* by its probability density function

$$D_{s,c}(x) = \frac{\rho_{s,c}(x)}{\int_{x \in \mathbb{R}^n} \rho_{s,c}(x)dx} = \frac{\rho_{s,c}(x)}{s^n}.$$

We can use these functions to define the discrete Gaussian distribution over a lattice.

Definition 2.3 (Discrete Gaussian)**.** *Let Λ be a lattice. Then the discrete Gaussian probability distribution over Λ is*

$$D_{\Lambda,s,c}(x) = \frac{D_{s,c}(x)}{D_{s,c}(\Lambda)} = \frac{\rho_{s,c}(x)}{\rho_{s,c}(\Lambda)}.$$

An important property of the discrete Gaussian is that if x is distributed according to $D_{s,c}$ conditioned on $x \in \Lambda$, then the conditional distribution of x is $D_{\Lambda,s,c}$.

It is possible to verify this property directly. However, it is easier to show it by realizing that we will only use approximations of $D_{s,c}$ in practice[2], and establishing the result for such approximations instead. Note that for all other proofs, we will assume that we can

[2]This is generally done by picking a fine (finite) grid and selecting points from the grid with probability approximately equal to $D_{s,c}$

sample $D_{s,c}$ exactly, as it leads to simpler arguments. These arguments can be made sufficiently accurate by picking a fine enough grid.

We will now show that if x is distributed according to $D_{s,c}$ conditioned on $x \in \Lambda$, then the conditional distribution of x is $D_{\Lambda,s,c}$ for a fine enough grid approximation of $D_{s,c}$. Note that if α is the volume of one cell in our grid, then the probability of obtaining a grid point x in a $D_{s,c}$ sample is very close to $\alpha D_{s,c}(x)$ (we can think of our grid approximation to $D_{s,c}$ as rounding any point x to the nearest grid point, and evaluating $D_{s,c}$ at that grid point instead of at x). On the other hand, the probability of $x \in \Lambda$ is very close to $\alpha D_{s,c}(\Lambda)$. Thus, the probability that x is distributed according to $D_{s,c}$, conditioned on $x \in \Lambda$ is

$$Pr(x \in D_{s,c} | x \in \Lambda) = \frac{Pr(x \in D_{s,c} \wedge x \in \Lambda)}{Pr(x \in \Lambda)} \approx \frac{\alpha D_{s,c}(x)}{\alpha D_{s,c}(\Lambda)} = \frac{D_{s,c}(x)}{D_{s,c}(\Lambda)} = D_{\Lambda,s,c}$$

so x is distributed according to $D_{\Lambda,s,c}$ by definition. We can make this argument as precise as needed by picking a fine enough grid.

These distributions are also used to define the lattice property known as the *smoothing parameter*. Intuitively, it is the smallest width for which the Gaussian on the lattice is ϵ-close to the uniform distribution, although the formal definition depends on the dual lattice.

Definition 2.4 (Smoothing Parameter)**.** *Let Λ be an n-dimensional lattice and $\epsilon > 0$ be a positive real. We define the smoothing parameter $\eta_\epsilon(\Lambda)$ as the smallest s such that $\rho_{1/s}(\Lambda^* \setminus \{0\}) \leq \epsilon$.*

The name *smoothing parameter* comes from the fact that if you take a random lattice point $x \in \Lambda$ and perturb it by a Gaussian of width η_ϵ, the resulting distribution is $(\epsilon/2)$-close to uniform.

2.3 Computational Problems on Lattices

There are many computational problems on lattices, many of which are used to define cryptographic primitives or prove their security. We will introduce the computational problems that are used in the worst-case to average-case reductions, as well as those used to prove the security of the Dilithium and Lyubashevsky's signature schemes.

One of the most important problems on lattices is the Shortest Vector Problem (SVP), which seeks to find a shortest nonzero vector in a lattice.

Definition 2.5 (Shortest Vector Problem (SVP))**.** *Given an n-dimensional lattice basis B, the* SVP *problem asks to find an $x \in \mathcal{L}(B)$ with $\|x\| = \lambda_1(B)$.*

This problem can be defined using any norm. The most commonly used norms are the ℓ_2 and ℓ_∞ norms. We can also define an approximation variant of SVP.

Definition 2.6 (Approximate-SVP)**.** *Let $\gamma \geq 1$. To solve the* SVP_γ *problem, one must find an $x \in \mathcal{L}(B)$ with $0 < \|x\| \leq \gamma\lambda_1(B)$.*

To measure the quality of a lattice vector v found by some algorithm, a natural choice would be to measure the SVP_γ approximation factor achieved, $\|v\|/\lambda_1(\mathcal{L}(B))$. In practice, since $\lambda_1(\mathcal{L}(B))$ is not known for a randomly chosen lattice $\mathcal{L}(B)$, the Hermite factor, which depends on the volume of the lattice, is used instead.

Definition 2.7 (Hermite Shortest Vector Problem (HSVP))**.** *Given an n-dimensional lattice basis B, the* HSVP_δ *problem asks to find an $x \in \mathcal{L}(B)$ such that $0 < \|x\| \leq \delta \sqrt[n]{vol\mathcal{L}(B)}$.*

For a problem that asks us to find a vector of length at most d, we often refer to $\delta = \frac{d}{\sqrt[n]{vol(\mathcal{L}(B))}}$ as the Hermite factor and $\delta_0 = \left(\frac{d}{\sqrt[n]{vol(\mathcal{L}(B))}}\right)^{1/n}$ as the root Hermite factor corresponding to this problem.

The root Hermite factor is a measure of the quality of a lattice vector that is independent of the dimension of the lattice. As such, it is often used to compare the quality of short vectors in lattices of different dimensions.

There is a reduction from SVP_{δ^2} to HSVP_δ, presented in [33].

The next two problems seek sets of short vectors instead of a single short one.

Definition 2.8 (Shortest Independent Vector Problem (SIVP))**.** *Given an n-dimensional lattice basis B, the* SIVP *problem asks to find a set of n linearly independent vectors $S \subset \mathcal{L}(B)$ such that $\|S\| = \lambda_n$. Similarly to the* SVP *problem, there is also an approximate version,* SIVP_γ*, for which we need $\|S\| \leq \gamma\lambda_n$.*

The following problem is a generalization of SIVP_γ, in which the function $\lambda_n(\mathcal{L}(B))$ is replaced by an arbitrary function of the lattice.

Definition 2.9 (Generalized Independent Vector Problem (GIVP^ϕ_γ))**.** *Given an n-dimensional lattice basis B and two functions γ, ϕ that can depend on the lattice $\mathcal{L}(B)$, the* GIVP^ϕ_γ *problem asks to find a set of n linearly independent vectors $S \subset \mathcal{L}(B)$ such that $\|S\| \leq \gamma(B)\phi(B)$.*

There are also many 'distance' problems which seek to find lattice vectors close to a given target. We present the ones that we will use in the worst-case to average-case reductions, although many others exist.

Definition 2.10 (Closest Vector Problem ($\mathrm{CVP}_{\mathcal{L},d}$))**.** *The closest vector problem ($CVP_{\mathcal{L},d}$) asks, on input of a lattice basis B and a point $x \in \mathbb{R}^n$ within distance d of $\mathcal{L}(B)$, to find a closest vector in $\mathcal{L}$ to x.*

Definition 2.11 (Guaranteed Distance Decoding ($\mathrm{GDD}^{\phi}_{\gamma}$) problem)**.** *On input of an n-dimensional lattice basis B, a target point t, and two functions γ, ϕ that often depend on the dimension of the lattice n, the goal is to output a lattice vector $x \in \mathcal{L}(B)$ within distance $\gamma(n)\phi(n)$ of the target, i.e. $\|x - t\| \leq \gamma(n)\phi(n)$.*

The Guaranteed Distance Decoding problem is usually instantiated with $\phi = \nu$, the covering radius of the lattice. (The *covering radius* is the maximum distance of a point $x \in \mathbb{R}^n$ from the lattice, and thus a solution is guaranteed to exist for any $\gamma \geq 1$.) The $\mathrm{GDD}^{\nu}_{\gamma}$ problem is similar to the Closest Vector Problem (CVP) but the quality of the solution is compared to the worst-case distance from the lattice, instead of the distance of the target point from the lattice.

The next problem is a bit contrived and shares its name with the GDD problem. Intuitively, it is called *incremental GDD* as it can be used to sequentially improve a potential solution to GDD (or other lattice problems) until it satisfies the appropriate bound.

Definition 2.12 (Incremental Guaranteed Distance Decoding (IncGDD))**.** *An $\mathrm{IncGDD}^{\phi}_{\gamma,g}$ instance is a tuple (B, S, t, r), where B is a lattice basis, $S \subset \mathcal{L}(B)$ is a set of n linearly independent vectors, t is a target point, and $r \geq \gamma(n)\phi(n)$ is a real number. A solution to the $\mathrm{IncGDD}^{\phi}_{\gamma,g}$ problem is a lattice vector $s \in \mathcal{L}(B)$ such that*

$$\|s - t\| \leq \frac{\|S\|}{g} + r.$$

IncGDD was used as an intermediate problem in the worst-case to average-case reduction from SIVP to SIS in [39]. Improvements to this reduction used similar problems, that were used to sequentially reduce the length of a set of independent vectors until it satisfies a bound. In particular, the reduction from [25] used an intermediate problem called *Incremental Independent Vectors Decoding*.

Definition 2.13 (Incremental Independent Vectors Decoding (IncIVD))**.** *An $\mathrm{IncIVD}^{\phi}_{\gamma,g}$ instance is a tuple (B, S, t) where B is a basis for a full-rank lattice in $\mathbb{R}^n$, $S \subset \mathcal{L}(B)$ is a full-rank set of lattice vectors such that $\|S\| \geq \gamma(n) \cdot \phi(B)$, and $t \in \mathbb{R}^n$ is a target point. The goal is to output a lattice vector $s \in \mathcal{L}(B)$ such that $\|s - t\| \leq \|S\|/g$.*

Finally, we introduce the well known Short Integer Solution (SIS) and Learning With Errors (LWE) problems, which are often used to construct one-way functions and other cryptographic primitives. The SIS and LWE problems (or related problems) are usually the average-case problems in worst-case to average-case reductions.

Definition 2.14 (SIS Problem)**.** *On input of a matrix $A \in \mathbb{Z}_q^{n \times m}$ and $\beta \in \mathbb{R}$, the goal is to find a vector $z \in \mathbb{Z}^m$ such that $Az = 0 \mod q$ and $0 < \|z\| \leq \beta$.*

Equivalently, the SIS problem asks to find a short (of norm $\|z\| \leq \beta$) nonzero vector in the (modular) lattice

$$\Lambda_q(A) = \{z \in \mathbb{Z}^m : Az \equiv 0 \mod q\}.$$

Thus the SIS problem is closely related to the shortest vector problem on a certain class of q-ary lattices.

A well known counterpart to SIS is the LWE problem introduced by Regev [45]. Defined as finding a solution to a system of slightly perturbed equations, the LWE problem can also be shown to be a dual problem to SIS.

Definition 2.15 (Learning With Errors)**.** *Let $p \geq 2$ be an integer, and let $\chi : \mathbb{Z}_p \to \mathbb{R}^+$ be a probability distribution on $\mathbb{Z}_p$. Let s be a vector in $\mathbb{Z}_p^n$ for some integer n, and define $A_{s,\chi}$ as the distribution on $\mathbb{Z}_p^n \times \mathbb{Z}_p$ where we sample $a \in \mathbb{Z}_p^n$ uniformly at random and $e \in \mathbb{Z}_p$ according to χ, and output $(a, \langle a, s \rangle + e \mod p)$.*

The $\mathrm{LWE}_{p,\chi}$ *problem asks to find s given samples from $A_{s,\chi}$ for some $s \in \mathbb{Z}_p^n$.*

Equivalently, given (A, b) where $A \in_R \mathbb{Z}_p^{m \times n}$, $b = As + e$, and the components of e are sampled according to χ, the LWE *problem asks us to find s.*

2.3.1 Dilithium Specific Definitions

This section will present the definitions that are only needed for the analysis of the signature scheme Dilithium (in Chapter 6). The definitions will first present the general concept, followed by the instantiation used by Dilithium as an example.

As a reminder, we start by presenting the definitions of a ring and a module.

Definition 2.16 (Ring)**.** *A ring $(R, +, \cdot)$ is comprised of a set R along with two binary operations $+$ and $\cdot$ such that $(R, +)$ is an abelian group, and the following conditions are satisfied:*

(i) There exists a unit element $1 \in R$ *such that* $1 \cdot a = a \cdot 1 = a$ *for all* $a \in R$.

(ii) $\cdot$ *is associative:* $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ *for all* $a, b, c \in R$.

(iii) Multiplication is distributive over addition: $a \cdot (b + c) = a \cdot b + a \cdot c$ *for all* $a, b, c \in R$.

The ring is commutative if $a \cdot b = b \cdot a$ *for all* $a, b \in R$.

As an example, the ring used in the signature scheme Dilithium is $R_q = \mathbb{Z}_q[X]/(X^n+1)$ for $n = 256$ and prime $q = 2^{23} - 2^{13} + 1$.

A module over a ring is a generalization of the concept of a vector space over a field, where the scalars come from the ring, and a multiplication function is defined between elements of the ring and elements of the module.

Definition 2.17 (Module)**.** *Let* R *be a commutative ring. An* R*-module* M *consists of an abelian group* $(M, +)$ *and an operation* $\times : R \times M \to M$ *such that for all* $x, y \in M$ *and* $s, r \in R$ *we have:*

(i) $r \times (x + y) = r \times x + r \times y$,

(ii) $(r + s) \times x = r \times x + s \times x$,

(iii) $1_R \times x = x$,

(iv) $(r \cdot s) \times x = r \times (s \times x)$.

The modules used in Dilithium are R_q-modules over the set R_q^k for some $k \in \mathbb{N}$. We next present a few definitions related to the ring R_q that will be used in Dilithium.

We define a norm on vectors over R_q as follows. The norm of $x \in \mathbb{Z}_q$ is given by $\|x\|_\infty = |x \mod^{\pm} q|$, where $w' = w \mod^{\pm} q$ is the unique element w' in the interval $[-\frac{q-1}{2}, \frac{q-1}{2}]$. The norm of $z = z_0 + z_1 X + \cdots + z_{n-1} X^{n-1} \in R_q$ is given by $\|z\|_\infty = \max_i \|z_i\|_\infty$. Finally, the norm of $w = (w_1, \ldots, w_k) \in R_q^k$ is given by $\|w\|_\infty = \max_i \|w_i\|_\infty$. (We can define the ℓ_2 norm similarly, but it is not used in the signature scheme.)

The following subsets of R_q will also be important. First, we define the set S_η as the set of elements $w \in R_q$ such that $\|w\|_\infty \leq \eta$, that is the set of elements of R_q with coefficients between $-\eta$ and η inclusively. We also need to define the set B_h, which is the set of elements of R_q that have exactly h coefficients that are either -1 or 1, the remainder being 0. We have $|B_h| = 2^h \binom{n}{h}$. The set B_h is the range of the hash function H used in

Dilithium. In particular, Dilithium picks $h = 60$, for which $|B_{60}| \geq 2^{256}$. For the definition of the hash function H, please refer to the Dilithium specification [36].

LWE and SIS can both be generalized to work over rings or over modules. Dilithium bases its security on the module versions of both problems, which we will present here. While these definitions can be given over any module (with a norm), we will only present the instantiation specific to Dilithium.

Definition 2.18 (Module Learning With Errors (MLWE))**.** *The Module-LWE distribution on $R_q^k \times R_q$ is defined by ℓ pairs $(a_i, b_i) \in R_q^k \times R_q$ such that $b_i = a_i^T s_1 + s_{2,i}$, where the secret $s = (s_1, s_2)$ is sampled uniformly from the set of short elements $S_\eta^k \times S_\eta^\ell$ and fixed for all pairs, and the a_i are sampled uniformly at random from R_q^k.*

The (decisional) Module-LWE problem [3] *asks to distinguish between m samples (a_i, b_i) chosen from the Module-LWE distribution and m samples (a_i, b_i') chosen from the uniform distribution.*

Definition 2.19 (Module Short Integer Solution (MSIS)[4])**.** *Given a matrix A selected uniformly at random from $R_q^{k \times \ell}$, the (homogenous)* $\text{MSIS}_{k,\ell,\beta}$ *problem over R_q (also denoted* MSIS_β *for brevity) asks for a $y \in R_q^{k+\ell}$ such that $0 < \|y\|_\infty \leq \beta$ and $[A|I] \cdot y = 0$.*

Note that MSIS is both a generalization of the SIS problem and of the Ring-SIS problem, as setting $R_q = \mathbb{Z}_q$ returns the SIS problem, while picking $k = 1$ gives us the Ring-SIS problem.

The next problem is used in the security proof for Dilithium as a safeguard against new message forgery. Its security is based on the combined hardness of MSIS and certain security properties of the hash function H.

Definition 2.20 (Self Target Module Short Integer Solution (SelfTargetMSIS))**.** *Let $H : \{0,1\}^* \to B_{60}$ be a hash function whose image B_{60} is a set of short elements of the ring R_q. Given a matrix A selected uniformly at random from $R_q^{m \times k}$ and a vector t selected uniformly at random from R_q^m, the* $\text{SelfTargetMSIS}_{H,m,k,\beta}$ *problem over R_q (also denoted* $\text{SelfTargetMSIS}_\beta$ *for brevity) asks for a $y \in R_q^{m+k+1}$ with $0 < \|y\|_\infty \leq \beta$ and a message M such that $H(M \| [A|t|I] \cdot y) = c$, where $y = \begin{bmatrix} r_1 \\ c \\ r_2 \end{bmatrix}$ with $r_1 \in R_q^k, r_2 \in R_q^m$, and $c \in B_{60}$.*

[3] We can think of the computational version of MLWE as a Learning With Errors problem in the R_q module over R_q^k.

[4] We can think of the MSIS problem as a Short Integer Solution problem in the R_q module over R_q^k.

2.4 Miscellaneous

The following is a method used for finding an orthogonal basis of $\mathbb{R}^n$[5].

Definition 2.21 (Gram-Schmidt Orthogonalization (GSO))**.** *Let $\{b_1, \ldots, b_n\}$ be a basis of a subspace of $\mathbb{R}^n$. The Gram-Schmidt vectors are a basis $\{b_1^*, \ldots, b_n^*\}$ of $\mathbb{R}^n$ such that*

(i) $b_1^* = b_1$*; and*

(ii) for all $2 \leq i \leq n$, $b_i^* = b_i - \text{proj}_i(b_i)$ *where* $\text{proj}_i(b_i) = \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$ *with* $\mu_{i,j} = \frac{\langle b_i, b_j \rangle}{\|b_j^*\|^2}$.

We call the $\mu_{i,j}$ the Gram-Schmidt coefficients.

Finally, we present the following Theorem, which is used in security reductions for signature schemes in the random oracle model.

Theorem 2.22 (Generalized Forking Lemma ([12, Lemma 1]))**.** *Let $t \geq 1$ be an integer, IG be a probability distribution from which x is drawn, and let H be a set of size h from which elements are drawn uniformly at random. Let A be a randomized algorithm such that*

$$A(x, h_1, \ldots, h_t) = (J, \sigma)$$

where $J \in [0, t]$. Let acc denote the probability that $J \geq 1$ if A is given inputs as described above. We can then define a Forking Algorithm $F_A(x)$ as folows:

1. *Pick a random tape r for A.*
2. *Pick $h_1, \ldots, h_t$ uniformly at random from H.*
3. *Run A on input $(x, h_1, \ldots, h_t; r)$ to produce (J, σ).*
4. *If $J = 0$, return $(0, 0, 0)$.*
5. *Pick $h'_J, \ldots, h'_t$ unformly at random from H.*
6. *Run A on input $(x, h_1, \ldots, h_{J-1}, h'_J, \ldots, h'_t; r)$ to produce (J', σ').*
7. *If $J' = J$ and $h_J \neq h'_J$, then return $(1, \sigma, \sigma')$, otherwise return $(0, 0, 0)$.*

[5]The same definition also works for any subspace of $\mathbb{R}^n$.

