Institute of Information Security

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

A Tool for the Estimation of Lattice Parameters

Nicolai Krebs

Course of Study:	Informatik, B.Sc.
Examiner:	Prof. Dr. Ralf Küsters
Supervisor:	Marc Rivinius, M.Sc.
Commenced:	April 22, 2021
Completed:	October 22, 2021

Abstract

<Short summary of the thesis>

Contents

1 Introduction **15**

2 Preliminaries **17**

2.1 Notation . . . 17
2.2 Norms . . . 17
2.3 Lattices . . . 17
2.3.1 Lattice Problems . . . 20
2.4 Discrete Gaussian Distribution . . . 21
2.5 LWE and SIS . . . 21
2.5.1 Learning with Errors (LWE) . . . 21
2.5.2 Short Integer Solution (SIS) . . . 23
2.5.3 Ring and Module Variants . . . 23

3 Algorithms and Estimates **27**

3.1 Lattice Basis Reduction . . . 27
3.1.1 The LLL Algorithm . . . 28
3.1.2 The BKZ Algorithm . . . 29
3.1.3 Cost Models for Lattice Reduction . . . 30
3.2 Algorithms for Solving LWE . . . 33
3.2.1 Lattice Problems . . . 33
3.2.2 Approaches . . . 34
3.2.3 Decoding Attack . . . 35
3.2.4 Primal-uSVP . . . 36
3.2.5 The BKW Algorithm . . . 38
3.2.6 Dual Attack . . . 40
3.2.7 Other Approaches . . . 41
3.3 Algorithms for Solving SIS . . . 41
3.3.1 Lattice Reduction . . . 41
3.3.2 Combinatorial Attack . . . 43

4 Lattice Parameter Estimation Tool **45**

4.1 Supported Distributions . . . 45
4.1.1 Gaussian Distribution . . . 45
4.1.2 Uniform Distribution . . . 46
4.2 Norms bound estimates . . . 47
4.3 Problem Classes . . . 48
4.4 Parameter Search . . . 51
4.5 Estimation Configuration Options . . . 52

5 Usage Examples 57
5.1 Two Problem Search 57
5.2 TODO: find other schemes to apply 57

6 Conclusion 59

Bibliography 61

Appendix 69

A Additional Math 71
A.1 Linear Codes 71

B Tables 73
B.1 Mapping of Parameters in Referenced Papers 73

List of Figures

3.1 An overview of the problems and algorithms used in this work. The diamond shaped boxes represent problems and arrows reductions between problems. Rectangular boxes represent approaches, classes of algorithms or algorithms. A connection between a problem and a rectangular box indicates that the latter can be used as a solver for the respective problem. Bold names indicate algorithms that can be used in our tool. 34

4.1 Problem classes 49

4.2 Cost models 53

4.3 SIS instance with $\sigma = 2.828,\ m = n^2,\ 2^{2n} < q < 2^{2n+1}$ 54

4.4 LWE instance with $\sigma = 0.125,\ m = \infty,\ 2^n < q < 2^{n+1}$ 55

4.5 LWE instance with $\sigma = 2.828,\ m = \infty,\ 2^n < q < 2^{n+1}$ 55

List of Tables

3.1 Overview of popular sieving algorithms 32
3.2 SVP cost models overview (based on [ACD+18, Table 4]) 33

4.1 LWE estimate algorithm priorities 56
4.2 SIS estimate algorithm prioritie 56

B.1 Parameter mapping from [LPR13] 73
B.2 Parameter mapping from [BDL+18] 73
B.3 Parameter mapping from [DOTT21] 73

List of Listings

List of Algorithms

1 Introduction

- rise of quantum computing (short history)

* conceptual

* reality

- problem: some hard classical problems no longer hard

* Shor's Algorithm (Peter Shor, 1994) => quantum computers can solve the factoring and the discrete logarithm problem in polynomial time