Then the probability that F_A outputs a triple $(1, \sigma, \sigma')$, given an input x chosen uniformly at random from IG is at least

$$acc \cdot \left(\frac{acc}{t} - \frac{1}{h} \right).$$

In particular, if $\mathcal{F}$ is a forger for a signature scheme that succeeds with probability δ, makes h queries to the random oracle and s queries to the signing oracle (with domain D_H), then the Forking Algorithm produces two different valid signatures σ, σ' for the same message m with probability

$$\delta \cdot \left(\frac{\delta}{h+s} - \frac{1}{|D_H|} \right) \approx \frac{\delta^2}{h+s}.$$

Chapter 3

Worst-Case to Average-Case Reduction

This chapter describes existing worst-case to average-case reductions for the Short Integer Solution (SIS, see Definition 2.14) problem. We will first describe Micciancio and Regev's reduction [39] in detail, and analyze its tightness gap. The reduction is in three parts. In Section 3.1 we have a worst-case to average-case reduction from the Incremental Guaranteed Distance Decoding ($\text{IncGDD}^{\eta_\epsilon}_{\beta\sqrt{n},g}$, see Definition 2.12) problem to the $\text{SIS}_{n,m,q,\beta}$ problem. This is followed in Section 3.2 by a worst-case to worst-case reduction from $\text{GIVP}^{\eta_\epsilon}_{g\gamma}$ to $\text{IncGDD}^{\eta_\epsilon}_{\gamma,g}$. Our study of Micciancio and Regev's reduction is concluded in Section 3.3, which presents a reduction from $\text{SIVP}_\gamma = \text{GIVP}^{\lambda_n}_\gamma$ to $\text{GIVP}^{\eta_\epsilon}_\gamma$, combines the results of the previous sections into a reduction from worst-case SIVP_γ to average-case $\text{SIS}_{n,m,q,\beta}$, and describes the reduction's tightness gap. Figure 3.1 gives an overview of the strategy of the reduction.

$$\text{SIVP}_{8\beta\sqrt{n}\omega(\sqrt{\log n})} = \text{GIVP}^{\lambda_n}_{8\beta\sqrt{n}\omega(\sqrt{\log n})} \to \text{GIVP}^{\eta_\epsilon}_{8\beta\sqrt{n}} \to \text{IncGDD}^{\eta_\epsilon}_{\beta\sqrt{n},8} \to \text{SIS}_{n,m,q,\beta}$$

Figure 3.1: SIVP_γ to SIS Reduction Overview

Finally, Sections 3.4 and 3.5 present the improved worst-case to average-case SIS reductions of Gentry, Peikert and Vaikuntanathan [25] and Micciancio and Peikert [38], which follow the same template as the reduction from [39] but achieve better parameters, and analyze their tightness gaps.

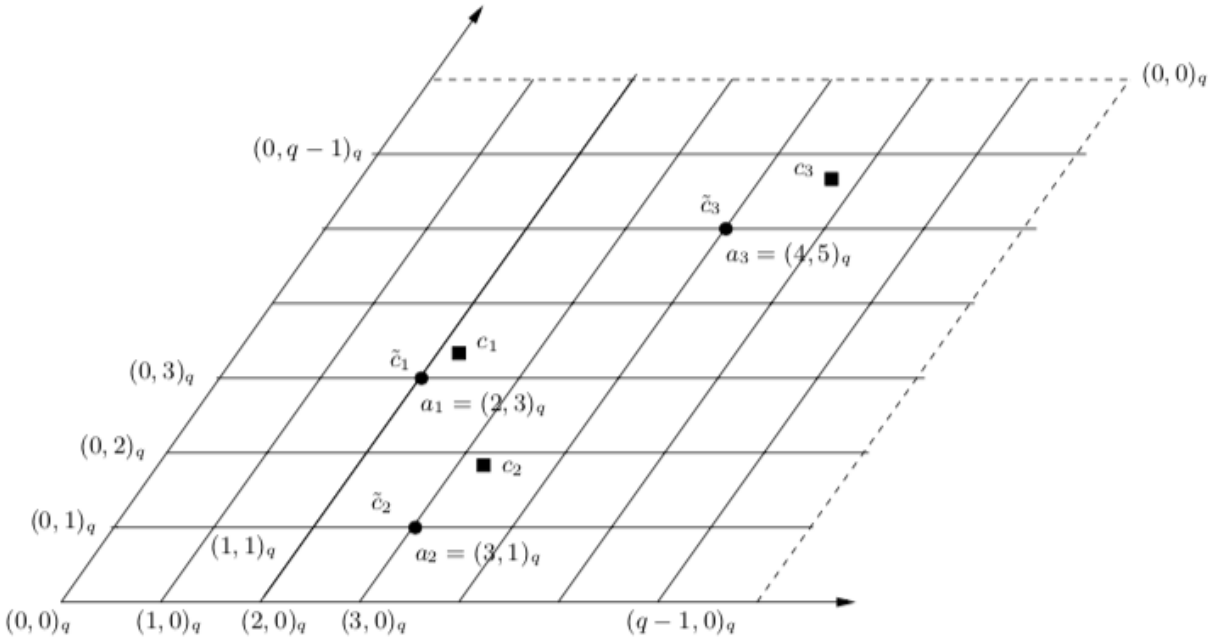

Figure 3.2: Simplified Reduction Diagram

3.1 IncGDD to SIS

3.1.1 Simplified Overview

We will first describe a simplified reduction. Suppose we are given an IncGDD instance (B, S, t, r) and access to an SIS oracle for parameters n, $m = n \log n$, $q = n^4$ and $\beta = n$, that on input of a random matrix $A \in \mathbb{Z}_q^{n \times m}$ outputs with high probability a vector $z \in \mathbb{Z}^m$ such that $Az = 0 \mod q$ and $0 < \|z\|_2 \leq \beta$. For the purposes of the simplified reduction, assume the target t is 0 and that the independent set S is B. We will also assume that $g = 4$, but will ignore ϕ, γ and r for the purposes of this simplified reduction. Note that although this IncGDD instance is trivial, as $t = 0$ is always a vector in the lattice $\mathcal{L}(B)$, we will nevertheless seek to find a non-zero lattice vector close to the target $t = 0$.

In the setup phase, subdivide the fundamental parallelepiped $\mathcal{P}(B)$ into a grid of q^n smaller parallelepipeds of equal size by subdividing each of the n sides of $\mathcal{P}(B)$ into q segments of equal size. Label each parallelepiped with an element of $\mathbb{Z}_q^n$ in the natural way, starting with $\vec{0}$ in the lower left corner of the parallelepiped; see Figure 3.2.

The first ingredient of the reduction is a sampling procedure $\mathcal{S}$ that samples m pairs (c_i, y_i), where each c_i is distributed uniformly at random mod $\mathcal{P}(B)$, and y_i is a lattice vector that's close to c_i (note that y_i does not have to be the *closest* lattice vector to c_i, only one of the vectors close to it; this property is fundamental for the reduction to work).

For each c_i, let $\widetilde{c}_i = B\lfloor qB^{-1}c_i \rfloor / q$ be the lower left corner of the subdivided parallelepiped that c_i is in. Notice that the distance between c_i and $\tilde{c}_i$ is at most $n\|B\|/q = \|B\|/n^3$. Define $a_i = \lfloor qB^{-1}c_i \rfloor \mod q$ to be the element of $\mathbb{Z}_q^n$ corresponding to the parallelepiped that c_i is in. This is depicted in Figure 3.2.

Define $C = [c_1 \dots c_m]$, $\widetilde{C} = [\widetilde{c}_1 \dots \widetilde{c}_m]$, and $A = [a_1 \dots a_m]$. Notice that since each c_i is distributed uniformly in $\mathcal{P}(B)$, each a_i is distributed uniformly in $\mathbb{Z}_q^n$. Therefore A is uniformly distributed in $\mathbb{Z}_q^{n \times m}$. We call the SIS oracle with the matrix A, and get back a short vector z such that $Az = 0 \mod q$ and $\|z\|_2 \leq \beta$. By properties of norms, we have $\|z\|_1 \leq \sqrt{m}\|z\|_2 \leq \sqrt{m}\beta \leq n^2$.

Observe that the mapping of $\mathbb{Z}_q^n$ vectors to the subdivided parallelepiped is such that all lattice vectors are labelled $\vec{0}$. Since $Az = 0 \mod q$, $\widetilde{C}z$ must then correspond to a point labelled $\vec{0}$ in the superlattice grid, and hence it is a lattice point. Finally, since each y_i was chosen to be a lattice point, Yz is also a lattice point as a linear combination of lattice points.

To conclude the simplified reduction, note that both $\widetilde{C}z$ and Yz are close to Cz, since the sampling procedure picked each (y_i, c_i) pair to be close together, and since $\widetilde{c}_i$ is simply a rounding of the corresponding c_i. Thus, $(Y - \widetilde{C})z$ is short, and is in the lattice as it is a difference of lattice vectors. We can bound its length by the triangle inequality, using the distances $\|Yz - Cz\|$ and $\|Cz - \widetilde{C}z\|$. It turns out that the dominant distance is usually that between Cz and $\widetilde{C}z$, which we can bound by

$$\begin{aligned}
\|Cz - \widetilde{C}z\|_2 &= \|\sum_{i=1}^{n}(c_i - \widetilde{c}_i)z_i\|_2 \\
&\leq \sum_{i=1}^{n}\|c_i - \widetilde{c}_i\|_2|z_i| \\
&\leq \sum_{i=1}^{n}\frac{\|B\|}{n^3}|z_i| \\
&= \|z\|_1\frac{\|B\|}{n^3} \leq n^2\frac{\|B\|}{n^3} = \frac{\|B\|}{n} \\
&\ll \frac{\|B\|}{4}
\end{aligned}$$

by the inequalities above. For the simplified reduction we ignore the distance $\|Yz - Cz\|$, which will be bounded by r (and dependent on ϕ, γ) in the actual proof. Therefore, $(Y - \widetilde{C})z$ is a solution to our IncGDD instance.

Recall that we originally made two simplifying assumptions, that $t = 0$ and $S = B$. To generalize the algorithm to work for any S (in the algorithms that call the IncGDD oracle we often consider $\|S\| \ll \|B\|$), we will first sample things uniformly in $\mathcal{P}(B)$, map them to a uniform distribution on $\mathcal{P}(S)$, and then continue as above, subdividing $\mathcal{P}(S)$ into smaller parallelepipeds. To account for the fact that t does not have to be 0 (and in general will not be), we simply modify the sampling procedure to take as additional input a point t' and to output a pair (c, y), where y is close to $c + t'$. By carefully choosing the vectors t' that we give as input to the sampling algorithm we can guarantee the solution will be close to the target t with reasonable probability.

In Section 3.1.2 we will describe the general algorithm in detail, and present a proof of its correctness.

3.1.2 Formal Reduction

Lemma 3.1 (Sampling Procedure)**.** *There is a polynomial time algorithm $S(B, t, s)$ that on input of an n-dimensional lattice basis B, a vector $t \in \mathbb{R}^n$, and real number $s \geq \eta_\epsilon(B)$, outputs a pair $(c, y) \in \mathcal{P}(B) \times \mathcal{L}(B)$ such that*

(i) c is $(\epsilon/2)$-close to uniform;

(ii) For any $\hat{c} \in \mathcal{P}(\mathcal{B})$, the conditional distribution of y given $c = \hat{c}$ is $D_{\mathcal{L}(B),s,t+c}$.

Proof. Proof Sketch The sampling procedure $S(B, t, s)$ is the following:

1. Generate a noise vector r with probability density $D_{s,t}$.
2. Output $c = -r \mod \mathcal{P}(B)$ and $y = r + c$.

To show that c is $(\epsilon/2)$-close to uniform, we would need Fourier analysis as well as results about the properties of the Gaussian distribution and of the smoothing parameter. As the details of this portion of the proof are not important for the remainder of the discussion, the proof of this property and a discussion of the relevant properties of the smoothing parameter are omitted from this work, but can be found in [39].

To show the second property, fix any $\hat{c} \in \mathcal{P}(B)$. By definition, the distribution of $r + \hat{c}$ is $D_{s,t+\hat{c}}$. Conditioning on $c = \hat{c}$ is the same as conditioning on $r + \hat{c} \in \mathcal{L}(B)$, since we only consider the value of $c, \hat{c}$ modulo $\mathcal{P}(B)$. By the discussion in Chapter 2, the distribution of $r + \hat{c}$ conditioned on $r + \hat{c} \in \mathcal{L}(B)$ is $D_{\mathcal{L}(B),s,t+\hat{c}}$ as desired.

Note that this sampling procedure is efficient, as there are efficient algorithms for sampling from a distribution with density function $D_{s,t}$. □

The sampling procedure is used in the IncGDD to SIS reduction to create inputs for the so-called *Combining Procedure*, which takes in a set of pairs (c, y) from the sampling procedure, maps each c to a $\tilde{c}$ and a vector a, and obtains a short linear combination of the vectors a that sums to 0 through a call to the SIS-oracle $\mathcal{F}$.

Lemma 3.2 (Combining Procedure)**.** *On input an n-dimensional lattice $\mathcal{L}(B)$, a full rank sublattice $\mathcal{L}(S) \subset \mathcal{L}(B)$, m vectors $C = [c_1 \dots c_m] \in \mathcal{P}(B)^m$, a positive integer q, a bound β and an $\mathrm{SIS}_{n,m,q,\beta}$ oracle $\mathcal{F}$, there is a polytime algorithm that makes a single oracle call $\mathcal{F}(A) = z$ and outputs a vector $x \in \mathbb{R}^n$ such that:*

(i) If C is uniformly distributed, then the input A to the oracle is also uniformly distributed;

(ii) If the oracle's answer z was in $\Lambda_q(A)$, then the output vector x is in $\mathcal{L}(B)$; and

(iii) The distance between x and Cz is at most $\sqrt{mn}\|S\| \cdot \|z\|/q$.

Proof. We define the combining procedure as follows:

1. Generate m uniformly random vectors $v_i \in \mathcal{L}(B) \mod \mathcal{P}(S)$.
2. Let $w_i = c_i + v_i \mod \mathcal{P}(S)$ for each $i = 1, \dots, m$.
3. Let $a_i = \lfloor qS^{-1}w_i \rfloor \in \mathbb{Z}_q^n$, for each $i = 1, \dots, m$.
4. Call the $\mathrm{SIS}_{n,m,q,\beta}$ oracle with $A = [a_1 \dots a_m]$.
5. Output $(C - W + SA/q)z$, where $W = [w_1 \dots w_m]$.

To show the first property, note that the sets $v + \mathcal{P}(B) \mod \mathcal{P}(S)$ for $v \in \mathcal{L}(B) \mod \mathcal{P}(S)$ form a partition of $\mathcal{P}(S)$ into sets of equal volume. This implies that if C is distributed uniformly, then so are both W and A.

For the second property, consider the output vector

$$(C - W + SA/q)z = \sum_i (c_i - w_i)z_i + SAz/q.$$

We have that SAz/q is in $\mathcal{L}(B)$ since $Az \equiv 0 \mod q$, whence Az/q is an integer vector. Additionally, $c_i - w_i = ((c_i + v_i) - w_i) - v_i$ is in $\mathcal{L}(B)$, as $c_i + v_i \equiv w_i \mod \mathcal{L}(S) \subseteq \mathcal{L}(B)$ and $v_i \in \mathcal{L}(B)$. Thus, $\sum_i (c_i - w_i)z_i$ and hence the output vector, $(C - W + SA/q)z$, are in the lattice.

For the third property, we have that

$$\begin{aligned}
\|x - Cz\| &= \|(C - W + SA/q)z - Cz\| \\
&= \|Wz - SAz/q\| \\
&= \|\sum_i (w_i - Sa_i/q)z_i\| \\
&= \frac{1}{q}\|S\sum_i (qS^{-1}w_i - a_i)z_i\| \\
&= \frac{1}{q}\|S\sum_i (u_i - \lfloor u_i \rfloor)z_i\|
\end{aligned}$$

where $u_i = qS^{-1}w_i$. Then since the vector $\sum_i (u_i - \lfloor u_i \rfloor)z_i$ has all its entries bounded by $\sum_i |z_i| \leq \sqrt{m}\|z\|$, the triangle inequality gives us

$$\|S\sum_i (u_i - \lfloor u_i \rfloor)z_i\| \leq n\sqrt{m}\|z\|\|S\|$$

and thus $\|x - Cz\| \leq n\sqrt{m}\|z\|\|S\|/q$. □

Finally, we use the combining procedure to obtain a solution to the IncGDD instance. The proof of this is presented in the next Theorem:

Theorem 3.3 (IncGDD to SIS reduction)**.** *Suppose*

(i) $g(n) > 0$ *and* $\epsilon(n) = n^{-\omega(1)}$*; and*

(ii) $m(n), \beta(n) = n^{O(1)}$ *and* $q(n) \geq g(n)\sqrt{m(n)}n\beta(n)$.

Then there exists a polynomial time reduction from solving $\text{IncGDD}^{\eta_\epsilon}_{\gamma,g}$ *(for* $\gamma = \beta\sqrt{n}$*) in the worst case to solving* $\text{SIS}_{n,m,q,\beta}$ *on the average with non-negligible probability (at least* $\frac{1}{6\beta m}$*), with a single call to the* SIS *oracle.*

Proof. Suppose n is fixed, and let $g = g(n), m = m(n), \epsilon = \epsilon(n)$ and $q = q(n)$. Given an $\mathrm{IncGDD}^{\eta_\epsilon}_{\gamma,g}$ instance (B, S, t, r) and an SIS oracle $\mathcal{F}$ that succeeds with probability δ on uniformly random inputs, the reduction does the following:

1. Pick an index $j \in \{1, \ldots, m\}$, and $\alpha \in \{-\beta, \ldots, -1, 1, \ldots, \beta\}$ uniformly at random. Let
$$t_i = \begin{cases} -t/\alpha, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$
2. Call the sampling procedure m times to get $(c_i, y_i) = S(B, t_i, 2r/\gamma)$.
3. Let $C = [c_1 \ldots c_m]$ and $Y = [y_1 \ldots y_m]$.
4. Call the combining procedure with inputs B, S, C, q and the $\mathrm{SIS}_{n,m,q,\beta}$ oracle to get back the vector x (and the output z of the oracle). Output $s = x - Yz$.

We will prove that this reduction takes polynomial time and succeeds with nonnegligible probability.

Suppose $j = j'$ and $\alpha = \alpha'$, and denote by $\delta_{j',\alpha'}$ the probability that the output z' of the oracle on a uniformly random input A' satisfies

$$z'_{j'} = \alpha' \quad \text{and} \quad z' \in \Lambda_q(A') \setminus \{0\} \quad \text{and} \quad \|z'\| \leq \beta.$$

Since z' is non-zero and bounded by β, we must have

$$\sum_{j,\alpha} \delta_{j,\alpha} \geq \delta.$$

Thus, there exists a (j', α') pair such that $\delta_{j',\alpha'} \geq \frac{\delta}{2\beta m}$ (which is non-negligible). For the rest of the reduction, assume that $j = j', \alpha = \alpha'$, as the event $j = j', \alpha = \alpha'$ happens with probability $\frac{1}{2\beta m}$ which is nonnegligible.

We now show that if $j = j'$ and $\alpha = \alpha'$, the remainder of the reduction succeeds with high probability. This will complete the proof, as it implies that the reduction succeeds with nonnegligible probability.

Let H be the event that $\mathcal{F}$ is successful and that $z_j = \alpha$. We will use properties of the statistical distance to show that the input C of the combining procedure to the SIS oracle is very close to uniform, which implies that H happens with probability very close to $\delta_{j,\alpha}$.

We will then show that the probability of success of the reduction conditioned on H is at least $\frac{1}{3}$, to conclude the proof.

By the Sampling Procedure, the distance between each c_i and the uniform distribution on $\mathcal{P}(B)$ is $\leq \epsilon/2$. Hence, as all of the c_i's are independent, we have

$$\Delta(C,\mathcal{U}(\mathcal{P}(B))^m) = \Delta([c_1 \dots c_m],\mathcal{U}(\mathcal{P}(B))^m) \leq \sum_{i=1}^{m} \Delta(c_i,\mathcal{U}(\mathcal{P}(B))) \leq \frac{\epsilon m}{2}.$$

Recall that the statistical distance between two random variables cannot increase by applying a (possibly randomized) function f. Hence, the distance between z and a z' obtained from a uniformly random input to the combining procedure is $\leq \frac{\epsilon m}{2}$, and the event H holds with probability at most $\delta_{j,\alpha} - \epsilon m/2$.

Finally, we show that the reduction conditioned on H succeeds with probability at least $1/3$. To establish this, we will show the stronger fact that the reduction succeeds with probability at least $1/3$ given fixed values of C, A and $z = \mathcal{F}(A)$ for which H is satisfied. Thus, assume we have C, A and z such that $z_j = \alpha$, $z \in \Lambda_q(A) \setminus \{0\}$ and $\|z\| \leq \beta$.

Note that

$$\|s - t\| = \|x - Cz + Cz - Yz - t\| \leq \|x - Cz\| + \|(C - Y)z - t\|$$

by definition of s and the triangle inequality. Notice that $Tz = [t_1 \dots t_m]z = -t/\alpha \cdot \alpha = -t$ by definition of T. Also note that $q \geq g\sqrt{m}n\beta$ or, equivalently, $\frac{1}{g} \geq \frac{n\sqrt{m}\beta}{q}$, and thus by Lemma 3.2 the distance between x and Cz is bounded by $\frac{\sqrt{m}n\|S\|\beta}{q} \leq \|S\|/g$. Thus, we have

$$\|s - t\| \leq \frac{\|S\|}{g} + \|(C - Y + T)z\|.$$

It remains to show that $\|(C - Y + T)z\| = \|(Y - (C + T))z\|$ is bounded by r. We will need the following two lemmas.