* application to encryption

* overview of current encryption methods that will become insecure

- one solution (among hash-based, code-based, isogeny-based, and multivariate): lattice crypto

* overview over history and capability of lattice crypto

* advantages: good (quasilinear) asymptotic key sized, good concrete runtimes and key sizes, worst-case secure instantiations, advanced cryptographic primitives previously infeasible

* including intro to LWE/SIS and applications to build crypto systems

. SIS: signature schemes, hash functions

. LWE: "cryptomania" applications (PKE, ...), signature schemes, lines:

- cryptographic applications

- establishing theoretical and asymptotic hardness [Reg05] [BLP+13; MP13] - concrete hardness of LWE: attacks, runtime estimates,

* briefly outline concept and benefits of hard-case to average-case reductions

- purpose of this thesis

* building schemes: need realistic hardness estimates of schemes for given parameter settings

* lack in the past: no unified/easy to use tool => thesis aims to solve this problem tool we call *Lattice Parameter Estimation* LWE instances are estimated by calling various estimation functions from the LWE Estimator [APS15], which we will refer to as *estimator*.

- overview of chapters/how to read

2 Preliminaries

2.1 Notation

In the following, we denote vectors by bold lower-case letters like $\mathbf{v}$ and matrices by bold upper-case letters $\mathbf{M}$. We interchangably use matrix notation and sets of column vectors $[\mathbf{v}_1 \cdots \mathbf{v}_n] = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$. Unless specified otherwise, by $\| \cdot \|$ or simply *norm* we refer to the Euclidean norm. By $[n]$ we denote the set $\{1, \dots, n\}$ for $n \in \mathbb{Z}^+$.

2.2 Norms

We can define the standard ℓ_p-norms on the vector space $\mathbb{R}^n$. In this work we will also consider rings and modules (see Section 2.5.3) and require a way to bound the length of ring and module elements. Let $\mathcal{R}_q = \mathbf{Z}_q[X]/\langle X^n + 1\rangle$ be a quotient ring as defined in [BDL+18] and $f \in \mathcal{R}_q$ with $f = \sum_i f_i X^i$. Following [BDL+18] we define the norms

$$\begin{aligned} \ell_1 &: \|f\|_1 = \sum_i |f_i|, \\ \ell_2 &: \|f\|_2 = \sqrt{\sum_i |f_i|^2} \text{ and} \\ \ell_p &: \|f\|_p = \left(\sum_i |f_i|^p\right)^{\frac{1}{p}} \text{ and} \\ \ell_\infty &: \|f\|_\infty = \max_i |f_i|. \end{aligned} \tag{2.1}$$

For standard vector spaces, the definitions are essentially equivalent to the above, except that f is a vector in $\mathbb{R}^n$ with coefficients f_i. For a module element $\mathbf{f} \in \mathcal{R}_q^d$, we can simply view $\mathbf{f}$ as a $n \cdot d$-dimensional vector.

2.3 Lattices

Lattices have been used in cryptography since the end of the last century. Lattice reduction algorithms such as the LLL algorithm resulted in a number of practical attacks on popular cryptosystems [NV10]. The landmark work of Ajtai [Ajt96] opened up the way for a new family of cryptosystems. He showed that any instance of a certain lattice problem, the approximate Shortest Vector Problem (SVP$_\gamma$) can be reduced to a randomized instance of the Short Integer Solution (SIS) problem. In particular, this implies that an average-case SIS instance is at least as hard as a worst-case instance of SVP$_\gamma$, resulting in extremely strong security guarantees. Later, Regev came up with a similar worst-case to average-case reduction for the Learning with Errors (LWE) problem [Reg05].

We now present the mathematical definition of lattices and related tools that we will use later on. Most of the background theory is based on material from [MG02] with some adaptions.