Lemma 3.4. *Let $v_1, \dots, v_m$ be m vectors chosen independently from probability distributions $V_1, \dots, V_m$ such that $E[\|v_i\|^2] \leq \ell$ and $\|E[v_i]\|^2 \leq k$ for all i. Then for any $z \in \mathbb{R}^m$,*

$$E[\|\sum_i v_i z_i\|^2] \leq (\ell + k \cdot m)\|z\|^2.$$

Proof. By linearity of expectation and the fact that $\sum_i |z_i| \leq \sqrt{m}\|z\|$, we have

$$\begin{aligned}
E[\|\sum_i v_i z_i\|^2] &= \sum_{i,j} z_i z_j E[\langle v_i, v_j \rangle] \\
&= \sum_i z_i^2 E[\|v_i\|^2] + \sum_{i \neq j} z_i z_j \langle E[v_i], E[v_j] \rangle \\
&\leq \ell \sum_i z_i^2 + \sum_{i \neq j} z_i z_j \|E[v_i]\| \|E[v_j]\| \text{ (by Cauchy-Schwartz inequality)} \\
&\leq \ell \|z\|^2 + \sum_{i \neq j} z_i z_j \max_{i,j} \|E[v_i]\|^2 \\
&\leq \ell \|z\|^2 + k \sum_{i \neq j} z_i z_j \\
&\leq \ell \|z\|^2 + k (\sum_i |z_i|)^2 \\
&\leq \ell \|z\|^2 + k (\sqrt{m}\|z\|)^2 \\
&= (\ell + km)\|z\|^2.
\end{aligned}$$

□

Lemma 3.5 ([39, Lemma 4.3])**.** *For any n-dimensional lattice Γ, vector $c \in \mathbb{R}^n$, $0 < \epsilon < 1$, and $s \geq \eta_\epsilon(\Gamma)$ (where η_ϵ is the smoothing parameter of the lattice), the following two properties hold:*

(i) $\|E_{x \sim D_{\Gamma,s,c}}[x - c]\|^2 \leq \left(\frac{\epsilon}{1-\epsilon}\right)^2 s^2 n$, *and*

(ii) $E_{x \sim D_{\Gamma,s,c}}[\|x - c\|^2] \leq \left(\frac{1}{2\pi} + \frac{\epsilon}{1-\epsilon}\right) s^2 n$.

As the proof is very technical, we won't present it here. It can be found in [39].

We will now prove that $\|(Y - (C + T))z\| \leq r$ with high probability. Since each y_i is distributed according to $D_{\mathcal{L}(B),s,c_i+t_i}$, Lemma 3.5 implies that

$$E[\|y_i - (c_i + t_i)\|^2] \leq \left(\frac{1}{2\pi} + \frac{\epsilon}{1-\epsilon}\right) s^2 n,$$

and

$$\|E[y_i - (c_i + t_i)]\|^2 \leq \left(\frac{\epsilon}{1-\epsilon}\right)^2 s^2 n.$$

Since $y_1, \ldots, y_m$ are independent, Lemma 3.4 implies that

$$E\left[\|\sum_{i=1}^{m}(y_i - (c_i + t_i))z_i\|^2\right] \leq \left(\frac{1}{2\pi} + \frac{\epsilon}{1-\epsilon} + \frac{\epsilon}{1-\epsilon}^2 m\right) s^2 n\|z\| \leq \frac{\|z\|s^2 n}{6}$$

since $\frac{1}{2\pi} \leq \frac{1}{6}$, $\epsilon = n^{-\omega(1)}$, and $m = n^{O(1)}$ imply that $\frac{\epsilon}{1-\epsilon}$ and $(\frac{\epsilon}{1-\epsilon})^2 m$ are negligible for sufficiently large n. As $\|z\| \leq \beta$, $s = 2r/\gamma$ and $\gamma = \beta\sqrt{n}$, we have

$$\frac{\|z\|s^2 n}{6} \leq \frac{2}{3}r^2.$$

By Markov's inequality, we know that for a random variable X, $Pr(X \geq a) \leq \frac{E(X)}{a}$. Hence, we have

$$Pr[\|(Y - (C+T))z\| > r] \leq Pr[\|(Y - (C+T))z\|^2 > r^2] \leq \frac{2r^2}{3} \cdot \frac{1}{r^2} = \frac{2}{3}.$$

The reduction succeeds if $\|(Y - (C+T))z\| \leq r$. By the above, this happens with probability $\geq \frac{1}{3}$, and hence the reduction succeeds with non-negligible probability

$$Pr[\text{reduction succeeds}] \geq \frac{1}{3 \cdot 2\beta m}.$$

This concludes the proof.

Note that the big components of the runtime of this reduction are

1. The time it takes to sample m pairs using the Sampling Procedure, where each pair takes the time needed to sample one element from $D_{s,t}$ ($O(m)$ operations);
2. A (small) number of matrix-vector multiplications (each requiring $O(m^2)$ operations);
3. One oracle call.

We make the simplifying assumption that the cost of an oracle call is at least as large as sampling vectors and matrix-vector multiplications, thus making the number of oracle calls the dominant term in the runtime. Note that this is a reasonable assumption, as even checking that a vector x is an SIS solution takes a matrix-vector multiplication, and we can expect finding an SIS solution to take at least as long as checking whether it is valid. □

3.1.3 Guaranteeing SIS Solutions

It is useful to restrict the remaining reductions and applications to $\text{SIS}_{n,m,q,\beta}$ instances that are guaranteed to have a solution (otherwise a reduction doesn't give us any useful information; if $\text{SIS}_{n,m,q,\beta}$ is unsolvable, then it is trivially at least as hard as any solvable problem). The following Theorem gives us a condition for the existence of a vector of norm at most β in the SIS lattice. If it is left unspecified, we will implicitly assume in the coming reductions that β satisifies this bound and that $\text{SIS}_{n,m,q,\beta}$ has a solution.

Theorem 3.6. *For any q and any $A \in \mathbb{Z}_q^{n\times m}$, the SIS instance (q, A, β) is guaranteed to have a solution if $\beta \geq \sqrt{m}q^{n/m}$.*

Proof. Consider all vectors in $\mathbb{Z}^m$ with coordinates in $\{0, \ldots, q^{n/m}\}$ (which have norm $\leq \sqrt{m}q^{n/m}$ as needed). There are $(\lfloor q^{n/m} \rfloor + 1)^m \geq (q^{n/m})^m = q^n$ of them, so the pigeonhole principle implies there must be two vectors z_1, z_2 such that $Az_1 = Az_2 \mod q$. (Think as Az as a function that maps a vector in $\mathbb{Z}_q^m$ to a vector in $\mathbb{Z}_q^n$. As there are at least q^n vectors in the set we're considering, they cannot all map to different values.) Then $z = z_1 - z_2$ satisfies $Az = Az_1 - Az_2 = 0 \mod q$ and $\|z_1 - z_2\| \leq \sqrt{m}q^{n/m}$ as each of the coordinates is between $-q^{n/m}$ and $q^{n/m}$. □

3.2 GIVP to IncGDD

The original paper by Micciancio and Regev [39] presented multiple worst-case to worst-case reductions from well known lattice problems to IncGDD. These include both adaptive and non-adaptive reductions from $\text{GDD}_\gamma^{\lambda_n}$, SIVP_γ and GapCRP_γ. Since SIVP_γ is arguably the most studied of the three in lattice cryptography, and many hard problems reduce to it, we will only present the SIVP_γ to IncGDD reduction.

Additionally, since the adaptive reductions have better parameters and tighter proofs, we will only present the adaptive versions.

Theorem 3.7 (GIVP to IncGDD)**.** *For any $\gamma(n) \geq 1$ and any ϕ, there exists a polynomial time reduction from $\text{GIVP}_{8\gamma}^{\phi}$ to $\text{IncGDD}_{\gamma,8}^{\phi}$. This reduction makes $O(n^2)$ calls to the* IncGDD *oracle and succeeds with high probability.*

Proof. Given a basis B and an $\text{IncGDD}_{\gamma,8}^{\phi}$ oracle, we want to construct a basis S of $L(B)$ such that $\|S\| \leq 8\gamma(n)\phi(B)$. This is done iteratively. Let s_i be the longest vector at the current step, with $S = \{s_1, \ldots, s_n\}$. Let t be a vector orthogonal to $s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n$

of length $\|S\|/2$. Call the IncGDD oracle on $(B, S, t, \|S\|/8)$. If it succeeds, we have a lattice vector u at distance $\leq \|S\|/8 + \|S\|/8 = \|S\|/4$ from t. Notice that $\|u\| \leq 3\|S\|/4$, and that u is independent from the vectors $s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n$ as the angle between u and the hyperplane spanned by the vectors $s_j, j \neq i$ is at least $\pi/4$ (t was orthogonal, of length $\|S\|/2$, and u is at distance at most $\|S\|/4$ from it).

When the oracle call fails, it must be the case that $\|S\|/8 \leq \gamma(n)\phi(B)$ or $\|S\| \leq 8\gamma(n)\phi(B)$ as we need.

We need to show that this terminates in a polynomial number of steps. To see this, note that the starting value of $\log(\prod \|s_i\|)$ is polynomial in n, and decreases by a constant every turn. Indeed, we can use the LLL algorithm to ensure the starting S satisfies $\|S\| \leq 2^n \lambda_n(B)$, which implies $\log(\prod \|s_i\|) \leq \log(\prod \|S\|) \leq \log(2^n \lambda_n)^n$. Then note that $\log \prod_i \|s_i\|$ decreases by $\log \frac{4}{3}$ at each iteration, as the chosen s_i is replaced by a new vector s_i' satisfying $\|s_i'\| \leq \frac{3}{4}\|s_i\|$.

Finally, note that the quantity $\log \prod_i \|s_i\|$ will always be bounded from below by $\log \prod_i \min\{\|s_i\|, \frac{3}{4}\lambda_n(B)\}$ (as any s_i that is picked at any iteration must satisfy $\|s_i\| \geq \lambda_n(B)$). Thus, the number of iterations will be bounded by

$$\frac{\log(2^n \lambda_n)^n - \log \prod_i \min\{\|s_i\|, \frac{3}{4}\lambda_n(B)\}}{\log 4/3} = O(n^2),$$

which is polynomial. □

3.3 SIVP to SIS

Recall that $\mathrm{SIVP}_\gamma = \mathrm{GIVP}_\gamma^{\lambda_n}$. Thus, to be able to build a reduction from SIVP_γ to SIS, we first need a reduction from $\mathrm{SIVP}_\gamma = \mathrm{GIVP}_\gamma^{\lambda_n}$ to $\mathrm{GIVP}_{\gamma'}^{\eta_\epsilon}$ for some γ'. We present this reduction in the following Lemma.

Lemma 3.8. *There exists a negligible $\epsilon = \epsilon(n)$ for which there is a (trivial) polynomial time reduction from $\mathrm{SIVP}_{\gamma\omega(\sqrt{\log n})}$ to $\mathrm{GIVP}_\gamma^{\eta_\epsilon}$.*

Proof. By properties of the smoothing parameter, there exists a negligible $\epsilon(n)$ such that $\eta_\epsilon(\Lambda) \leq \sqrt{\omega(\log n)}\lambda_n(\Lambda)$ for any n-dimensional lattice Λ.

Thus we have a polynomial time reduction from $\mathrm{SIVP}_{\gamma\omega(\sqrt{\log n})}$ to $\mathrm{GIVP}_\gamma^{\eta_\epsilon}$ as desired. □

Finally, we can combine all of the reductions so far to get a reduction from SIVP_γ to $\text{SIS}_{n,m,q,\beta}$. Recall from Section 3.1.3 that we need $\beta \geq \sqrt{m}q^{n/m}$ to guarantee the existence of $\text{SIS}_{n,m,q,\beta}$ solutions. If that holds, then we can construct a reduction from SIVP_γ to SIS.

Theorem 3.9. *Let $n \in \mathbb{N}, \beta = n^{O(1)}, m = n^{O(1)}, q \geq 8n\sqrt{m}\beta$ and $\gamma = 8\sqrt{n}\omega(\sqrt{\log n})\beta$. Then there exists a reduction from worst-case SIVP_γ to average-case $\text{SIS}_{n,m,q,\beta}$. The reduction has tightness gap $O(n^2\beta m)$.*

Proof. Recall that we have:

1. A reduction from $\text{IncGDD}^{\eta_\epsilon}_{\gamma,g}$ to $\text{SIS}_{n,m,q,\beta}$ for $g(n) > 0, m(n) = n^{O(1)}$, negligible function $\epsilon(n) = n^{-\omega(1)}$, $\beta(n) = n^{O(1)}$ and $q(n) \geq g(n)\sqrt{m(n)}n\beta(n)$ (Theorem 3.3), for $\gamma = \beta\sqrt{n}$.
2. A reduction from $\text{GIVP}^{\eta_\epsilon}_{8\gamma}$ to $\text{IncGDD}^{\eta_\epsilon}_{\gamma,8}$ (Theorem 3.7).
3. A reduction from $\text{SIVP}_{8\gamma\omega(\sqrt{\log n})}$ to $\text{GIVP}^{\eta_\epsilon}_{8\gamma}$ (Lemma 3.8).

Setting $g(n) = 8$, we can combine these into a reduction from SIVP_γ to $\text{SIS}_{n,m,q,\beta}$ for $\gamma = 8\beta\sqrt{n}\omega(\sqrt{\log n})$, whenever $q \geq 8n\sqrt{m}\beta$.

Consider the tightness gap of this proof. The IncGDD to SIS reduction makes one call to the SIS oracle and succeeds with probability $\geq \frac{1}{6\beta m}$, and the GIVP to IncGDD reduction makes $O(n^2)$ calls to the IncGDD oracle and succeeds with probability close to 1 (if the SIS instance has a solution). Thus, the total tightness gap is given by

$$t \cdot \frac{1}{\epsilon} = O(n^2)6\beta m = O(n^2\beta m).$$

□

3.4 Gentry, Peikert, Vaikuntanathan Reduction

Theorem 3.10 ([25, Proposition 5.2])**.** *Let $g > 1$ be a constant. Let $m = n^{O(1)}$ be an integer, $\beta = n^{O(1)}$, and $q \geq \beta\sqrt{n}\omega(\sqrt{\log n})$. Then there exists a reduction from worst-case SIVP_γ to average-case $\text{SIS}_{n,m,q,\beta}$ for $\gamma = g \cdot \beta\sqrt{n}\omega(\sqrt{\log n})$.*

The tightness gap of this reduction is $O(n^2\beta m)$.

Proof Sketch. This reduction follows the same template as the reduction from Theorem 3.3. We first present the high-level overview in Figure 3.3. We then give a proof sketch and an analysis of the tightness of the proof.

$$\mathrm{SIVP}_{g\cdot\beta\sqrt{n}\omega(\sqrt{\log n})} = \mathrm{GIVP}^{\lambda_n}_{g\cdot\beta\sqrt{n}\omega(\sqrt{\log n})} \to \mathrm{GIVP}^{\eta_\epsilon}_{g\cdot\beta\sqrt{n}} \to \mathrm{IncIVD}^{\eta_\epsilon}_{g\cdot\beta\sqrt{n},g} \to \mathrm{SIS}_{n,m,q,\beta}$$

Figure 3.3: SIVP_γ to SIS Reduction Overview (Gentry, Peikert & Vaikuntanathan)

The first part of the reduction is a reduction from IncIVD to SIS. IncIVD (see Definition 2.13) is an intermediate problem that's very similar to IncGDD, but finds a lattice vector slightly closer to the target point. This reduction is similar to that of Theorem 3.3, but instead of using a continuous Gaussian to sample uniformly amongst the cosets of $\frac{1}{q}\mathcal{L}(B)$ in $\mathcal{L}(B)$ to create the input matrix A to the SIS oracle, we use a discrete Gaussian to sample uniformly amongst the cosets of $\mathcal{L}(B)$ in $q\mathcal{L}(B)$[1].

This part succeeds with probability $\frac{1}{2\beta m}$. Indeed, as in Theorem 3.3, we choose an index $j \leftarrow [m]$ and $\alpha \leftarrow \{-\beta, \ldots, -1, 1, \ldots, \beta\}$ and succeed if the j^{th} component of the vector returned by the SIS oracle is equal to α (which happens with probability $\geq \frac{1}{2\beta m}$).

For the second part of the proof, we reduce $\mathrm{GIVP}^{\eta_\epsilon}_{g\gamma}$ to $\mathrm{IncIVD}_{\gamma,g}$, which requires $O(n^2)$ calls to the IncIVD oracle. The proof of this reduction is very similar to that of Theorem 3.7. Finally, we use Lemma 3.8 to reduce $\mathrm{SIVP}_{\gamma\omega(\sqrt{\log n})}$ to $\mathrm{GIVP}^{\eta_\epsilon}_{\gamma}$ (which is tight).

Chaining these reductions, we obtain a reduction from $\mathrm{SIVP}_{O(\beta\sqrt{n}\omega(\sqrt{\log n}))}$ to $\mathrm{SIS}_{n,m,q,\beta}$ with tightness gap $O(n^2\beta m)$. □

3.5 Micciancio and Peikert Reduction

Theorem 3.11 ([38, Theorem 1])**.** *Let n and $m = n^{O(1)}$ be positive integers, and let $\beta \geq \beta_\infty \geq 1$. Let q be a prime satisfying $q \geq \beta n^\delta$ for some constant $\delta > 0$. Let $\gamma = \max\{1, \beta\beta_\infty/q\} \cdot O(\beta\sqrt{n})\omega(\sqrt{\log n})$. Then there is a reduction from worst-case SIVP_γ to average-case $\mathrm{SIS}_{n,m,q,\beta,\beta_\infty}$*[2]*.*

The tightness gap of this reduction is n^2.

[1]The change from sampling from a continuous Gaussian to a discrete one is what allows us to find a closer lattice point to the target t using the SIS oracle (along with improved discrete Gaussian sampling techniques and bounds).

[2]$\mathrm{SIS}_{n,m,q,\beta,\beta_\infty}$ is the problem of finding a solution z to $\mathrm{SIS}_{n,m,q,\beta}$ that also satisfies $\|z\|_\infty \leq \beta_\infty$.

Proof Sketch. Again, this proof follows a similar approach to Theorems 3.9 and 3.10. The main difference is that the intermediate problem no longer resembles a closest vector problem (which was then used to iteratively find shorter and shorter sets of vectors), but is replaced with the problem of iteratively sampling from Gaussian distributions of decreasing widths. We first present the high-level overview in Figure 3.4[3]. We then give a proof sketch and an analysis of the tightness of the proof.

$$\text{SIVP}_{\gamma\omega(\sqrt{\log n})} = \text{GIVP}^{\lambda_n}_{\gamma\omega(\sqrt{\log n})} \to \text{GIVP}^{\eta_\epsilon}_{\gamma} \to \text{test } \eta_i \to \text{IncDGS}^{\eta_i}_{s_i} \to \text{SIS}_{n,m,q,\beta,\beta_\infty}$$

Figure 3.4: SIVP_γ to SIS Reduction Overview (Micciancio and Peikert)

The $\text{IncDGS}^{\eta_i}_{s_i}$ problem takes as input a set $\{(y_i, s_i) : y_i \sim D_{\Lambda,s_i}\}$ of discrete Gaussian samples (of varying widths), calls the SIS oracle on a matrix of coefficients $A = [a_1, \ldots, a_m]$ defined by $a_i = B^{-1}y_i \mod q$, and uses the output of the oracle to find a sample from a narrower discrete Gaussian distribution $D_{\Lambda,s}$, where $s < \max s_i$. If q is a prime (as is usually the case in our applications), then this part succeeds with high probability with a single call to the SIS oracle. Otherwise, it succeeds with probability $1/d_q$, where d_q denotes the number of proper divisors of q.

The function test η_i takes in a candidate upper bound η_i for the smoothing parameters, and recursively uses an $\text{IncDGS}^{\eta_i}_{s_i}$ oracle to find samples of width $\leq q\sqrt{2}\beta_\infty\eta_i$. This takes (up to) n calls to the $\text{IncDGS}^{\eta_i}_{s_i}$ oracle, and succeeds with high probability.

Finally, to solve $\text{GIVP}^{\eta_\epsilon}_{\gamma}$, the reduction calls the function test η_i with candidate upper bounds $\eta_i = 2^i, 1 \leq i \leq n$ for the smoothing parameter η_ϵ (note that we can guarantee that η_ϵ is in the range $[1, 2^n]$ by first LLL reducing the basis). Each call either fails, or returns a set of linearly independent vectors of length bounded by $\gamma/2\eta_i < \gamma\eta_\epsilon$ by a property of the smoothing parameter, thus solving $\text{GIVP}^{\eta_\epsilon}_{\gamma}$. This takes n calls to the test η_i oracle, and succeeds with high probability.

Finally, we can reduce $\text{SIVP}_{\gamma\omega(\sqrt{\log n})}$ to $\text{GIVP}^{\eta_\epsilon}_{\gamma}$ using Lemma 3.8 (which is tight).

Thus, we have a reduction from SIVP_γ to $\text{SIS}_{n,m,q,\beta,\beta_\infty}$, where

$$\gamma = \max(1, \beta\beta_\infty/q) \cdot O(\beta\sqrt{n})\omega(\sqrt{\log n}),$$

that makes n^2 calls to the SIS oracle and suceeds with high probability. □

[3]The intermediate problems were not named in [38]. We added names to better highlight the similarities and differences between this reduction and previous ones.

3.6 Tightness Discussion

We have seen three different reductions. Theorems 3.9 and 3.10 both achieve an approximation factor of $\gamma_0 = O(\beta\sqrt{n}\omega(\sqrt{\log n}))$ and have a tightness gap of $O(n^2\beta m)$. The only non-technical difference between the two results is the lower bound on the value of q (q can be smaller in the reduction from Theorem 3.10).

Theorem 3.11 further decreases the required bound on q and achieves an approximation factor $\gamma_1 \in [1, \beta]O(\beta\sqrt{n}\omega(\sqrt{\log n}))$[4], but has a smaller tightness gap of only n^2. If $\gamma_1 > \gamma_0$, then Theorem 3.11 reduces a strictly easier instance of SIVP_γ to $\mathrm{SIS}_{n,m,q,\beta}$ (for a fixed β). While this doesn't affect the tightness gap, it needs to be taken into account when analyzing the concrete security guarantees that the worst-case to average-case reductions grant to cryptographic constructions based on SIS. Methods for estimating the cost of solving SIS and SIVP problems will be presented in Chapter 4.

We will discuss the concrete security guarantees obtained from the worst-case to average-case reductions for two signature schemes based on the hardness of SIS in Chapters 5 and 6.

[4]We have that $\beta\beta_\infty/q < 1$ if $q \geq \beta\beta_\infty$, and $\beta\beta_\infty/q = \beta/n^\delta$ (which can be as big as β if $\delta \approx 0$) if $\beta = \beta_\infty$.

Chapter 4

Solving SIS and SIVP

This chapter discusses the fastest known algorithms for solving hard problems on lattices. In particular, we will present the best algorithms for solving the SIS and SIVP_γ problems, and methods for estimating the security level of a particular SIS or SIVP_γ instance.

The Shortest Integer Solution problem (Definition 2.14) is often found at the heart of lattice-based hash functions and signature schemes. By presenting the best algorithms for solving SIS, we describe the methods used for evaluating the security of these cryptographic constructions directly.

The Shortest Independent Vector problem (Definition 2.8) is a problem at the heart of many security proofs in lattice cryptography. However, not many people have analyzed the hardness of solving SIVP. Micciancio and Goldwasser [37] showed that $\text{SIVP}_{\gamma\sqrt{n}}$ polytime reduces to SVP_γ, thereby establishing that the asymptotic hardness of $\text{SIVP}_{\gamma\sqrt{n}}$ is not significantly more than that of SVP_γ. Note that for general lattices, no polynomial reduction from SVP_γ to $\text{SIVP}_{\gamma'}$ is known. Thus, it is possible that $\text{SIVP}_{\gamma\sqrt{n}}$ is significantly easier than SVP_γ.

Many lattice-based cryptographic constructions base their hardness on Ring or Module variants of SVP or SIVP. It is known that Ring-SVP is equivalent to Ring-SIVP, since if v is a short polynomial in $\mathbb{Z}[x]/(f)$, where $f(x)$ is an irreducible polynomial of degree n, then $v, vx, vx^2, \ldots, vx^{n-1}$ are linearly independent ring elements and have the same length as v. It is not known whether Module-SIVP is strictly harder than Ring-SIVP, or strictly easier than integer SIVP, so the best current algorithms are the same for SIVP and Module-SIVP.

Section 4.1 will describe the best algorithm paradigms for solving exact-SVP and the running times of state-of-the-art exact-SVP algorithms. Then, Section 4.2 will describe the

best algorithms for approximate SVP (SVP_γ). Finally, Section 4.3 will discuss methods for solving other lattice problems using an SVP_γ oracle, in particular showing how to solve SIVP_γ (Section 4.3.1), SIS (Section 4.3.2) and SIS and SVP_γ in different norms (Section 4.3.3).

4.1 Algorithms for Exact SVP

We will now describe the best algorithms for solving exact-SVP. Note that the Shortest Vector Problem is NP hard, so no efficient algorithm is expected to exist for solving it exactly.

There are two main categories of algorithms for SVP, *enumeration* and *sieving* algorithms. Both have exponential runtimes, with sieving algorithms being faster asymptotically. However, the constants in the exponent are such that enumeration algorithms are faster up to relatively large ($n \geq 70$) lattice instances. To further complicate the issue, experimental runtimes have been lower than the ones that have been proved, which means it's unclear at which point sieving becomes faster than enumeration. Lattices of the size used in cryptographic protocols currently fall into this grey zone, so we can only provide upper bounds on the runtimes of algorithms for SVP.

We will present a basic enumeration algorithm and a sieving algorithm, as well as improvements and best known runtimes for both categories of algorithms. Our description of the algorithms is primarily based on the survey work of Laarhoven, van de Pol and de Weger [30].

4.1.1 Enumeration

The main idea behind enumeration algorithms is to try all possible combinations of the basis vectors and see which one is the shortest. However, as there is an infinite number of such combinations in a lattice, it is necessary to find a good enumeration strategy and to bound the number of vectors we need to enumerate. Let R be an upper bound on the length of the short vector we are seeking. If we seek an exact solution to the SVP problem, then $R = \lambda_1$.

One way to bound this is through projective lattices. Let Λ be an n-dimensional lattice with basis $\{b_1, \ldots, b_n\}$. Let $\pi_k(u)$ denote the projection of the vector u onto the projected lattice, that is, Λ projected onto the orthogonal complement of the span of the vectors

$\langle b_1, \ldots, b_{k-1} \rangle$. If u is a vector of length at most R, then it clearly holds that

$$\|\pi_n(u)\|^2 \leq \|\pi_{n-1}(u)\|^2 \leq \cdots \leq \|\pi_1(u)\|^2 = \|u\|^2 \leq R^2.$$

By finding bounds on the length of $\|\pi_i(u)\|^2$ for the relevant i's, we can bound the number of u's that satisfy the inequalities, and recursively use short vectors in $\pi_i(\Lambda)$ to find short vectors in $\pi_{i-1}(\Lambda)$. We will not cover the derivation of the bounds, but suffice to say that the vectors we need to check are exponential in number.

One can think of enumeration algorithms as a search through a tree where each node at the i^{th} level corresponds to a short vector v in $\pi_{n-i+1}(\Lambda)$ and its children are short vectors u in $\pi_{n-i+2}(\Lambda)$ that project to v by π_{n-i+1}, that is $v = \pi_{n-i+1}(u)$.

The running times of such algorithms grow exponentially, but are guaranteed to find a vector of the desired length (if such a vector exists). Other enumeration algorithms make use of a tradeoff between probability of success and performance, pruning branches of the search tree when the probability of them containing the short vector is small and thus decreasing the size of the tree. There are also extreme pruning algorithms which prune large portions of the tree, but are fast enough that running them repeatedly makes the probability of success sufficiently large.

The fastest enumeration algorithm has a runtime of approximately $2^{0.187n \log n + 1.019n + 16.1}$ [6].

4.1.2 Sieving

An alternative to enumeration algorithms are the so-called sieving algorithms. The first sieving algorithm was described by Ajtai et al. [4]. Their algorithm started with a huge list of lattice vectors and iteratively reduced both the size of the list and the norms of the vectors in it. Later sieving algorithms such as the one of Micciancio and Voulgaris [40] use ideas from [4] to describe a way to build a list that hopes to exhaust the space of short lattice vectors. We will describe the main ideas behind the algorithm of [40].

The idea is as follows: Start with an empty list Q. Suppose we have an estimate $\mu \approx \lambda_1(\Lambda)$ (if the estimate is wrong, we can always increase or decrease μ and re-run the algorithm). Given the estimate μ, the algorithm samples short error vectors e_i, finds an associated lattice vector v_i, reduces the vector $r_i = v_i + e_i$ to a shorter vector $r_i' = v_i' + e_i$ using lattice vectors already in Q, and adds v_i' to the list Q. Thus, the algorithm builds a list Q of short lattice vectors (of norm at most $\|B\|$ and pairwise at least μ apart). The algorithm stops if two lattice vectors in Q are at most μ apart or we have used too many

samples. It suffices to show that at any point in time, it is likely that either a solution is found, or a new vector is added to the list. Showing that the list size is bounded by a finite constant concludes the proof, as it guarantees that we will escape the loop by finding a short vector with high probability.

To show that the size of the list Q is bounded by a constant (depending on the dimension n) one can note that the bounds on the length and pairwise distance of vectors in Q mean that there is a cone around each vector that contains no other vectors from Q. It is possible to show that each such cone has width at least $\pi/3$, so the size of the list Q is bounded by the number of cones of angle $\pi/3$ needed to cover an n-dimensional sphere. Details on this can be found in [40] or [30]. This gives us a lower bound of $2^{0.2075n+o(1)}$ on the space complexity of any sieving algorithms (as the list Q needs to be close to its maximal possible length before we are likely to find a short vector). Experimentally, this lower bound has proved to be pretty accurate for the space used by sieving algorithms. The runtime of sieving algorithms is approximately quadratic in the size of the list Q, or about $2^{0.415n+o(1)}$, as a sieving algorithms needs to find the shortest distance between vectors for all pairs of vectors in Q.

Sieving algorithms are asymptotically faster than enumeration algorithms with runtime $O(2^{cn})$, but still exponential and with much larger constants. Micciancio and Voulgaris' algorithm finds a solution in time at most $poly(n)2^{3.199n}$ and space at most $poly(n)2^{1.325n}$ with high probability. Currently, the best sieving algorithms are from [7], and have asymptotic runtime predictions of $2^{0.349n+o(n)}$. However, similarly to enumeration algorithms, the practical runtimes are often much smaller, taking $2^{cn+o(n)}$ with $c = 0.292$. A "paranoid" lower bound on the runtime of sieving algorithms is given by $2^{0.2075n}$, as any such algorithm must take at least this time to construct the list Q [7].

4.1.3 Sieving vs. Enumeration

It is not immediately obvious whether sieving or enumeration is the faster approach for a given lattice. Enumeration has smaller multiplicative constants but an extra log factor in the exponent, while sieving has larger multiplicative constants (but no $\log n$ factor in the exponents). Thus, for smaller instances, enumeration is better, but sieving is believed to be faster for lattices of dimension $n \geq 250$ (such as the ones in many concrete proposals for lattice-based post-quantum cryptosystems) [8]. Some better estimates have been obtained recently, and suggest the dimension at which sieving outperforms enumeration could be as low as $n = 70$ [7]. However, sieving and enumeration algorithms keep being improved, so it is possible this crossover point will change in either direction.

To test the exact and approximate-SVP algorithms, there are a number of lattice challenge problems at [1]. The largest instance of exact-SVP that has been solved to date is a lattice of dimension $n \approx 100$, by Albrecht et al. [7], using the sieving method[1].

4.2 Algorithms for Approximate-SVP

Algorithms for SVP_γ depend on the approximation factor we desire. If we want a solution for $\gamma = O(1)$ (which includes exact solutions), we need to use sieving or enumeration algorithms that have exponential running time. These were covered in Section 4.1. Algorithms for solving SVP_γ for larger γ's are generally so-called basis reduction algorithms, and produce a basis of reasonably short vectors (of which the shortest is a solution to SVP_γ).

Recall that the Gram-Schmidt Orthogonalization (GSO, see Definition 2.21) of a basis $B = \{b_1, \ldots, b_n\}$ of $\mathbb{R}^n$ is a basis $B^* = \{b_1^*, \ldots, b_n^*\}$ of vectors that are pairwise orthogonal and reasonably short. Lattice reduction algorithms are based on trying to mimic GSO for lattice bases.

4.2.1 LLL

The LLL algorithm [32] attempts to mimic GSO on an integer lattice Λ, by subtracting integer multiples (based on the Gram-Schmidt coefficients) of vectors $b_1^*, \ldots, b_{i-1}^*$ from b_i to obtain b_i^*. The LLL algorithm seeks to find vectors that are length reduced (i.e. it is impossible to find a shorter vector by subtracting integer multiples of other vectors in the basis) and such that the ratio between two GSO-vectors is sufficiently large (i.e. the lengths of the GSO vectors of the basis do not decrease too quickly).

We say that a basis B is LLL reduced if

1. For every $i < j$, we have $|\mu_{i,j}| \leq \frac{1}{2}$; and
2. For every $1 \leq i < n$, we have $\frac{\|b_i^*\|^2}{\|b_{i-1}^*\|^2} \geq \delta - \mu_{i,i-1}^2$ for $\delta \in (1/4, 1)$.

[1] There is no fast method to determine whether a short vector is indeed the shortest vector in a lattice. It is thus possible that algorithms for solving approximate-SVP in higher dimensions have in fact solved exact-SVP. The number presented is the dimension of the largest lattice on which Albrecht et al. [7] ran a sieving algorithm.

The LLL algorithm (Algorithm 1) returns an LLL-reduced basis. In the two-dimensional case, we can use Gauss's algorithm to return almost-optimal bases, that is bases $\{b_1^*, b_2^*\}$ such that the GSO-ratio satisfies

$$\frac{\|b_2^*\|^2}{\|b_1^*\|^2} \geq 1 - \mu_{2,1}^2 \geq \frac{3}{4}.$$

While achieving this ratio would be ideal in higher dimensions, no one has found an algorithm that achieves it with provably polynomial runtime, and so Lovász's condition represents a relaxation of this goal, replacing $1 - \mu_{i,i-1}^2$ with $\delta - \mu_{i,i-1}^2$ for $\delta \in (1/4, 1)$. The LLL algorithm seeks to find a basis that is locally (that is for each adjacent pair of vectors) Gauss reduced, by applying Gauss's algorithm to each pair of vectors b_i, b_{i-1} while ensuring the basis is size-reduced at all times.

We now present the LLL algorithm. For simplicity, we omit the updates to the Gram-Schmidt vectors, and instead assume that the Gram-Schmidt vectors are known and up-to-date with the current basis at all times. The LLL algorithm returns a basis B satisfying $\|B\| \leq 2^n \lambda_n(B)$, i.e. a basis with aproximation factor 2^n. The runtime is $O(n^6(\log \|B\|)^3)$ (it can be improved slightly via optimizations) [32].

Algorithm 1 LLL basis-reduction algorithm

Require: a basis $\{b_1, \ldots, b_n\}$ of L and a constant $\delta \in (\frac{1}{4}, 1)$.
Ensure: the output basis $\{b_1, \ldots, b_n\}$ of L is LLL reduced.

```
i ← 2
while i ≤ n do
    b_i ← b_i − Σ_{j=1}^{i−1} ⌊μ_{i,j}⌉ b_j
    if ‖b_i^*‖^2 ≥ (δ − μ_{i,i−1}^2)‖b_{i−1}^*‖^2 then
        i ← i + 1
    else
        swap(b_i, b_{i−1})
        i ← max{2, i − 1}
```

4.2.2 Korkine-Zoltarev and the BKZ Algorithm

Korkine-Zoltarev (KZ) reduction (Algorithm 2) is a different notion of reduction, which returns bases with small approximation and Hermite factors. The idea is to obtain a basis $\{b_1, \ldots, b_n\}$ such that the Gram-Schmidt vectors of the basis satisfy $\|b_i^*\| = \lambda_1(\pi_i(L))$, where $\pi_i(L)$ is the lattice projected onto the complement of the linear span $\langle b_1, \ldots, b_{i-1} \rangle$.

Given an SVP oracle that works for small (up to β dimensons), we can use it to find a KZ-reduced basis for a lattice of dimension β. The SVP subroutine can be either a sieving or an enumeration algorithm, generally chosen based on how the block size compares to the cutoff at which sieving becomes more efficient than enumeration, which implies the KZ basis-reduction algorithm has exponential runtime. We will describe this algorithm in more detail in Section 4.3.1.

Algorithm 2 KZ basis-reduction algorithm

Require: a basis $\{b_1, \ldots, b_\beta\}$ of L and an SVP oracle for up to β dimensions.

Ensure: the output basis $\{b_1, \ldots, b_\beta\}$ of L satisfies $\|b_i^*\| = \lambda_1(\pi_i(L))$ for each $i \in \{1, \ldots, \beta\}$.

1: **for** $i = 1$ to β **do**
2: call the SVP oracle to find $b_i^* \in \pi_i(L)$ of length $\lambda_1(\pi_i(L))$
3: lift b_i^* to a lattice vector b_i such that $\{b_1, \ldots, b_i\}$ is size reduced
4: update the vectors $\{b_{i+1}, \ldots, b_\beta\}$ by changing them to lattice vectors such that
5: $\{b_1, \ldots, b_\beta\}$ is a basis for L.

The block Korkine-Zolotarev (BKZ) algorithm (Algorithm 3) provides a way to harness the KZ notion of reduction, while keeping the runtime reasonable. The BKZ algorithm returns a basis that is locally KZ-reduced, that is, a basis in which each block of β consecutive vectors is KZ-reduced. The idea is similar to the LLL algorithm, but focuses on ensuring all β-dimensional sub-bases are optimal (in the Korkine-Zolotarev sense), instead of all 2-dimensional bases. If the blocksize is $\beta = 2$, then the BKZ algorithm is equivalent to the LLL algorithm.

Algorithm 3 BKZ basis-reduction algorithm

Require: a basis $\{b_1, \ldots, b_n\}$ of L, a constant $\delta \in (\frac{1}{4}, 1)$, and an SVP oracle for up to β dimensions.

Ensure: the output basis $\{b_1, \ldots, b_n\}$ of L is LLL-reduced with factor δ and satisfies $\|b_i^*\| = \lambda_1(\pi_i(b_i, \ldots, b_{i+\beta-1}))$ for each $1 \leq i \leq n - \beta + 1$.

1: $i \leftarrow 2$
2: **repeat**
3: **for** $i = 1$ to $n - \beta + 1$ **do**
4: KZ-reduce the basis $\pi_i(b_i, \ldots, b_{i+\beta-1})$
5: size-reduce the basis $\{b_1, \ldots, b_n\}$
6: **until** No changes occur.

The runtime of the BKZ algorithm has not been proven polynomial, but when the algorithm halts we know that the basis is good.

Specifically, we know that BKZ with blocksize β on a basis of dimension n achieves a provable root Hermite factor of

$$\delta_0 = \left(\sqrt{\gamma_\beta}^{1+\frac{n-1}{\beta-1}}\right)^{1/n}$$

where γ_β is the Hermite constant in dimension β. Recall that this means that the shortest vector b_1 in the basis satisfies $\|b_1\| \leq \delta_0^n \left(vol(\mathcal{L})\right)^{1/n}$. Exact values of γ_β are not known for $\beta \geq 24$, but can be bounded by $\gamma_\beta \leq \frac{2}{\pi}\left((1+\frac{\beta}{2})!\right)^{2/\beta}$ [30]. Figure 4.1 depicts this provable upper bound. Better upper bounds on γ_β can be computed numerically, although it doesn't seem to have been done for $n \geq 50$. A few examples of improved provable root Hermite bounds can be found in Table 4.1.