A *lattice* Λ is a discrete additive subgroub of the vector space $\mathbb{R}^m$. Every lattice can be defined by a basis $\mathbf{B}$ of n linearly independent basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ with $m \geq n$. A lattice then is the set of all integer linear combinations of the basis vectors in $\mathbf{B}$.

Definition 2.3.1 (Lattice)
Given a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{m \times n}$. *We call* Λ *a lattice generated by the column vectors of* $\mathbf{B}$ *if*

$$\Lambda(\mathbf{B}) = \left\{ \mathbf{x} \in \mathbb{R}^m \;\middle|\; \exists c_1, \ldots, c_n \in \mathbb{Z} : \mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{b}_i \right\}. \tag{2.2}$$

We call n the *rank* of the lattice. A lattice has full rank if the $n = m$. The basis of a lattice $\mathbf{B}$ is not unique. If $\mathbf{B}$ is a basis of a lattice Λ then for any unimodular matrix $\mathbf{U} \in \mathbb{Z}^{n \times n}$ with determinant ± 1 the basis $\mathbf{B} \cdot \mathbf{U}$ is also a basis of Λ. In cryptographic applications, we usually start with a lattice basis $\mathbf{B}$ with long basis vectors that are not very orthogonal to each other. For example, the problem of finding a short vector (SVP or SVP$_\gamma$, see Definition 2.3.2) in $\Lambda(B)$ turns out to be very difficult. If we can somehow find a "nicer" basis for $\Lambda(B)$ with shorter and more orthogonal basis vectors, finding a solution becomes a lot easier. We can improve the quality of a basis or "reduce" a basis in such a way by means of lattice basis reduction. But more to that later.

Similar to a quotient ring $\mathbb{Z}/q\mathbb{Z}$ of integers modulo some positive integer q with cosets $c + q\mathbb{Z}$, we can define the quotient group $\mathbb{R}^m/\mathbf{\Lambda}$ with *cosets*

$$\mathbf{c} + \Lambda = \{\mathbf{c} + \mathbf{v} \mid \mathbf{v} \in \Lambda\} \tag{2.3}$$

where $\mathbf{c} \in \mathbb{R}^m$ [Pei16].

An important concept is the *fundamental domain* of a lattice. The fundamental domain is a subset of $\mathbb{R}^m$ that contains exactly one representative of every coset of $\mathbb{R}^m/\mathbf{\Lambda}$. The most commonly considered fundamental domain of a lattice with basis $\mathbf{B}$ is the (shifted) *fundamental parallelipiped*

$$\mathcal{P}_{\frac{1}{2}}(\mathbf{B}) = \mathbf{B} \cdot \left[-\frac{1}{2}, \frac{1}{2}\right)^n = \left\{ \mathbf{x} \in \mathbb{R}^m \;\middle|\; \mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{b}_i,\ c_i \in \left[-\frac{1}{2}, \frac{1}{2}\right) \right\} \tag{2.4}$$

Another region that is often used is the *Voronoi region* $\mathcal{V}$ [GJS15] and is defined as

$$\mathcal{V} = \{\mathbf{x} \in \mathbb{R}^n \mid \forall \mathbf{y} \in \Lambda : ||\mathbf{x}|| \leq ||\mathbf{x} - \mathbf{y}||\}. \tag{2.5}$$

The n-dimensional volume of the fundamental parallelipiped of a lattice $\Lambda(\mathbf{B})$ is equivalent to the *determinant* of the lattice $\det(\Lambda(\mathbf{B})) = \sqrt{\det(\mathbf{B}^\intercal \mathbf{B})}$. For a full-ranked lattice the determinant becomes $\det(\Lambda(\mathbf{B})) = |\det(\mathbf{B})|$. The determinant is independent from the used basis which can be easily verified by considering $\det(\Lambda(\mathbf{UB})$.