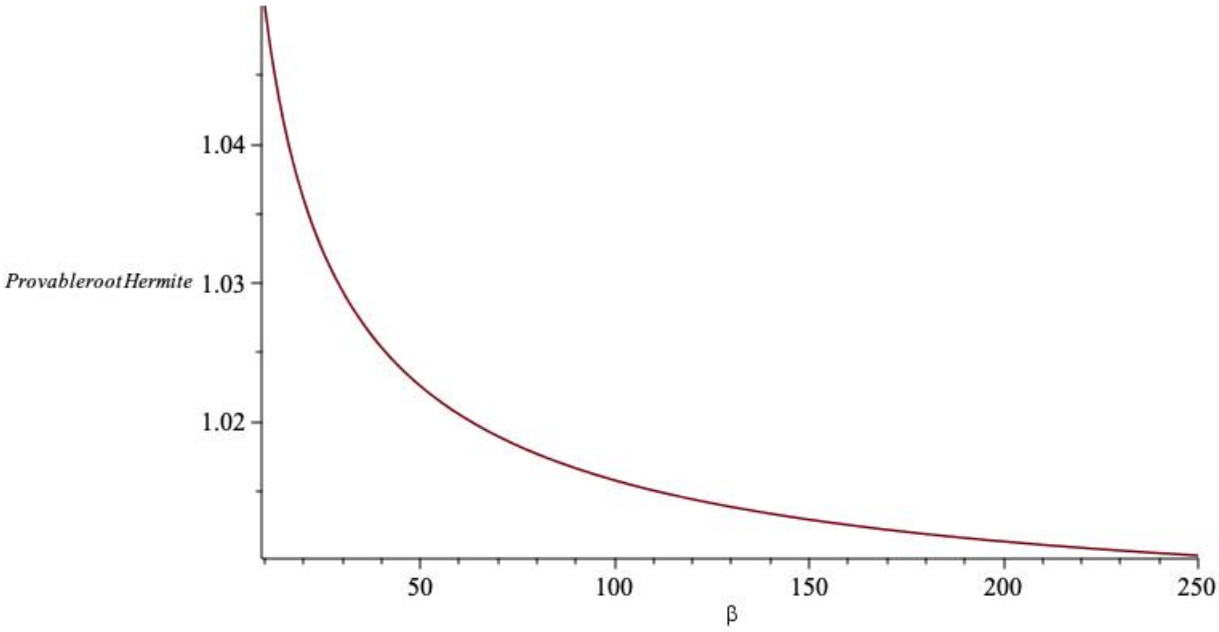

Figure 4.1: Provable BKZ root Hermite factor ($\delta_0(\beta)$) achieved by blocksize

As is often the case with lattice algorithms, experiments show that the root Hermite factors achieved in practice are often better (smaller) than the ones we can prove. For blocksizes $\beta \geq 50$ and $\beta \ll n$, the root Hermite factor achieved experimentally seems to

follows the asymptotic formula (see [7]):

$$\delta_0(\beta) = \left[(\beta\pi)^{1/\beta}\beta/(2\pi e)\right]^{1/2(\beta-1)}. \tag{4.1}$$

This seems to be the best tool for estimating the root Hermite factor BKZ achieves for blocksizes $\beta_{BKZ} \geq 50$. Figure 4.2 depicts the behaviour of the function $\delta_0(\beta)$.

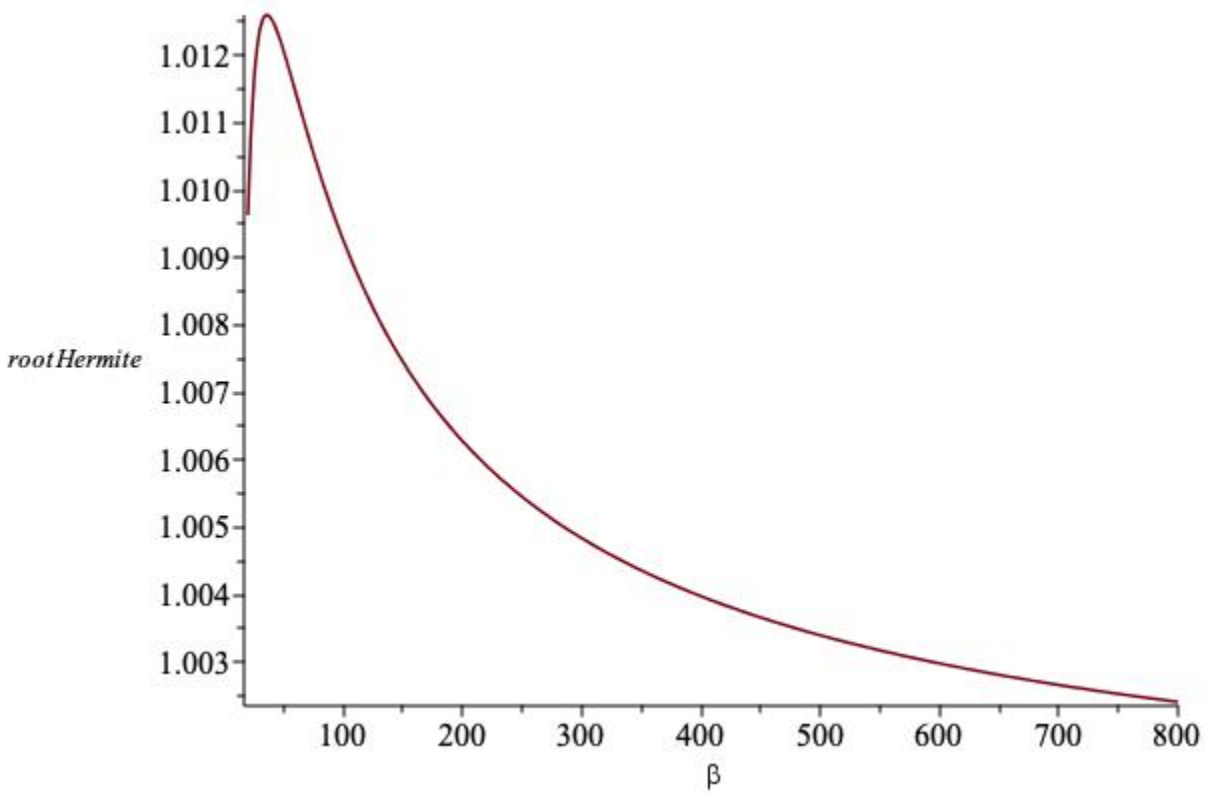

Figure 4.2: BKZ root Hermite factor ($\delta_0(\beta)$) achieved by blocksize

For smaller blocksizes, the achievable root Hermite factor is usually measured experimentally. We reproduce some results from [30, Table 1] in Table 4.1.

Various improvements to BKZ have been proposed since its introduction. The most influential one is BKZ 2.0 [21], which uses several optimizations and a technique known as pruned enumeration to increase the achievable blocksize β in BKZ and to minimize the number of calls to the exact-SVP oracle needed by BKZ. Many other improvements have been proposed since, most recently by Albrecht et al. [7].

Algorithm-β	LLL (BKZ-2)	BKZ-20	BKZ-28
Experimental root-Hermite factor δ_0	1.0219	1.0128	1.0109
Theoretical proven upper bound	1.0746	1.0337	1.0282

Table 4.1: BKZ root Hermite factor for small blocksizes.

Some work has also been done towards analyzing the quality of a basis output by so-called truncated BKZ, which halts the BKZ algorithm after some number of iterations instead of waiting until the basis stops changing. Experiments have shown that BKZ achieves good results in relatively few calls to the SVP oracle, and so ending the algorithm early can provide a worthwhile tradeoff between runtime and the quality of the output vectors.

4.2.3 Estimating Security Levels from BKZ

To estimate the cost of running BKZ, there are two main factors to consider. The first is the cost of the exact-SVP oracle, and the second is the number of calls to the SVP oracle used by BKZ. Albrecht et al. [6] provide a comprehensive list of the cost models chosen for the various lattice-based schemes in the NIST-PQC competition. The runtimes of enumeration and sieving algorithms were presented in Sections 4.1.1 and 4.1.2 respectively. When determining the number of SVP oracle calls required by BKZ, the most common choice was the "Core-SVP" model introduced in [42] which considers a single call to the SVP oracle. An alternate model estimates the number of calls to be $8n$, where n is the dimension of the starting lattice.

The Dilithium team [36] chose to consider an SVP oracle based on sieving with cost $2^{0.292\beta}$, and the core-SVP methodology of a single SVP call to estimate the cost of an attack on their signature scheme. We will be using the same cost model for our analysis in Chapters 5 and 6.

Thus, given a lattice problem that can be solved using lattice reduction, one can use the following steps to determine the cost of using BKZ to solve it. If this lattice problem is obtained from the security proof of a cryptosystem, one can then use the cost to determine the security level of the cryptosystem.

1. Based on the problem that you're trying to solve, determine the root Hermite factor needed to solve it. When solving SIS directly, the root Hermite factor is given as

$\left(\frac{\beta}{q^{n/m}}\right)^{1/m}$ (see Section 4.3.2). When solving SIVP_γ, the approximation factor γ corresponds to a root Hermite factor of $\left(\sqrt{\frac{\gamma}{\sqrt{n}}}\right)^{\frac{1}{n}}$ (see Section 4.3.1).

2. Use the asymptotic formula for the root Hermite factor (4.1) to find the BKZ blocksize β_{BKZ} needed to achieve the root Hermite factor from part 1.

3. Use your favorite BKZ cost model to estimate the runtime of BKZ with a blocksize of β_{BKZ}. Use this to determine the security parameter.

4.3 Solving Lattice Problems Using SVP

4.3.1 Solving SIVP

We will first present the polynomial-time reduction from $\text{SIVP}_{\gamma\sqrt{n}}$ to SVP_γ described in [37]. This is composed of a first reduction from KZP_γ to SVP_γ, followed by a reduction from $\text{SIVP}_{\gamma\sqrt{n}}$ to KZP_γ. We will then use this reduction to estimate the computational cost of solving an SIVP_γ instance.

Definition 4.1 (γ-approximate Korkine-Zoltarev problem (KZP_γ))**.** *Let $B = \{b_1, \ldots, b_n\}$ be a basis, and B^* be the corresponding Gram-Schmidt orthogonalized basis, defined by $b_1^* = b_1$ and $b_i^* = b_i - \text{proj}_i(b_i)$ for $i \geq 2$. The projection is defined by the formula $\text{proj}_i(b_i) = \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$ where $\mu_{i,j} = (b_i \cdot b_j^*)/\|b_j^*\|^2$ are the Gram-Schmidt coefficients.*

We call a basis B γ-approximate Korkine-Zoltarev (KZ_γ) reduced if

(i) $\|b_i^*\| \leq \gamma\lambda_i$ for each $1 \leq i \leq n$; and

(ii) For all $j < i$, the Gram-Schmidt coefficients $\mu_{i,j}$ of B satisfy $|\mu_{i,j}| \leq \frac{1}{2}$.

The γ-approximate Korkine-Zoltarev problem asks, given a basis B, to output a basis equivalent to B that is KZ_γ reduced.

Theorem 4.2. *There is a polynomial-time reduction from KZP_γ to SVP_γ*

Proof. Let Λ be a full rank basis. First, we call the SVP_γ oracle on Λ to get a vector b_1 such that $\|b_1\| \leq \gamma\lambda_1$. By definition of the Gram-Schmidt vectors, $\|b_1^*\| = \|b_1\| \leq \gamma\lambda_1$ as desired. Suppose we have found vectors $b_1, \ldots, b_{i-1}$, and want to find the vector b_i. Let

$\Lambda_i = \pi_i(\Lambda)$ be the projection of Λ onto the orthogonal complement of $b_1, \ldots, b_{i-1}$, and call the SVP_γ oracle to find a γ-approximate vector $\bar{b_i}$ in this lattice.

We now lift $\bar{b_i} \in \Lambda_i$ to a lattice vector $b_i \in \Lambda$ that satisfies $\pi_i(b_i) = \bar{b_i}$. Note that it is possible find such a vector b_i which is at distance at most $\pm\frac{1}{2}\|b_j^*\|$ from each of the hyperplanes spanned by vectors $[b_1, \ldots, b_{j-1}]$[2]. This implies that $|\mu_{i,j}| \leq \frac{1}{2}$ by definition, as $\mu_{i,j}$ measures the (normalized) distance between b_i and the hyperplane spanned by b_j^*.

Finally, we show that this b_i also satisfies $\|b_i^*\| \leq \gamma\lambda_i(\Lambda)$. Notice that $b_i^* = \pi_i(b_i)$ by definition. Then, since we found $\bar{b_i} = \pi_i(b_i)$ using an SVP_γ oracle in Λ_i, we have $\|\pi_i(b_i)\| \leq \gamma\lambda_1(\Lambda_i)$. We also know that Λ must contain at least i linearly independent vectors of length $\leq \lambda_i(\Lambda)$, so there must be a vector in Λ_i of length at most $\lambda_i(\Lambda)$. Thus, we have $\lambda_1(\Lambda_i) \leq \lambda_i(\Lambda)$ and so

$$\|b_i^*\| = \|\pi_i(b_i)\| \leq \gamma\lambda_1(\Lambda_i) \leq \gamma\lambda_i(\Lambda)$$

as desired. We repeat this for all i, and output the basis $[b_1, \ldots, b_n]$ which is KZ_γ reduced by definition.

Thus, we can find a solution to KZP_γ by making n calls to the SVP_γ oracle. □

Finally, we reduce $\text{SIVP}_{\gamma\sqrt{n}}$ to KZP_γ.

Theorem 4.3. *There is a polynomial-time reduction from* $\text{SIVP}_{\gamma\sqrt{n}}$ *to* KZP_γ.

Proof. We show that if B is KZ_γ reduced, then B is a solution to $\text{SIVP}_{\gamma\sqrt{n}}$ by showing that $\|b_i\| \leq \sqrt{n}\gamma\lambda_i$.

It is clear that a KZ_γ-reduced set is a basis (as each vector has a non-trivial portion that is orthogonal to the ones picked previously). We know by definition that $|\mu_{i,j}| \leq \frac{1}{2}$.

[2]This can done by an algorithm known as the Nearest Plane Algorithm, which finds a close lattice point to $\bar{b_i}$ by finding the closest point on each hyperplane spanned by the Gram-Schmidt vectors $[b_1^*, \ldots, b_j^*]$.

Thus, we have

$$\begin{aligned}
\|b_i\|^2 &= \|b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*\|^2 \\
&\leq \|b_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 \|b_j^*\|^2 \\
&\leq \gamma^2 \lambda_i^2 + \frac{1}{4} \sum_{j=1}^{i-1} \gamma^2 \lambda_j^2 \\
&\leq \gamma^2 \lambda_i^2 + \frac{i-1}{4} \gamma^2 \lambda_i^2 \\
&= \left(\frac{i+3}{4}\right) \gamma^2 \lambda_i^2.
\end{aligned}$$

We have $\|b_i\| \leq \sqrt{i}\gamma\lambda_i \leq \sqrt{n}\gamma\lambda_n$ as desired. Thus, $\text{SIVP}_{\gamma\sqrt{n}}$ reduces to KZP_γ (and SVP_γ).

Note that this reduction is tight, as a single call to the KZP_γ oracle returns a basis that is also a solution to $\text{SIVP}_{\gamma\sqrt{n}}$. □

Thus, if the running time of the SVP_γ oracle is $T(i)$ on a lattice of rank i, then the running time of the $\text{SIVP}_{\gamma\sqrt{n}}$ to SVP_γ reduction is $\sum_{i=1}^{n} T(i)$, which is $O(T(n))$[3], so the reduction is tight.

To estimate the cost of solving an SIVP_γ instance using BKZ, we have two options. First, we could directly study this cost. Unfortunately, there doesn't seem to be anything in the literature on the length of the vectors output by BKZ[4]. The second option is to note that SIVP_γ reduces to $\text{SVP}_{\gamma/\sqrt{n}}$, which in turn reduces to $\text{HSVP}_{\sqrt{\gamma/\sqrt{n}}}$. Thus, we can use an $\text{HSVP}_{\sqrt{\gamma/\sqrt{n}}}$ solver (i.e. one that achieves root Hermite factor $\left(\sqrt{\gamma/\sqrt{n}}\right)^{1/n}$) to solve SIVP_γ. This gives us an upper bound on the cost of solving SIVP_γ.

Finally, note that we are concerned with the cost of solving a worst-case SIVP_γ instance, as the reductions from Chapter 3 are all from worst-case SIVP_γ. If we pick blocksizes that provably achieve the root Hermite factor $\left(\sqrt{\gamma/\sqrt{n}}\right)^{1/n}$, then we will clearly be able to solve the worst-case instance of SIVP_γ. However, evidence suggests (see [30]) that the

[3]This holds as we can expect the cost of any SVP solver to be fully exponential.

[4]The Geometric Series Assumption lets us estimate the length of the Gram-Schmidt vectors of the output basis, but it is not clear how to relate those to the length of the basis vectors.

provable upper bound is only an upper bound for a worst-case basis given as input to the basis reduction algorithms, and that the true cost of solving a worst-case lattice problem is much lower, as we can pick one of many bases for the lattice in question.

For instance, in the case of LLL (or BKZ-2), there are input bases for which the output of the LLL algorithm satisfies the provable upper bound. However, the output of running LLL on a re-randomized basis of the same lattice again achieves the experimental root Hermite factor [30]. Similarly, one can show that there are bases that achieve the provable upper bound for KZ-reduction (although whether there are bases for which BKZ returns a basis that achieves the provable upper bound root Hermite factor is unknown), but these worst-case instances again depend on the shape of the basis, and not on the underlying lattice. Thus, as the worst-case root Hermite factor of lattice reduction algorithms depends on the shape of the input basis, it seems likely that the BKZ blocksize needed to solve a worst-case lattice problem (such as worst-case SIVP_γ) is much closer to the BKZ blocksize found experimentally than the BKZ blocksize obtained from the provable root Hermite factor formulas. As far as we can tell, there doesn't seem to be any literature analyzing the hardness of worst-case lattice problems.

Thus, we will use the experimental root Hermite factor formula (4.1) to estimate the BKZ blocksize needed to solve an SIVP_γ instance. However, as the provable upper bound on the BKZ root Hermite factor (see Figure 4.1) is the only bound on the hardness of worst-case SIVP_γ, we will also provide estimates of the provable hardness of SIVP_γ to the reader (although the reader should keep in mind that these figures are likely to be a significant overestimate of the hardness of worst-case SIVP_γ).

4.3.2 Solving SIS

Note that SIS is very similar to SVP_γ, as we're looking for a short vector in an integer lattice. There are a couple of differences between SIS and SVP. The first is that the bound on the length of the short vector is given in absolute terms, and not as a multiple of the length of the shortest vector.

The second is that an SIS lattice is a q-ary lattice, as it is defined by a matrix in $\mathbb{Z}_q^{n\times m}$. This means that there could be an algorithm that solves an instance of SIS faster than an SVP instance of the same dimension. No such algorithm is currently known, so the best algorithm known for solving SVP (the BKZ algorithm, see Section 4.2.2) is also the best algorithm for solving SIS.

However, the fact that an SIS lattice $\Lambda \subseteq \mathbb{R}^m$ is q-ary implies that its volume is equal to q^n with high probability. This in turn implies that the root Hermite factor needed to

solve an $\text{SIS}_{n,m,q,\beta}$ instance is

$$\left(\frac{\|v\|}{vol(\Lambda)^{1/m}}\right)^{1/m} \leq \left(\frac{\beta}{q^{n/m}}\right)^{1/m}.$$

Thus, we can use the methodology of Section 4.2.3 with the root Hermite factor $\left(\frac{\beta}{q^{n/m}}\right)^{1/m}$ to estimate the cost of solving an SIS instance.

Note that it is possible to do slightly better by ignoring some of the columns of the SIS matrix (which is over-determined). A more thorough analysis of the root Hermite factor needed to solve an $\text{SIS}_{n,m,q,\beta}$ is presented in [15].

4.3.3 Lattice Problems in Other Norms

Lattice problems are most commonly defined with respect to the ℓ_2 norm. Other metrics, for example the ℓ_∞ metric, are more intuitive when defininining cryptosystems based on lattice problems. In particular, the signature scheme Dilithium [36] is based on the hardness of the $\ell_\infty-$SIS problem.

There has not been a lot of work done to evaluate the hardness of lattice problems in different norms. Since most lattice algorithms have been studied in the ℓ_2 norm, a reduction from a problem in the ℓ_2 norm to a problem in the ℓ_p norm would allow us to make statements about the hardness of the ℓ_p problem (as such a reduction would imply that the ℓ_p problem is at least as hard as the ℓ_2 problem), and thus about the security of a cryptosystem based on the hardness of that problem in the ℓ_p norm. On the other hand, a reduction from an ℓ_p problem to an ℓ_2 problem implies that there is an algorithm for solving the ℓ_p problem using an oracle for solving the ℓ_2 problem (possibly a polynomial number of times), thus giving an upper bound on the hardness of the ℓ_p problem. We will present known reductions to problems in the ℓ_∞ norm, and discuss what this means about the hardness of ℓ_∞ problems.

There is an immediate reduction from $\ell_2-\text{SVP}_{\sqrt{n}\gamma}$ to $\ell_\infty-\text{SVP}_\gamma$, as for any vector $x \in \mathbb{R}^n$ we have that $\|x\|_\infty \leq c$ implies $\|x\|_2 \leq \sqrt{n}c$. This shows that the cost of solving an $\ell_\infty-\text{SVP}_\gamma$ instance is at least that of solving an $\ell_2-\text{SVP}_{\gamma\sqrt{n}}$ instance. We also have a reduction from $\ell_\infty-\text{SVP}_\gamma$ to $\ell_2-\text{SVP}_\gamma$, as $\|x\|_2 \leq c$ implies $\|x\|_\infty \leq c$. This shows that the cost of solving an $\ell_\infty-\text{SVP}_\gamma$ instance is at most that of solving an $\ell_2-\text{SVP}_\gamma$ instance.

Note that we can use the same argument to get reductions from $\ell_2-\text{SIS}_{n,m,q,\sqrt{m}\beta}$ to $\ell_\infty-\text{SIS}_{n,m,q,\beta}$ and from $\ell_\infty-\text{SIS}_{n,m,q,\beta}$ to $\ell_2-\text{SIS}_{n,m,q,\beta}$ which gives us upper and lower bounds on the hardness of $\ell_\infty-$SIS instances.

Similar reductions exist for other ℓ_p norms, but they are less commonly used in applications so we will not present them here. If we want a better approximation factor, Regev and Rosen presented reductions from $\ell_2-\text{SVP}_\gamma$ to $\ell_p-\text{SVP}_{\gamma'}$ for all p and closer approximation factors γ, γ' [46]. Since we will need to estimate the hardness of lattice problems in the ℓ_∞ norm, we will present the result about the $\ell_2-\text{SVP}$ to $\ell_\infty-\text{SVP}$ reduction by Regev and Rosen.

Theorem 4.4. *[46, Theorem 1.2] For any $\gamma \geq 1$, there exists a randomized polynomial time dimension-preserving reduction from solving $\text{SVP}_{O(\sqrt{\log n}\gamma)}$ in the ℓ_2 norm to solving SVP_γ in the ℓ_∞ norm. This reduction makes one call to the ℓ_∞ oracle and succeeds with probability $\geq 1 - \frac{1}{n}$.*

The additional structure of an SIS instance causes Regev and Rosen's reduction to fail between SIS instances, so it is not clear whether similar results can be shown for SIS.

Note that Micciancio and Peikert's SIVP to SIS reduction (see Theorem 3.11) applies to $\ell_\infty-\text{SIS}$ instances, as the above discussion implies we can use $\beta = \sqrt{m}\beta_\infty$ in Theorem 3.11 to establish a reduction from SIVP_γ to $\ell_\infty-\text{SIS}$. However, as we will see in Chapter 6, this reduction is not generally used for setting parameters and determining security of $\ell_\infty-\text{SIS}$ instances (and is perhaps not the best tool to do so).

Finally, note that these reductions are only useful if they reduce to well-defined $\ell_2-\text{SIS}$ instances. In particular, note that it is trivial to find a vector of ℓ_2-norm q in an SIS lattice, as the vector $(q, 0, \ldots, 0)$ has this length and is a solution to the equation $[I|A] \cdot z \equiv 0 \mod q$. This can be a problem when looking for reductions involving the SIS problem in ℓ_∞ norm as they might reduce from $\ell_2-\text{SIS}$ problems of norm $\geq q$ (which is trivial, and thus doesn't give us any information about the $\ell_\infty-\text{SIS}$ instance).

We will use these tools in Chapter 6 to estimate the security level of Dilithium, which is based on an $\ell_\infty-\text{SIS}$ instance.

Chapter 5

Lyubashevsky Signatures

Lyubashevsky [35] presented a signature scheme based on the hardness of the $\text{SIS}_{n,m,q,\beta}$ problem. The main innovation introduced in the paper was the concept of rejection sampling, which is a method of ensuring that the distribution of signatures is independent of the secret key. This method has since been used in many lattice-based signature schemes, including multiple NIST submissions, and so the signature scheme presented by Lyubashevsky can be seen as an early precursor to some of these schemes. Lyubashevsky provided concrete parameters for the signature scheme, but the relatively large signature sizes (when compared to the state-of-the art lattice signature schemes that exist today) make this signature scheme impractical.

In the paper, Lyubashevsky stated that the worst-case to average-case proof gives confidence in the security of the signature scheme:

> On the theoretical side [...] the scheme constructed in this paper [... is] based on the hardness of finding a vector of length $\widetilde{O}(n)$ in SIS instances, which by the worst-case to average-case reduction of Micciancio and Regev is as hard as solving approximate SIVP with a factor of $\widetilde{O}(n^{1.5})$ in all n-dimensional lattices.

However, he didn't set parameters based on the reduction itself, as he was concerned by the security implications of the difference in dimensions between the SIVP_γ instance ($dim = n$) and the SIS one (between n and m, where $m \geq n$). All of the comments Lyubashevsky made in his paper about the security implications of the reduction from SIVP_γ are informal.

In this chapter, we will determine the security guarantees provided by the worst-case to average-case reduction, taking into consideration the tightness of the reduction and the relative hardness of the SIS problem and the corresponding SIVP_γ problem.

5.1 Signature Scheme

We will start by presenting Lyubashevsky's signature scheme. Let $D_H = \{v : v \in \{-1, 0, 1\}^k, \|v\|_1 \leq \kappa\}$ and let $H : \{0,1\}^* \to D_H$ be a cryptographic hash function. Also recall from Chapter 2 that $D_{\mathbb{Z}^m,c,\sigma} = D^m_{\mathbb{Z},c,\sigma}$ denotes the discrete Gaussian distribution on the integer lattice $\mathbb{Z}^m$, centered at c (if $c = 0$, we simply denote this distribution $D^m_{\mathbb{Z},\sigma}$).

Algorithm 4 Key Generation

Output: Public key $pk = (A, T)$ and secret key $sk = S$.

1: $S \leftarrow \{-d, \ldots, 0, \ldots, d\}^{m \times k}$
2: $A \leftarrow \mathbb{Z}_q^{n \times m}$
3: $T \leftarrow AS$
4: Return $pk = (A, T)$ and secret key $sk = S$.

Algorithm 5 Signature Generation

Input: Secret key $sk = S$ and message $\mu \in \{0,1\}^*$.
Output: Signature (z, c) on μ.

1: $y \leftarrow D^m_{\mathbb{Z},\sigma}$
2: $c \leftarrow H(Ay, \mu)$
3: $z \leftarrow Sc + y$
4: Output (z, c) with probability $\min\left(\frac{D^m_{\mathbb{Z},\sigma}(z)}{MD^m_{Sc,\sigma}(z)}, 1\right)$

Algorithm 6 Signature Verification

Input: $pk = (A, T)$ and signed message (μ, z, c).
Output: "Accept" or "Reject".

1: Return "Accept" if $\|z\| \leq 2\sigma\sqrt{m}$ and $c = H(Az - Tc, \mu)$, "Reject" otherwise.

The main idea of the signature scheme is that the output distribution of z should be statistically close to $D^m_{\mathbb{Z},\sigma}$. However, since $z = Sc + y$, z is actually distributed according to $D^m_{\mathbb{Z},Sc,\sigma}$. To modify the output distribution, the signing algorithm only outputs a signature (z, c) if z is also distributed according to $D^m_{\mathbb{Z},\sigma}$. This is formalized in Lemma 5.1.

Sample parameter sets for the Lyubashevsky signature scheme can be found in Table 5.1. All the parameter choices are for the same security level; They serve to demonstrate different trade-offs between key and signature size. We will discuss the security level of the Lyubashevsky signature scheme in Section 5.2.

Table 5.1: Parameter options for Lyubashevsky's signature scheme.

	I	II	III
n	512	512	512
q	2^{27}	2^{25}	2^{33}
d	1	1	31
k	80	512	512
$m \approx 64 + n \cdot \log q / \log(2d+1)$	8786	8139	3253
κ	28	14	14
$\sigma \approx 12 \cdot d \cdot \kappa \cdot \sqrt{m}$	31496	15157	300926
M	2.72	2.72	2.72
Secret key size (bits)	2^{20}	$2^{22.5}$	2^{23}
Public key size (bits)	2^{20}	$2^{22.5}$	2^{23}
Signature size (bits)	163000	142300	73000

5.1.1 Correctness and Security Proofs

We will present sketches of the proofs of correctness and security. In order to prove these, we need the *"rejection sampling"* Lemma from [35].

Lemma 5.1. *Let V be a subset of $\mathbb{Z}^m$ in which all elements have norm less than τ, σ be some element in $\mathbb{R}$ such that $\sigma = \omega(\tau\sqrt{\log m})$, and $h : V \to \mathbb{R}$ be a probability distribution. Then there exists a constant $M = O(1)$ such that the distribution of the algorithm A:*

1. *$v \leftarrow h$*
2. *$z \leftarrow D^m_{\mathbb{Z},v,\sigma}$*
3. *Output (z, v) with probability $\min\left(\frac{D^m_{\mathbb{Z},\sigma}(z)}{MD^m_{\mathbb{Z},v,\sigma}(z)}, 1\right)$*

is within statistical distance $\frac{2^{-\omega(\log m)}}{M}$ of the distribution of the algorithm F:

1. *$v \leftarrow h$*
2. *$z \leftarrow D^m_{\mathbb{Z},\sigma}$*
3. *Output (z, v) with probability $\frac{1}{M}$.*

Moreoever, A outputs something with probability $\geq \frac{1-2^{\omega(\log m)}}{M}$.

In particular, if $\sigma = \alpha\tau$ for some $\alpha > 0$, then $M = e^{12/\alpha+1/(2\alpha^2)}$, the output of algorithm A is within statistical distance $\frac{2^{-100}}{M}$ of the output of F, and A outputs something with probability at least $\frac{1-2^{-100}}{M}$.

The parameters for Lyubashevsky's signatures scheme are selected so that they satisfy the conditions of Lemma 5.1.

To show correctness, note that if $\|z\| \leq 2\sigma\sqrt{m}$, the signature verification procedure will always accept an honestly generated signature, since

$$Az - Tc = A(Sc + y) - (AS)c = Ay.$$

The probability that the verifier accepts a signature created by an honest signer is thus the probability that $\|z\| \leq 2\sigma\sqrt{m}$. Since the rejection sampling lemma tells us the output z's are distributed according to a distribution statistically close to $D^m_{\mathbb{Z},\sigma}$, we can use the probability that an element sampled from $D^m_{\mathbb{Z},\sigma}$ has length at most $2\sigma\sqrt{m}$. By properties of the discrete Gaussian distribution, we know that this will happen with probability $\geq 1 - 2^m e^{m/2(1-2^2)} \geq 1 - 2^{-100}$.

To prove security, Lyubashevsky reduces solving the Short Integer Solution problem to forging a signature. We will present the relevant Theorem and a sketch of the proof.

Theorem 5.2. *[35, Theorem 4.1] If there is a (strong) polynomial-time forger, who makes at most s queries to the signing oracle and h queries to the random oracle H, and who breaks the Lyubashevsky signature scheme with probability δ, then there is a polynomial time algorithm that can solve the ℓ_2−SIS$_{n,m,q,\beta}$ problem for $\beta = (4\sigma + 2d\kappa)\sqrt{m} = \widetilde{O}(dn)$ with probability $\approx \frac{\delta^2}{2(h+s)}$.*

Proof. We prove the theorem as a sequence of games.

Game 0 is the signature scheme as defined above.

Game 1 (Algorithm 7) is the same signature scheme, but we generate c uniformly at random from the range of H (and update the random oracle with the assigned hash values).

We define Game 2 (Algorithm 8) as Game 1, but replacing $z = Sc + y$ with z generated according to $D^m_{\mathbb{Z},\sigma}$. This makes signatures independent of S.

We can show that the statistical distance between Game 0 and Game 2 is at most $\epsilon = s(h + s) \cdot 2^{-n+1} + s \cdot \frac{2^{-100}}{M}$. Indeed, the only difference between Game 0 and Game 1

Algorithm 7 Game 1

Input: Secret key $sk = S$ and message μ.
Output: Signature (z, c) on μ.

$y \leftarrow D^m_{\mathbb{Z},\sigma}$
$c \leftarrow \{v : v \in \{-1, 0, 1\}^k, \|v\|_1 \leq \kappa\}$
$z \leftarrow Sc + y$
With probability $\min\left(\frac{D^m_{\mathbb{Z},\sigma}(z)}{MD^m_{\mathbb{Z},Sc,\sigma}(z)}, 1\right)$
    Output (z, c)
    Program $H(Az - Tc, \mu) = c$

Algorithm 8 Game 2

Input: Secret key $sk = S$ and message μ.
Output: Signature (z, c) on μ.

$y \leftarrow D^m_{\mathbb{Z},\sigma}$
$c \leftarrow \{v : v \in \{-1, 0, 1\}^k, \|v\|_1 \leq \kappa\}$
$z \leftarrow D^m_{\mathbb{Z},\sigma}$
With probability $1/M$
    Output (z, c)
    Program $H(Az - Tc, \mu) = c$

is that the output of the random oracle is chosen randomly in Game 1, and the random oracle is programmed without checking if that value had been assigned any other hash previously. Thus, the advantage of the distinguisher is given by the probability that there is a collision amongst the $h + s$ values ever assigned to the random oracle. A bit of careful consideration gives us a probability of $s(s + h)2^{-n+1}$.

We then use Lemma 5.1 to show that the statistical distance between Game 1 and Game 2 is at most $s \cdot \frac{2^{-100}}{M}$. This shows that the total statistical distance between Game 0 and Game 2 is at most $\epsilon = s(h + s)2^{-n+1} + s\frac{2^{-100}}{M}$. Thus, Game 0 and Game 2 are ϵ-close statistically, and we can replace the signature scheme with Game 2 to prove security.

To conclude the proof, we show that a forger that is successful against Game 2 lets us solve SIS for the given β. Note that with very high probability (there's only a $1/|D_H|$ chance of guessing a c such that $c = H(Az - Tc, \mu)$, where $(\mu, (z, c))$ is the forged signed message produced by the forger) the random oracle must have produced the value c in response to some query. This query could have been made during a signing query, or for a query directly to the random oracle H.

In the first case, suppose the signer programmed the random oracle $H(Az' - Tc, \mu') = c$ when signing a message μ'. If the forger outputs a valid forgery (z, c) for some message μ, then we have $H(Az' - Tc, \mu') = c = H(Az - Tc, \mu)$ by correctness. Since $(\mu, (z, c))$ is a forgery, we must have $(\mu', z') \neq (\mu, z)$. If $\mu \neq \mu'$ or $Az - Tc \neq Az' - Tc$, then the forger has found a collision for H (which we assume is infeasible by the definition of the hash function H). Thus, we have $\mu = \mu'$ and $Az - Tc = Az' - Tc$. Hence, $A(z - z') = 0$, and $z - z' \neq 0$ (since the forged signature must be different from the queried signature). Finally, we have that $\|z\|, \|z'\| \leq 2\sigma\sqrt{m}$ (since the verifier accepts both signatures), and hence $\|z - z'\| \leq 4\sigma\sqrt{m}$, as desired.

In the second case, suppose c was the response to a query the forger made directly to the random oracle. We begin by recording the signature (z, c) of the forger on the message μ, and then use the Forking Lemma (Lemma 2.22) to rewind the forger to obtain a second valid signature (z', c') on the message μ. Suppose that the random oracle query that returned c and c' was a string s (it must be the same string, by the proof of the Forking Lemma). Then by correctness of signatures (z, c) and (z', c'), we must have $Az - Tc = s = Az' - Tc'$ and so (using the fact $T = AS$), we have

$$A(z - z' + Sc - Sc') = 0.$$

Since $\|z\|, \|z'\| \leq 2\sigma\sqrt{m}$ and $\|Sc\|, \|Sc'\| \leq d\kappa\sqrt{m}$, we have

$$\|z - z' + Sc' - Sc\| \leq (4\sigma + 2d\kappa)\sqrt{m}$$

as desired.

Finally, we must show that $z - z' + Sc' - Sc \neq 0$. To show this, note that there is a high chance that there exists another key S' such that $AS = AS'$, and if $z - z' + Sc' - Sc = 0$ then $z - z' + S'c' - S'c \neq 0$. Since the signature algorithm in Game 2 is independent of the signing key S, the adversary has no way of knowing which of S, S' was used. Thus, with probability at least $\frac{1}{2}$, we will get $z - z' + Sc' - Sc \neq 0$.

The probability of success of this reduction is given by $\frac{1}{2}$ times the probability of success of the Forking Lemma, or

$$\frac{\delta^2}{2(h+s)},$$

as desired. □

5.1.2 Reducing to SIVP

Lyubashevsky notes that we could reduce the signature scheme further to SIVP_γ, but doesn't do so because he believes it is harder to solve the SIS instance than the associated

SIVP_γ instance. We present a proof of this reduction here.

Theorem 5.3. *Let $\mathcal{F}$ be a polynomial-time forger who makes at most s queries to the signing oracle and h queries to the random oracle H, and breaks the signature scheme with probability δ. Let $\beta = (4\sigma + 2d\kappa)\sqrt{m}$, with parameters as defined above.*

(i) If $q \geq 8n\sqrt{m}\beta$, then there is an algorithm that can solve the SIVP_γ problem for

$$\gamma = 8\beta\sqrt{n}\omega(\sqrt{\log n})$$

in time $O(n^2) \times \text{Time}(\mathcal{F})$ and with probability

$$\epsilon = O\left(\frac{1}{\beta m}\right)\frac{\delta^2}{2(h+s)}.$$

(ii) If $q \geq g\beta\sqrt{n}\omega(\sqrt{\log n})$, then there is an algorithm that can solve the SIVP_γ problem for

$$\gamma = g\beta\sqrt{n}\omega(\sqrt{\log n})$$

for any constant $g > 1$ in time $O(n^2) \times \text{Time}(\mathcal{F})$ and with probability

$$\epsilon = O\left(\frac{1}{\beta m}\right)\frac{\delta^2}{2(h+s)}.$$

(iii) If $q \geq \beta n^c$ for some constant $c > 0$, then there is an algorithm that can solve the SIVP_γ problem for

$$\gamma = \frac{\beta}{n^c} \cdot \beta\sqrt{n}\omega(\sqrt{\log n})$$

in time $n^2 \times \text{Time}(\mathcal{F})$ and with probability

$$\epsilon = \frac{\delta^2}{2(h+s)}.$$

Proof. From Theorem 5.2, we know that solving $\text{SIS}_{n,m,q,\beta}$ for $\beta = (4\sigma + 2d\kappa)\sqrt{m}$ with probability $\approx \frac{\delta^2}{2(h+s)}$ reduces to breaking the signature scheme, with at most two calls to the forging oracle.

Recall from Theorem 3.9 that if $q \geq 8n\sqrt{m}\beta$, we can reduce SIVP_γ to $\text{SIS}_{n,m,q,\beta}$ for

$$\gamma = 8\beta\sqrt{n}\omega(\sqrt{\log n}) = 8(4\sigma + 2d\kappa)\sqrt{m}\sqrt{n}\omega(\sqrt{\log n}) = 8(4\sigma + 2d\kappa)\sqrt{mn}\omega(\sqrt{\log n})$$

with probability $\frac{1}{6\beta m}$. Combining these proves the first part of the Theorem.

The second and third parts are similarly derived by combining Theorems 3.10 and 3.11 with Theorem 5.2. □

From Table 5.1, we can compute the value of $\beta = (4\sigma + 2d\kappa)\sqrt{m}$ for parameter sets I, II, and III. We can then use this β to compare the value of q in the parameter sets against the minimum value of q required by parts (i), (ii), and (iii) of Theorem 5.3. These computations are summarized in Table 5.2.

Remark. We use $\omega(\sqrt{\log n}) = \sqrt{\log n}$, $n^c = 1$, and $g = 1$ in the computations in Table 5.2, as they are tight lower bounds for these values. We also ignore the constants from $O(\cdot)$ notation.

Table 5.2: Lower bounds on q for Lyubashevsky's signature scheme (cf. Theorem 5.3).

	I	II	III
$\beta = (4\sigma + 2d\kappa)\sqrt{m}$	2^{23}	2^{22}	2^{26}
q	2^{27}	2^{25}	2^{33}
$8\beta n\sqrt{m}$ (Theorem 3.9)	2^{42}	2^{40}	2^{43}
$g\beta\sqrt{n}\omega(\sqrt{\log n})$ (Theorem 3.10)	2^{31}	2^{30}	2^{33}
βn^c (Theorem 3.11)	2^{23}	2^{22}	2^{26}

Note that none of the parameter sets satisfy the bound $q \geq 8\beta n\sqrt{m}$ of Theorem 3.9 (all values of q are at least a factor of 2^{10} too small). They also do not satisfy the bound $q \geq \beta\sqrt{n}\omega(\sqrt{\log n})$ of Theorem 3.10. However, they are a relatively small factor (between g and $2^5 g$, where g can be as small as 4) from satisfying the bound, so the parameters could easily be modified to satisfy the conditions of Theorem 3.10. They also all satisfy the conditions of Theorem 3.11.

Thus, by Theorem 5.3, we can either reduce SIVP_{γ_1} (for $\gamma_1 = g\beta\sqrt{n}\omega(\sqrt{\log n})$) to breaking Lyubashevsky's signature scheme with tightness gap

$$g_1 = T\frac{1}{\epsilon} = O\left(n^2\beta m\right) \cdot \frac{2(h+s)}{\delta},$$

or reduce SIVP_{γ_2} (for $\gamma_2 = \frac{\beta}{n^c} \cdot \beta\sqrt{n}\omega(\sqrt{\log n})$) to breaking Lyubashevsky's signature scheme with tightness gap

$$g_2 = T\frac{1}{\epsilon} = n^2 \cdot \frac{2(h+s)}{\delta}.$$

5.2 Concrete Security Guarantees

In this section, we will determine concrete security guarantees for Lyubashevsky's signature scheme. We will first present the analysis that Lyubashevsky used to determine security, and then compare this value to the bits of security that the signature scheme gets from the worst-case SIVP_γ problem. We will present this analysis for parameter set I, and list the numbers for parameter sets II and III in Table 5.3.