The *minimum distance* $\lambda_1(\Lambda)$ of a lattice is the length of its shortest nonzero vector

$$\lambda_1(\Lambda) = \min_{v \in \Lambda \setminus \{0\}} \|\mathbf{v}\|. \tag{2.6}$$

Furthermore, we can define *ith successive minima* $\lambda_i(\Lambda)$ by considering an m-dimensional ball $\mathcal{B}(\mathbf{0}, r)$ with increasing radius $r \in \mathbb{R}$ and center $\mathbf{0} \in \mathbb{R}^m$ at the origin of Λ. Then $\lambda_i(\Lambda)$ is the smallest radius r such that the ball $\mathcal{B}(\mathbf{0}, r)$ contains exactly i linearly independent lattice vectors. Note that an optimally reduced basis with basis vectors $\mathbf{b}_1 \leq \mathbf{b}_2 \cdots \leq \mathbf{b}_n$ satisfies $\lambda_i(\Lambda) = \|mathbf{b}_i\|$ for all $i \in [n]$.

In general it is hard to determine the exact values of $\lambda_i(\Lambda(\mathbf{B}))$ for a given basis $\mathbf{B}$. *Minkowski theorem* states that $\lambda_1(\Lambda) \leq \sqrt{n} \cdot (\det(\Lambda))^{\frac{1}{n}}$ given that Λ has rank n. The *Gaussian heuristic* is commonly used to estimate the minimum distance λ_1 of a lattice Λ given the determinant $\det \Lambda$:

$$\lambda_1(\Lambda) \approx \frac{\Gamma(1 + n/2)^{1/n}}{\sqrt{\pi}} \det(\Lambda)^{1/n} \tag{2.7}$$

By applying Stirling's formula to estimate the Γ-function as described in [Göp16], we can simply the estimate to

$$\lambda_1(\Lambda) \approx \sqrt{\frac{n}{2\pi e}} \det(\Lambda)^{1/n} \tag{2.8}$$

The dual $\Lambda^\perp$ of a lattice $\Lambda(\mathbf{B})$ is defined as the set of vectors $\mathbf{y}$ in the span of $\mathbf{B}$ such that the inner product $\langle \mathbf{y}, \mathbf{v} \rangle$ is an integer for all $\mathbf{v} \in \Lambda(\mathbf{B})$. The basis the dual of a lattice with basis $\mathbf{B}$ is given by $\mathbf{B}' = \mathbf{B}(\mathbf{B}^\intercal \mathbf{B})^{-1}$.

In cryptography, we are mainly interested in modular integer (or q-ary) lattices. A q-ary lattice is a lattice Λ_q such that $q\mathbb{Z}^m \subseteq \Lambda \subseteq \mathbb{Z}^m$ given $q \in \mathbb{N} \setminus \{0\}$. This means that a vector $\mathbf{x} \in \mathbb{Z}^m$ is in Λ if and only if $\mathbf{x} \mod q$ also is in Λ.

We now look at two important ways of specifying a q-ary lattice given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ [BBGS19].

$$\Lambda_q(\mathbf{A}^\intercal) = \{\mathbf{v} \in \mathbb{Z}^m \mid \exists \mathbf{y} \in \mathbb{Z}^n : \mathbf{v} = \mathbf{A}^\intercal \mathbf{y} \mod q\} \tag{2.9}$$

$\Lambda_q(\mathbf{A}^\intercal)$ is commonly referred to as the *(primal) LWE lattice* as finding a short vector in $\Lambda_q(\mathbf{A}^\intercal)$ corresponds to solving LWE. The second, referred to as the *(dual) SIS lattice*, is given by

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{v} = \mathbf{0} \mod q\}. \tag{2.10}$$

Finding a short vector in $\Lambda_q^\perp(\mathbf{A})$ corresponds to solving the Short Integer Solution problem.