5.2.1 Security Guarantees from SIS

To set parameters, Lyubashevsky assumed that achieving a root Hermite factor of $\delta_0 = 1.007$ would be out of reach of lattice reduction algorithms for the foreseeable future. He then picked parameters such that the root Hermite factor needed (which is $\approx \left(\frac{\beta}{q^{n/m}}\right)^{1/m}$) to find a vector of length $\beta = (4\sigma + 2d\kappa)\sqrt{m}$ is smaller than δ_0.

Lyubashevsky did not convert this root-Hermite factor to some number of bits of security[1]. Using the heuristic

$$\delta(\beta_{BKZ}) = \left[(\beta_{BKZ}\pi)^{1/\beta_{BKZ}}\beta_{BKZ}/(2\pi e)\right]^{1/2(\beta_{BKZ}-1)}$$

from (4.1), a root-Hermite factor of 1.007 could be achieved using a BKZ blocksize of $\beta_{BKZ} = 175$. The cost of running a sieving algorithm in dimension β_{BKZ} is given by $2^{0.292\beta_{BKZ}} \approx 2^{51}$, which means the signature scheme has approximately 51 bits of security with the recommended parameters (for each of the three parameter sets).

5.2.2 Security Guarantees from SIVP

We will now determine the security guarantees that Lyubashevsky's signature scheme gets from the worst-case SIVP_γ instances.

Recall from Theorem 5.3 that the tightness gap of the reduction from SIVP_γ is given by either

$$g_1 = O(n^2) \cdot \frac{2(h+s)}{\delta} \cdot 2\beta m$$

[1]The heuristics we use to estimate bits of security were presented after Lyubashevsky's paper was published, and our knowledge of lattice reduction algorithms has also improved since then.

or

$$g_2 = n^2 \cdot \frac{2(h+s)}{\delta},$$

where $\beta = (4\sigma + 2d\kappa)\sqrt{m}$. Substituting the values of n, m, β for parameter set I, we get that the tightness gap is

$$g_1 = 2^{55}\frac{2(h+s)}{\delta}$$

or

$$g_2 = 2^{18}\frac{2(h+s)}{\delta}.$$

We can also use the methodology from Section 4.3.1 to estimate the hardness of SIVP_γ for

$$\gamma_1 = g\beta\sqrt{n}\omega(\sqrt{\log n})$$

and

$$\gamma_2 = \frac{\beta}{n^c} \cdot \beta\sqrt{n}\omega(\sqrt{\log n}).$$

We get the corresponding root Hermite factors

$$\delta_1 = 1.0176$$

and

$$\delta_2 = 1.0332.$$

From Table 4.1, we see that we can expect to achieve the root Hermite factor $1.0195 \approx \delta_1$ using the BKZ algorithm with blocksize 20. From Figure 4.1, we see that the blocksize necessary to achieve the provable root Hermite factor δ_1 is 85. Similarly, the expected and provable BKZ blocksizes necessary for the root Hermite factor δ_2 are 2 and 28. As discussed in Section 4.3.1, the expected value represents our best guess at the hardness of worst-case SIVP_γ, while the provable value is the only known upper bound on the hardness of worst-case SIVP_γ.

Finally, we can convert BKZ blocksizes to bits of security using the estimated runtime of $2^{0.292*\beta_{BKZ}}$ for BKZ with blocksize β_{BKZ} from Section 4.2.3. This gives us estimated costs of 2^6 and 2, and provable costs of 2^{24} and 2^8 for δ_1 and δ_2 respectively.

Thus, parameter set I gets at most $6 - 55 = -49$ bits[2] of security (or $24 - 55 = -31$ bits of provable security) from the worst-case to average-case reduction of Theorem 3.10.

[2]A security level of $-\ell$ bits (where $\ell > 0$) means that the fastest attack takes time $2^{-\ell}$ operations. Note that this implies that to achieve this security level breaking the cryptosystem must take less than 1 operation, which is of no use as a security guarantee.

Alternatively, it gets at most $1-18 = -17$ bits of security (or $8-18 = -10$ bits of provable security) from the worst-case to average-case reduction of Theorem 3.11.

We can contrast this with the 51 bits of security that Lyubashevsky claimed for parameter set I. Note that the direct bits of security (i.e. the value obtained by evaluating the hardness of SIS against known attacks) would again be slightly higher, as we would need to increase the value of q for Theorems 3.10 and 3.11 to apply.

We can do a similar analysis for parameter sets II and III. The results are presented in Table 5.3. Note that the tightness gaps presented in Table 5.3 do not include the factor $\times \frac{2(h+s)}{\delta}$ that comes from the Forking Lemma.

	g_1	γ_1	BKZ-β_1	Security Level	g_2	γ_2	BKZ-β_2	Security Level
I	2^{55}	1.0176	20 (85)	6 (24)	2^{18}	1.0332	2 (28)	1 (8)
II	2^{54}	1.0168	20 (90)	6 (26)	2^{18}	1.0317	2 (28)	1 (8)
III	2^{56}	1.0194	20 (70)	6 (20)	2^{18}	1.0368	2 (28)	1 (8)

Table 5.3: Tightness Gaps and hardness of SIVP_γ instances. BKZ blocksizes and security levels are written as expected (provable).

We conclude that Lyubashevsky's signature scheme does not get any useful concrete security guarantees from the worst-case to average-case reductions, even ignoring the tightness gap that arises from the use of the Forking Lemma.

Chapter 6

Dilithium

The Crystals-Dilithium signature scheme [36] is one of the round 2 contenders for stateless signatures in the NIST Post-Quantum Cryptography standardization contest. Its security is based on the hardness of the SIS and LWE problems. We will present the signature scheme, its parameters, and a proof of correctness in Section 6.1. Section 6.2 presents a security reduction from MSIS to forging Dilithium signatures. Finally, Section 6.3 presents an analysis of the security guarantees for Dilithium, starting with an analysis of the security assurances Dilithium gets from SIS (in Section 6.3.1), and followed by an analysis of the concrete security assurances that Dilithium gets from worst-case SIVP_γ (in Section 6.3.2).

6.1 The Dilithium Signature Scheme

6.1.1 Parameters and Roles

Dilithium is defined over the ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, where $q = 2^{23} - 2^{13} + 1$ and $n = 256$. Both n and q are fixed for all security levels (Dilithium bases security levels on the size of the MSIS matrix) and were picked for efficient computation, but other choices of q and n are also possible.

There are multiple parameters that can be chosen in Dilithium. Many are fixed for all parameter sets, while others vary depending on the security level. We will first present the roles of the parameters, and then list the values they take in Table 6.1.

η A small integer, between 3 and 7, depending on the security level.

k, ℓ The size parameters for the keys. The matrix A is in $R_q^{k \times \ell}$, and the secret key (s_1, s_2) is in $S_\eta^k \times S_\eta^\ell$ (defined in Section 2.3.1). The values of k and ℓ vary depending on the security level.

γ_1 This parameter determines the maximum size of the coefficients of the masking vector of polynomials y. The bound for the Module Short Integer Solution problem (see Definition 2.19) that forgeries reduce to also depends on γ_1. Thus, γ_1 is picked large enough to mask the secret key, but small enough to ensure forgeries are hard.

γ_2 This parameter is used to determine how many bits of a vector are considered "high-order" bits. A validity check during signing also depends on γ_2, which is necessary for both security and correctness.

β_r This parameter is used in the rejection sampling used in Dilithium. In particular, β_r bounds the largest possible coefficient of cs_i. Then, if any coefficient of z is larger than $\gamma_1 - \beta_r$, we reject and restart the signing procedure. There is an efficiency/security tradeoff to consider when picking β_r, as smaller β_r implies fewer repetitions to get a signature, but a β_r that is too small can skew the distribution of signatures and reveal information about the secret key.

d Fixed parameter that determines how many bits are considered "high order bits" in the decomposition $x = x_1 2^d + x_0$ (where $x_0 \equiv x \mod^{\pm} 2^d$).

The fixed parameters and parameter choices for the different security levels of Dilithium are summarized in Table 6.1.

6.1.2 Simplifed Signature Scheme

The Dilithium signature scheme is based on a modified Fiat-Shamir with Aborts framework. We will first present a simplified version of the Dilithium signature scheme in Algorithms 9, 10, 11, and then describe the modifications that are added in the full signature scheme.

The function HighBits(x, α) is defined as follows. For each coefficient w of the element $x \in R_q$, write $w = w_1 \alpha + w_0$, where $|w_0| \leq \alpha/2$. The function HighBits(x, α) then returns a vector $\mathbf{w_1}$ comprised of all the elements w_1. Similarly, the function LowBits(x, α) returns a vector $\mathbf{w_0}$ comprised of all the elements w_0.

The reason why we can only output $z = y + cs_1$ some of the time is because the high-order coefficients in z can leak information about the secret key s_1. To avoid this leakage

Algorithm 9 Key Generation

Output: Public key $pk = (A, t)$ and secret key $sk = (s_1, s_2)$.

1: $A \leftarrow R_q^{k \times \ell}$
2: $(s_1, s_2) \leftarrow S_\eta^\ell \times S_\eta^k$
3: $t \leftarrow As_1 + s_2$
4: **return** $pk = (A, t), sk = (s_1, s_2)$

Algorithm 10 Signature Generation

Input: $sk = (s_1, s_2)$, message $M \in \{0,1\}^*$.
Output: Signature $\sigma = (z, c)$, where $z \in R_q$ and $c \in B_{60}$.

1: $z = \perp$
2: **while** $z = \perp$ **do**
3: $\quad y \leftarrow S_{\gamma_1 - 1}^\ell$
4: $\quad \mathbf{w_1} \leftarrow \text{HighBits}(Ay, 2\gamma_2)$
5: $\quad c \leftarrow H(M || \mathbf{w_1})$, where the range of H is B_{60}.
6: $\quad z \leftarrow y + cs_1$
7: $\quad$ **if** $\|z\|_\infty \geq \gamma_1 - \beta_\tau$ or $\|\text{LowBits}(Ay - cs_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta_\tau$ **then** $z = \perp$
8: **return** $\sigma = (z, c)$

Algorithm 11 Signature Verification

Input: $pk = (A, t)$ and signed message (M, σ).
Output: "Accept" or "Reject".

1: $\mathbf{w_1}' \leftarrow \text{HighBits}(Az - ct, 2\gamma_2)$
2: "Accept" if $\|z\|_\infty < \gamma_1 - \beta_\tau$ and $c = H(M || \mathbf{w_1}')$; else "Reject".

Table 6.1: Dilithium parameters

Parameter Set	I	II	III	IV
q	$2^{23} - 2^{13} + 1 = 8380417$			
d	14			
n	256			
Weight of c	60			
$\gamma_1 = (q-1)/16$	523776			
$\gamma_2 = \gamma_1/2$	261888			
(k, ℓ)	$(2, 3)$	$(4, 3)$	$(5, 4)$	$(6, 5)$
η	7	6	5	3
β_r	375	325	275	175
Security Level from MSIS (in bits)	68	103	138	176

of information, Dilithium uses rejection sampling, which ensures that no coefficients of the signature component z are too large ($\geq \gamma_1 - \beta_\tau$) and similarly that no coefficients of the low order bits of $Ay - ct$ are too large ($\geq \gamma_2 - \beta_\tau$). The parameters are selected so that the expected number of repetitions is not too large (between 4 and 7 repetitions for the proposed parameters).

6.1.3 Proof of Correctness

Notice that

$$Az - ct = Az - c(As_1 + s_2) = Az - Acs_1 - cs_2 = Ay - cs_2$$

by definition. Additionally, we know that the low-order bits of $Ay - cs_2$ are strictly less than $\gamma_2 - \beta_r$. Since the coefficients of cs_2 are at most β_r (by definition of β_r), this means that $Ay - cs_2 + cs_2 = Ay$ has the same high-order bits as $Ay - cs_2 = Az - ct$, and hence that $w_1 = w_1'$. Thus, $H(M||w_1) = H(M||w_1')$ and this concludes the proof of correctness.

6.1.4 Full Signature Scheme

Dilithium introduces some modifications to the above signature scheme template. We will briefly discuss them here, but will not present the signature scheme in full detail. The

concrete details are unnecessarily complicated for the purposes of this chapter, and do not affect the discussion that follows.

The first major improvement is to reduce the size of the public key by replacing the matrix $A \in R_q^{k \times \ell}$ by a seed ρ. The matrix A is then generated from the seed ρ by using the extendable output function SHAKE-128.

A second improvement further reduces the public key size, in exchange for a small increase in signature size. This is done by only including the high-order bits (in the binary representation) of $t = As_1 + s_2$, and modifying the HighBits function to take as input an additional one bit hint (for each coefficient) that lets us compute $Az - ct$ given only the high-order bits of t.

6.2 Security Proof

From the NIST submission, we can characterize the security of Dilithium by the following statement:

Theorem 6.1. *If H is a quantum random oracle, the advantage of an adversary A breaking the SUF-CMA security of the signature scheme is*

$$\mathrm{Adv}^{\mathrm{SUF-CMA}}_{\mathrm{Dilithium}} \leq \mathrm{Adv}^{\mathrm{MLWE}}_{k,\ell,S_\eta} + \mathrm{Adv}^{\mathrm{SelfTargetMSIS}}_{H,k,\ell+1,4\gamma_2} + \mathrm{Adv}^{\mathrm{MSIS}}_{k,\ell,4\gamma_2+2} + 2^{-254}.$$

Remark. The authors of Dilithium did not present a classical security result. We will assume that this statement of security is also valid for a classical adversary.

The proof of Theorem 6.1 can be found in [29]. Note that the MLWE assumption protects against key recovery attacks. Thus, the hardness of forging Dilithium signatures depends on the hardness of the $\mathrm{MSIS}_{k,\ell,4\gamma_2+2}$ and $\mathrm{SelfTargetMSIS}_{k,\ell,4\gamma_2}$ problems. We will only be analyzing the hardness of forging Dilithium signatures.

SelfTargetMSIS (see Definition 2.20) is not a standard lattice problem, as it's essentially the convolution of a hash function and the MSIS problem. Lemma 6.2 presents a classical reduction from MSIS to SelfTargetMSIS. A sketch of this reduction was presented in the NIST Round 2 submission as well as in [29].

Lemma 6.2. *There is a classical reduction from* $\mathrm{MSIS}_{2\beta}$ *to* $\mathrm{SelfTargetMSIS}_{\beta}$.

If the SelfTargetMSIS *oracle succeeds with probability δ and makes h queries to the random oracle, then this reduction succeeds with probability $\frac{\delta^2}{h}$ in two calls to the oracle.*

Proof. Suppose we have access to an oracle that solves SelfTargetMSIS$_\beta$ (this oracle has access to the random oracle H), i.e., given as input some matrix A and a vector t, finds an M and a $y = \begin{bmatrix} r_1 \\ c \\ r_2 \end{bmatrix}$ such that $H(M||[A|t|I] \cdot y) = c$ and $0 < \|y\|_\infty \leq \beta$.

Suppose now that we are given an MSIS$_{2\beta}$ instance $[A|t]$. We run the SelfTargetMSIS oracle with input (A, t) to obtain M and $y = \begin{bmatrix} r_1 \\ c \\ r_2 \end{bmatrix}$ such that

$$H(M||[A|t|I] \cdot y) = c$$

and $0 < \|y\|_\infty \leq \beta$. Then, using the Forking Lemma (Lemma 2.22), we rewind the SelfTargetMSIS oracle and return different responses to its random oracle queries, to obtain $y' = \begin{bmatrix} r'_1 \\ c' \\ r'_2 \end{bmatrix}$ such that

$$H(M||[A|t|I] \cdot y') = c'$$

and $0 < \|y\|_\infty \leq \beta$ for some hash value $c' \neq c$. If the SelfTargetMSIS$_\beta$ oracle succeeds with probability δ, then this succeeds with probabilty $\frac{\delta^2}{h+s}$.

Since the random oracle must have been queried with the same string $M||s$, we must have

$$[A|t|I] \cdot y = s = [A|t|I] \cdot y',$$

and hence

$$[A|t|I](y - y') = 0.$$

We know that $0 < \|y - y'\|_\infty \leq \|y\|_\infty + \|y'\|_\infty \leq 2\beta$ by the triangle inequality. Thus, $y - y'$ is a solution to the MSIS$_{2\beta}$ instance $[A|t]$.

The probability of success of this reduction is the probability of success of the Forking Lemma, $\frac{\delta^2}{h}$. □

We can combine Theorem 6.1 and Lemma 6.2 to show that if we can forge Dilithium signatures, then we can solve either MSIS$_{k,\ell,4\gamma_2+2}$ or MSIS$_{k,\ell+1,8\gamma_2}$, with tightness gap

$$2\frac{\delta}{h+s}$$

where h is the number of queries to the random oracle (made during the reduction of Theorem 6.1 or of Lemma 6.2) and s the number of queries to the signing oracle (made during the reduction of Theorem 6.1).

Note that the authors of Dilithium know that this reduction is non-tight, and instead assume that the only way to find an SelfTargetMSIS solution is to pick some w, compute $H(M\|w) = c$, and then find z', u' such that $Az' + u' = w + ct$ [36]. Thus, solving the above SelfTargetMSIS problem is at least as hard as finding z', u' with ℓ_∞ norm at most $4\gamma_2$ such that

$$Az' + u' = t'$$

for some given t'. This is essentially ignoring the loss of tightness of the Forking Lemma in the proof of security, instead arguing [36] that

> since H is a cryptographic hash function whose structure is completely independent of the algebraic structure of its inputs, choosing some M "strategically" should not help – so the problem would be equally hard if the M were fixed.

Note that finding a vector of length $8\gamma_2$ is no harder than finding a vector of length $4\gamma_2+2$. Additionally, the approximation factors of the SIVP_γ problems from Theorems 3.9, 3.10 and 3.11 do not depend on the value of m, and their tightness gaps increase as m increases. Thus, we will be basing our analysis on $\text{MSIS}_{k,\ell,4\gamma_2+2}$, as this gives us an upper bound on the security assurances that Dilithium gets from worst-case SIVP_γ.

6.3 Concrete Security Guarantees

6.3.1 Security Guarantees from SIS

In this section, we will examine the hardness of the $\ell_\infty\text{−SIS}_{n,m,q,\beta}$ problem related to Dilithium. We will briefly discuss the method that the Dilithium team uses to determine their security estimates, followed by an alternate method supporting this security estimate.

The Dilithium team determines the best ℓ_2−SVP algorithm is BKZ (which is also the best ℓ_2−SIS algorithm) and considers its effectiveness at solving ℓ_∞−SIS. They describe well known methods for describing the shape and quality of BKZ output vectors (in the ℓ_2 norm) for a given blocksize, and compare the length of the shortest vectors output by BKZ (in the ℓ_2-norm) to their ℓ_∞−SIS bound. They do not directly address the relative costs

of solving ℓ_2-SIS and $\ell_\infty-$SIS, but rather evaluate the effectiveness of current algorithms at solving $\ell_\infty-$SIS.

The Dilithium team also consider the effects of ignoring columns of the SIS matrix (which is overdetermined) and of re-randomizing parts of the matrix in their cost analysis. The algorithms they use to determine cost of $\ell_\infty-$SIS can be found at https://github.com/pq-crystals/security-estimates. For their level III parameters, which they say gives them 138 bits of security, they give the best estimated blocksize to break the MSIS instance they reduce to as $\beta_{BKZ} = 475$. According to (4.1), this blocksize corresponds to a root Hermite factor $\delta_0 = 1.003478$.

On the other hand, we can build on the discussion from Section 4.3.3 to use the hardness of $\ell_2-\text{SIS}_{n,m,q,\beta}$ and $\ell_2-\text{SVP}_\gamma$ to bound the hardness of an $\ell_\infty-\text{SIS}_{n,m,q,\beta_\infty}$ instance.

Recall that our $\ell_\infty-$SIS instance is given as a tuple

$$(n, m, q, \beta_\infty) = (256k, 256(k+\ell), q, \frac{q-1}{8}),$$

and defined by a (random) matrix $[A|I] \in \mathbb{Z}_q^{256k \times 256(k+\ell)}$. The lattice related to this SIS instance is

$$\mathcal{L}^\perp(A) = \{x \in \mathbb{Z}^m : [A|I]x = 0 \mod q\},$$

and it has (with high probability) volume q^n.

From Section 4.3.3, we know that we have a reduction from $\ell_2-\text{SIS}_{n,m,q,\beta_\infty\sqrt{m}}$ to $\ell_\infty-\text{SIS}_{n,m,q,\beta_\infty}$, so $\ell_\infty-\text{SIS}_{n,m,q,\beta_\infty}$ is at least as hard as $\ell_2-\text{SIS}_{n,m,q,\beta_\infty\sqrt{m}}$. By the discussion in Section 4.3.2, this means that solving an $\ell_\infty-\text{SIS}_{n,m,q,\beta_\infty}$ instance is at least as hard as using BKZ to solve an $\ell_2-\text{SIS}_{n,m,q,\beta_\infty\sqrt{m}}$ instance and achieve a root Hermite factor of

$$\left(\frac{\sqrt{m}\beta_\infty}{q^{n/m}}\right)^{1/m}.$$

For the Dilithium level III parameters, we get a root Hermite factor $\delta_{\sqrt{m}\beta} \approx 1.00386$, which corresponds (based on Figure 4.2) to a BKZ blocksize of around 420.