In many scenarios, it is convenient to have an explicit formula that descibes the relationship between the determinant of the above two lattices and the dimensions of $\mathbf{A}$. For q prime and m sufficiently larger than n we have that the rank of $\mathbf{A}$ is n as the rows of $\mathbf{A}$ are linearly independent with high probability. As a result, the lattice $\Lambda_q(\mathbf{A}^\intercal)$ has q^n points in $\mathbb{Z}_q^m$. Consider the fundamental domain $D = \mathcal{P}(\Lambda_q(\mathbf{A}^\intercal))$ and the fact that $\Lambda_q(\mathbf{A}^\intercal) + (D \mod q) = \mathbb{R}^m / q\mathbb{R}^m$ is a partition as described in [Mar21]. The volume of $\mathbb{R}^m / q\mathbb{R}^m$ is given by $q^m = |\Lambda_q(\mathbf{A}^\intercal)||D \mod q|$ and thus

$$\det(\Lambda_q(\mathbf{A}^\intercal)) = |D \mod q| = \frac{q^m}{|\Lambda_q(\mathbf{A}^\intercal)|} = \frac{q^m}{q^n} = q^{m-n}. \tag{2.11}$$

Furthermore, we know that the dual lattice $\Lambda_q^\perp(\mathbf{A})$ has q^{m-n} points in $\mathbb{Z}_q^m$ as the dimension of the kernel of $\mathbf{A}$ is $m-n$. Analogously, we obtain

$$\det(\Lambda_q(\mathbf{A}^\top)^\perp) = |D \mod q| = \frac{q^m}{|\Lambda_q(\mathbf{A}^\top)^\perp|} = \frac{q^m}{q^{m-n}} = q^n. \tag{2.12}$$

Another useful tool it the *Gram-Schmidt orthogonalization* . Given a basis $\mathbf{B} = [\mathbf{b}_1 \cdots \mathbf{b}_n] \in \mathbb{Z}_q^{m\times n}$, we write $\pi_{\text{span}(\mathbf{B})}(\mathbf{t})$ for projection of a vector $\mathbf{t}$ onto the span of the vectors in $\mathbf{B}$. Define $\tilde{\mathbf{b}}_i$ as follows: $\tilde{\mathbf{b}}_1 = \mathbf{b}_1$. For $i \in \{2, \ldots, n\}$ let $\tilde{\mathbf{b}}_i$ be the component of $\mathbf{b}_i$ that is orthogonal to the span of $\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}$. In other words,

$$\tilde{\mathbf{b}}_i = \mathbf{b}_i - \pi_{\text{span}(\mathbf{b}_1,\ldots,\mathbf{b}_{i-1})}(\mathbf{b}_i). \tag{2.13}$$

Then, $\tilde{\mathbf{B}} = \left[\tilde{\mathbf{b}}_1 \cdots \tilde{\mathbf{b}}_n\right]$ is called the Gram-Schmidt orthogonalization of the basis $\mathbf{B}$. We define the Gram-schmidt coefficients as follows:

$$\mu_{i,j} = \frac{\langle \tilde{\mathbf{b}}_j, \mathbf{b}_i \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \tag{2.14}$$

We define $\text{dist}(\mathbf{t}, \Lambda(\mathbf{B}))$ where $\Lambda(\mathbf{B}) \subset \mathbb{R}^m$ as the *distance* of a vector $\mathbf{t} \in \mathbb{R}^m$ to the closest lattice vector $\mathbf{v} \in \Lambda(\mathbf{B})$, i.e., $\text{dist}(\mathbf{t}, \Lambda(\mathbf{B})) = \min_{\mathbf{v}\in\Lambda(\mathbf{B})} \|\mathbf{t} - \mathbf{v}\|$.

2.3.1 Lattice Problems

In the previous section, we already mentioned the Shortest Vector Problem or short SVP. In this section, we want to briefly list a number of lattice problems used in this work.

Definition 2.3.2 (SVP$_\gamma$)
Given a basis $\mathbf{B}$ *of a lattice* Λ, *the (approximate) Shortest Vector Problem (SVP$_\gamma$) is the problem of finding a short lattice vector* $v \in \Lambda$ *such that* $0 < \|v\| \leq \gamma\lambda_1(\Lambda)$.