Note that the discussion from Section 4.3.3 on the hardness of $\ell_\infty-\text{SVP}_\gamma$ suggests that re-randomizing the lattice basis might improve the reduction from $\ell_2-\text{SIS}_{n,m,q,\beta}$ to $\ell_\infty-\text{SIS}_{n,m,q,\beta_\infty}$ to $\beta = \beta_\infty\sqrt{\ln m}$. It is interesting to note that the Dilithium team also remarked that

> re-randomizing[1] the input basis before running the BKZ algorithm [...] always improves over the [non randomizing] strategy for the parameter ranges considered.

The root Hermite factor corresponding to $\beta = \beta_\infty\sqrt{\ln m}$ is 1.00262, which corresponds to a BKZ blocksize of around 700.

Finally, we note that $\ell_\infty\text{−SIS}_{n,m,q,\beta_\infty}$ must be at most as hard as $\ell_2\text{−SIS}_{n,m,q,\beta_\infty}$, since if $\|v\|_2 \leq \beta$, then $\|v\|_\infty \leq \beta$ for any β. The root Hermite factor corresponding to $\beta = \beta_\infty$ is 1.002175, which corresponds to a BKZ blocksize of around 925.

Unfortunately, all reductions but the last one are meaningless for the specific Dilithium parameters as we have $\frac{q-1}{8} = \beta_\infty < q < \beta = \beta_\infty\sqrt{m}$, and it is trivial to find a vector of norm q for SIS instances. However, it is interesting to note that the BKZ blocksize found by the Dilithium team falls in the range of blocksizes presented here, suggesting that this approach has potential. Additionally, in a hypothetical situation where $\beta < q$ (which could be achieved by increasing q for the same values of β, β_∞), the methodology presented here would be applicable and the blocksizes found would provide a lower bound on the BKZ blocksizes needed to solve the SIS instance, as increasing q can only decrease the root Hermite factor.

6.3.2 Security Guarantees from SIVP

In this section, we will discuss the security guarantees that Dilithium gets from worst-case SIVP_γ to average-case SIS reductions. Recall from Section 6.2 that forging Dilithium signatures is at least as hard as solving $\ell_\infty\text{−MSIS}_{k,\ell,\beta_\infty}$ over $\mathcal{R}_q$ for $\beta_\infty = 4\gamma_2$.

We know that there is a reduction from MSIS to SIS:

Lemma 6.3. *There is a reduction from* $\ell_\infty\text{−MSIS}_{k,\ell,\beta_\infty}$ *over* $\mathcal{R}_q$ *to* $\ell_\infty\text{−SIS}_{256\cdot k,256\cdot(k+\ell),q,\beta_\infty}$.

Proof. We can solve an $\ell_\infty\text{−MSIS}_{k,\ell,\beta_\infty}$ instance for $A \in \mathcal{R}_q^{k\times\ell}$ by solving a related $\ell_\infty\text{−SIS}_{256\cdot k,256\cdot(k+\ell),q,\beta_\infty}$ instance determined by the matrix $\text{rot}(A) \in \mathbb{Z}_q^{256k\times 256\ell}$, defined by

[1] The meanings of re-randomizing are slightly different in both cases. The reduction from [46] transforms the basis into a basis for a different lattice, while the re-randomizing described in Dilithium changes the basis into a different basis for the same lattice.

expanding each $a_{i,j}$ in A into the 256×256 matrix

$$\mathrm{rot}(a) = \begin{bmatrix} a_0 & a_{n-1} & \dots & a_1 \\ a_1 & a_0 & \dots & a_2 \\ \vdots & \vdots & & \vdots \\ a_{n-1} & a_{n-2} & \dots & a_0 \end{bmatrix}.$$

This works since the coefficient of the product of R_q elements is given by a cyclic convolution of their coefficient vectors. □

Note that whereas there is no reduction in the reverse direction, there is also no known way to exploit the additional structure of an MSIS instance, so it is often assumed that the MSIS instance and the corresponding SIS instance are equivalent.

Additionally, recall from Chapter 3 that we have worst-case hardness results for an $\mathrm{SIS}_{n,m,q,\beta}$ instance where:

1. $q \geq 8n\sqrt{m}\beta$ (Theorem 3.9), or
2. $q \geq \beta\sqrt{n}\omega(\sqrt{\log n})$ (Theorem 3.10), or
3. $q \geq \beta n^c$ for some constant $c > 0$ (Theorem 3.11).

From the above, we can see that there are two obstacles with applying the worst-case to average-case reductions to Dilithium:

1. There is no known reduction from SIS to MSIS, and
2. For Dilithium parameters, we have $\beta = \sqrt{m}\beta_\infty > q$, so none of the reductions apply.

To address the first obstacle, we have a few options, which all lead to identical hardness results. First, we could assume there is a reduction from SIS to MSIS, and apply Theorems 3.9, 3.10, and 3.11.

Alternatively, we could note that Theorem 3.10 has been proved in the module setting, as Langlois and Stehlé [31] presented a reduction from Mod-SIVP$_\gamma$ to MSIS that has the same bounds on parameters (for n and m given by the dimensions of $rot(A)$) and achieves the same approximation factor[2]. Since the fastest known algorithm to solve Mod-SIVP$_\gamma$

[2]The tightness gap might be smaller, but the gap remains at least $O(n^2)$ as the reduction from GIVP to the intermediate problem remains unchanged.

is the same as the one for solving SIVP_γ, we get the same hardness for the worst-case problem that MSIS reduces to in both cases (with a possible difference in tightness factor). The MSIS to Mod-SIVP_γ reduction has not been generalized to Theorems 3.9 and 3.11, although there is no reason to believe that this is not possible.

Since Theorem 3.11 has not been generalized to the module setting and there are interesting tradeoffs between the parameter bounds and approximation factors achieved in Theorems 3.10 and 3.11, we will take the first approach and assume for the analysis in the rest of the chapter that MSIS and SIS (of the dimensions of $\text{rot}(A)$) are equivalent problems.

To address the second obstacle, one could increase q until it satisfies one of the conditions of Theorems 3.9, 3.10, or 3.11. Note that this wouldn't change the tightness gap of the proof or the approximation factor γ of the worst-case SIVP_γ problem that the MSIS instance reduces to. It would, however, increase the average-case security of the original MSIS problem and the key and signature sizes.

We will now present an analysis of the concrete security assurances that Dilithium gets from worst-case SIVP_γ to average-case SIS reductions from Theorems 3.10 and 3.11 for the Dilithium parameter set II. The worst-case security guarantees for other parameter sets are presented in Table 6.2.

Recall that parameter set II has $(k, \ell) = (4, 3)$. Thus, we have $n = 256 \cdot k = 1024$, $m = 256 \cdot (k + \ell) = 1792$, $\beta_\infty = 4\gamma_2 + 2 = 4 \cdot 261888 + 2 = 1047552$, and $q = 8380417$ (see Table 6.1). To solve the MSIS instance that reduces to Dilithium, we need to solve an instance of $\ell_\infty\text{–SIS}_{n,m,q,\beta_\infty}$.

From Theorem 3.10, we get an SIVP approximation factor of

$$\gamma_1 = \beta\sqrt{n}\omega(\sqrt{\log n}).$$

Choosing $\beta = \sqrt{m}\beta_\infty$ gives us

$$\gamma_1 = \beta_\infty\sqrt{nm}\omega(\sqrt{\log n}).$$

Substituting the parameters for set II, and converting γ_1 to a root Hermite factor δ_1 using the methodology from Section 4.3.1, we get

$$\delta_1 = \sqrt{\gamma_1/\sqrt{n}}^{1/n} = 1.00911.$$

From this root Hermite factor, we can use equation 4.1 to obtain a BKZ blocksize of 110 to achieve the root Hermite factor δ_1. We can also use Figure 4.1 to determine that we

would need to run BKZ with a blocksize of 360 to provably achieve the root Hermite factor δ_1. However, as discussed in Section 4.3.1, we believe the expected blocksize of 110 to give us a more accurate picture of the true hardness of worst-case SIVP_{γ_1}, as the values of Figure 4.1 are only an upper bound on the true worst-case behaviour of BKZ, and as the provable bounds in lattice reduction algorithms depend on a worst-case basis and not a worst-case lattice. Using the estimated runtime of $2^{0.292\beta_{BKZ}}$ from Section 4.2.3, we get that BKZ blocksizes of 110 and 360 correspond to 32 and 105 bits of security, respectively.

Finally, we can compute the tightness gap of this reduction, which is given by

$$g_1 = O\left(n^2\beta m\right) \times \frac{2(h+s)}{\delta} = n^2\beta_\infty\sqrt{m}m \times \frac{2(h+s)}{\delta} = 2^{56} \times \frac{2(h+s)}{\delta}$$

(by Section 6.2).

Ignoring the $\frac{2(h+s)}{\delta}$ tightness gap that arises from the Forking lemma, we see that parameter set II gets $32 - 56 = -24$ bits of security (or $105 - 56 = 49$ bits of provable security) from the worst-case to average-case reduction of Theorem 3.10. We can contrast this with the 103 bits of security that the Dilithium authors claim for parameter set II.

Note that the minimum value of q for Theorem 3.10 (and thus the above analysis) to apply is 2^{37} (which is a significant increase from the current value of $q = 2^{23}$). Note also that increasing q decreases the root Hermite factor of an SIS instance, so the SIS instance with q increased to 2^{37} would have more than 103 bits of security, and thus the difference between direct bits of security and worst-case bits of security would be even larger.

We can perform a similar analysis using the reduction from Theorem 3.11, which needs a minimum value of $q = 2^{25}$ to apply. For this reduction, we get

$$\gamma_2 = \frac{\beta_\infty}{n^c}\beta_\infty\sqrt{nm}\omega(\sqrt{\log n})$$

which converts to a root Hermite factor

$$\delta_2 = \left(\sqrt{\gamma_2/\sqrt{n}}\right)^{1/n} = 1.01596.$$

The root Hermite factor can be achieved by expected and provable BKZ blocksizes of 20 and 100, which correspond to 6 and 29 bits of security, respectively.

The tightness gap of this reduction is given by

$$g_2 = n^2 \times \frac{2(h+s)}{\delta} = 2^{19} \times \frac{2(h+s)}{\delta}$$

by Section 6.2.

Ignoring the $\frac{2(h+s)}{\delta}$ tightness gap that arises from the Forking lemma, we see that parameter set II gets $6 - 19 = -13$ bits of security (or $29 - 20 = 9$ bits of provable security) from the worst-case to average-case reduction of Theorem 3.11. We can contrast this with the 103 bits of security that the authors of Dilithium claim for parameter set II. Note that the direct bits of security would again be slightly higher, as we would need to increase q to 2^{25} for Theorem 3.11 to apply.

These computations, as well as similar computations for parameter sets I, III, and IV are summarized in Table 6.2.

Remark. We use $\omega(\sqrt{\log n}) = \sqrt{\log n}$, $n^c = 1$, and $g = 1$ in the computations in Table 5.2, as they are tight lower bounds for these values. We also omit the tightness gap from the Forking Lemma in the tightness gaps presented in Table 6.2, since it is not clear how to bound the values of h, s and δ. We leave it to the reader to choose appropriate values of these parameters. Thus, the tightness gap presented here is an underestimate of the true tightness gap.

Also note that we belive the expected root Hermite factors and BKZ blocksizes to more accurately represent the worst-case hardness of SIVP_γ. We include provable root Hermite factors and the corresponding BKZ blocksizes for reference purposes only. However, it is interesting to note that for some parameter sets, Dilithium does not get any useful concrete security guarantees even if we consider the security guarantees given by the provable BKZ output factors.

6.3.3 Discussion

In Section 6.3.2, we surmised that currently known worst-case to average-case reductions for the Short Integer Solution problem do not apply to Dilithium. Additionally, Table 6.2 shows that even if the reductions did apply to Dilithium (which could be achieved by increasing q and using equivalent reductions for module lattices), Dilithium would not derive any useful concrete security guarantees from the worst-case to average-case reductions.

However, Dilithium does get stronger security guarantees from the worst-case to average-case reductions than Lyubashevsky's signature scheme (in particular, Dilithium gets positive bits of security), which suggests it might be possible to construct an efficient signature scheme with a cryptographically significant number of bits of security from the worst-case hardness of SIVP_γ.

Parameter Set	I	II	III	IV
(k, ℓ)	$(3, 2)$	$(4, 3)$	$(5, 4)$	$(6, 5)$
$(n, m) = (256 \cdot k, 256 \cdot (k + \ell))$	$(768, 1280)$	$(1024, 1792)$	$(1280, 2304)$	$(1536, 2816)$
$\beta_\infty = 4\gamma_2 + 2$	1047552	1047552	1047552	1047552
Current q	2^{23}	2^{23}	2^{23}	2^{23}
Best Estimated Blocksize to Break SIS	235	355	475	605
Claimed Bits of Security (From SIS)	68	103	138	176
Min value of q for Theorem 3.10 to apply:	2^{37}	2^{37}	2^{38}	2^{38}
δ_1 (from $\gamma_1 = g\beta_\infty\sqrt{nm}\omega(\sqrt{\log n})$)	1.01204	1.00911	1.00734	1.00615
Estimated BKZ blocksize	55 (180)	110 (360)	155 (> 500)	210 (> 700)
Bits of Security	16 (52)	32 (105)	45 (> 138)	61 (> 176)
$g_1 = n^2\beta m$ $(\times\frac{2(h+s)}{\delta})$	2^{54}	2^{56}	2^{57}	2^{58}
Min value of q for Theorem 3.11 to apply:	2^{25}	2^{25}	2^{26}	2^{26}
δ_2 (from $\gamma_2 = \frac{\beta_\infty}{n^c}\beta_\infty\sqrt{nm}\omega(\sqrt{\log n})$)	1.02121	1.01596	1.01281	1.0107
Estimated BKZ blocksize	20 (60)	20 (100)	28 (155)	80 (250)
Bits of Security	6 (17)	6 (29)	8 (45)	23 (73)
$g_2 = n^2$ $(\times\frac{2(h+s)}{\delta})$	2^{19}	2^{20}	2^{20}	2^{21}

Table 6.2: Tightness Gaps and hardness of SIVP instances for Dilithium. BKZ blocksizes and Security Levels are written as expected (provable).

Chapter 7

Concluding Remarks

We analyzed the tightness of various reductions from worst-case SIVP_γ to average-case $\text{SIS}_{n,m,q,\beta}$. We then used these to analyze the security assurances that the worst-case to average-case reductions grant to two real-world signature schemes based on SIS or one of its variants.

For Lyubashevsky's signature scheme (see Chapter 5), we found that the reductions cited as granting worst-case security guarantees (see Theorems 3.9 and 3.10) were not applicable to any of the presented parameter sets. However, a later reduction by Peikert and Micciancio (see Theorem 3.11) does apply.

We also studied the signature scheme Crystals-Dilithium (see Chapter 6) from the NIST PQC competition. For the signature scheme Dilithium, we found that none of the known worst-case to average-case reductions for SIS or MSIS apply.

For both signature schemes, we considered the worst-case to average-case assurances that the signature schemes would enjoy if parameters were modified so that the reductions from Theorems 3.9, 3.10, and 3.11 applied. We found that the SIVP_γ instance that the signature schemes reduce to are much easier to solve (in the worst-case) than the starting SIS instance (in the average-case). Additionally, the significant tightness gap of the reduction implies that neither signature scheme would get any useful security guarantees from the worst-case to average-case reductions.

We note that there were portions of this work that haven't been studied much (if at all) in the literature. We believe that the following directions of research would improve our understanding of the worst-case hardness of SIS-based cryptosystems, as well as of cryptosystems based on the worst-case hardness of other lattice problems. We leave their study to future work.

1. Better provable bounds for the worst-case behaviour of BKZ.
2. A better understanding of the structure and hardness of worst-case lattice problems.
3. Proving results corresponding to Theorems 3.9 and 3.11 in the module lattice setting.
4. Studying the hardness of $\ell_\infty - \mathrm{SIS}_{n,m,q,\beta_\infty}$ when the ℓ_2 bound of solutions is $> q$, both in the direct and in the worst-case settings.
5. Finding an efficient SIS-based signature scheme (or finding a parameter set for Lyubashevsky's signature scheme) that gets 2^{128} bits of worst-case security.
6. Selecting Dilithium parameters so that the worst-case to average-case reduction provides meaningful security assurances for Dilithium, without severely impacting performance.

Finally, we note that the methodology used to analyze the worst-case security assurances is at least as important as the security assurances we have explicitly computed. Our analysis can be easily modified to obtain worst-case security assurances for other cryptosystems or if new reductions or bounds are obtained in the future.

References

[1] TU Darmstadt Lattice Challenge. https://www.latticechallenge.org/.

[2] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillippe Gaborit, and Gilles Zémor. RQC. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[3] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and Gilles Zémor. HQC. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[4] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. pages 601–610.

[5] Martin Albrecht, Carlos Cid, Kenneth G. Paterson, Cen Jung Tjhai, and Martin Tomlinson. NTS-KEM. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[6] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the LWE, NTRU schemes! pages 351–367.

[7] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The General Sieve Kernel and New Records in Lattice Reduction. In *Advances in Cryptology – EUROCRYPT 2019*, pages 717–746. Springer International Publishing, 2019.

[8] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum Key Exchange - A New Hope. pages 327–343.

[9] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zémor. BIKE. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[10] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zemor, Carlos Aguilar Melchor, Slim Bettaieb, Loic Bidoux, Magali Bardet, and Ayoub Otmani. ROLLO. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[11] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. LEDAcrypt. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[12] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. pages 390–399.

[13] Daniel J. Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[14] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU Prime. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[15] Bernstein, Daniel J., Buchmann, Johannes, Dahmen, Erik (Eds.). *Post-Quantum Cryptography*. Springer, 2009.

[16] Ward Beullens, Bart Preneel, Alan Szepieniec, and Frederik Vercauteren. LUOV. Technical report, National Institute of Standards and Technology, 2019. avail-

able at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[17] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qTESLA. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[18] A. Casanova, J.-C. Faugère, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. GeMSS. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[19] Sanjit Chatterjee, Neal Koblitz, Alfred Menezes, and Palash Sarkar. Another look at tightness II: Practical issues in cryptography. In *Paradigms in Cryptology – Mycrypt 2016,* volume 10311 of *Lecture Notes in Computer Science*, pages 21–55. Springer International Publishing, 2017.

[20] Sanjit Chatterjee, Alfred Menezes, and Palash Sarkar. Another look at tightness. pages 293–319.

[21] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. pages 1–20.

[22] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. SABER. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[23] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[24] Oscar Garcia-Morchon, Zhenfei Zhang, Sauvik Bhattacharya, Ronald Rietman, Ludo Tolhuizen, Jose-Luis Torre-Arce, Hayo Baan, Markku-Juhani O. Saarinen, Scott Fluhrer, Thijs Laarhoven, and Rachel Player. Round5. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[25] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. pages 197–206.

[26] Mike Hamburg. Three Bears. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[27] Andreas Hulsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, and Peter Schwabe. SPHINCS+. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[28] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik. SIKE. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[29] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. pages 552–586.

[30] Thijs Laarhoven, Joop van de Pol, and Benne de Weger. Solving hard lattice problems and the security of lattice-based cryptosystems. Cryptology ePrint Archive, Report 2012/533, 2012. http://eprint.iacr.org/2012/533.

[31] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[32] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[33] Lászlo Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*. SIAM, 1986.

[34] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, and Zhenfei Zhang. LAC. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[35] Vadim Lyubashevsky. Lattice signatures without trapdoors. pages 738–755.

[36] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[37] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems*. Springer US, 2002.

[38] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. pages 21–39.

[39] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

[40] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. pages 1468–1480.

[41] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[42] Thomas Poppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, and Douglas Stebila. NewHope. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[43] Thomas Poppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, and Douglas Stebila. NewHope. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[44] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[45] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. pages 84–93.

[46] Oded Regev and Ricky Rosen. Lattice problems and norm embeddings. pages 447–456.

[47] Simona Samardjiska, Ming-Shing Chen, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. MQDSS. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[48] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[49] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. pages 124–134.

[50] Herbert Wilf. Backtrack: An O(1) expected time algorithm for the graph coloring problem. *Information Processing Letters*, pages 119–121.

[51] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, and Daniel Slamanig. Picnic. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[52] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, William Whyte, John M. Schanck, Andreas Hulsing, Joost Rijneveld, Peter Schwabe, and Oussama Danba. NTRU. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.