The corresponding decision version is the GapSVP$_\gamma$ problem in which we are asked to decide whether $\lambda_1(\Lambda) \leq 1$ or $\lambda_1(\Lambda) \geq \gamma$ given a basis $\mathbf{B}$ of Λ. If neither is the case, any answer is accepted. In an alternative version of GapSVP$_\gamma$ we have to decide between $\lambda_1(\Lambda) \leq d$ and $\lambda_1(\Lambda) \geq \gamma d$ for some positive real number d [LM09].

Definition 2.3.3 (SIVP$_\gamma$)
Given a basis $\mathbf{B}$ *of a lattice* Λ *of rank n, the (approximate) Shortest Independent Vector Problem (SIVP) is the problem of finding n linearly independent lattice vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \Lambda$ *such that* $\|\mathbf{v}_i\| \leq \gamma \cdot \lambda_n(\Lambda)$ *for all* $i \in \{1, \ldots, n\}$.

Both GapSVP$_\gamma$ and SIVP$_\gamma$ are NP-hard for any constant approximation factor γ [BS99; Kho05]. In Chapter 3, we will look at some selected algorithms that rely on reductions to the following problems.

2.4 Discrete Gaussian Distribution

Oftentimes in lattice cryptography, we work with samples or vectors drawn from a discrete uniform or Gaussian distribution. We can define a discrete Gaussian distribution as follows:

Definition 2.4.1 (Discrete Gaussian Distribution [GJS15])
The discrete Gaussian distribution $D_{\Lambda,s,\mathbf{c}}$ over an m-dimensional lattice Λ with width parameter $s > 0$ and center $\mathbf{c}$ is the probability distribution we obtain by assigning each vector $\mathbf{x} \in \Lambda$ a probability proportional to $e^{-\pi\|\mathbf{x}-\mathbf{c}\|^2/s^2}$. If $\mathbf{c} = \mathbf{0}$ we simply write $D_{\Lambda,s}$.

A discrete Gaussian sampler over a lattice can be efficiently realized (see for example [GPV08]). In order to avoid confusion, throughout this work we use σ to denote the standard deviation, where $\sigma = \frac{s}{\sqrt{2\pi}}$, and define $\alpha = \frac{s}{q} = \frac{\sqrt{2\pi}\sigma}{q}$.

2.5 LWE and SIS

In Chapter 1 we informally introduced the Learning with Errors (LWE) problem and the Short Integer Solution (SIS) problem that constitute the main focus of this work. Both problems have given rise to a great variety of cryptosystems, in particular, in the context of the imminent advent of quantum computing. Hence, their importance in modern cryptography cannot be understated. We will now continue to describe them in more detail.

2.5.1 Learning with Errors (LWE)

The *Learning with Errors* (LWE) problem asks us to recover some secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ from a sequence of perturbed random linear equations on $\mathbf{s}$. The perturbed linear equations, also called samples, are of form $\langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \mod q$, where $\mathbf{a}_i \in \mathbb{Z}_q^n$ are e_i are an error term. We obtain a total of m samples and can thus also express the equation system that is to be solved as $\mathbf{z} = \mathcal{A}^\intercal \mathbf{s} + \mathbf{e} \mod q$, where $\mathcal{A} \in \mathbb{Z}^{n \times m}$ and $\mathbf{e} \in \mathbb{Z}^m$. Note that we only know $\mathbf{z}$ and $\mathbf{A}$. A formal definition follows.

Definition 2.5.1 (LWE Distribution [Reg10])
Given an integer $n \geq 1$, a modulus $q \geq 2$, an error distribution χ on $\mathbb{Z}_q$, and a fixed secret vector $\mathbf{s}$, let $\mathcal{A}_{\mathbf{s},\chi}$ be the probability distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ by choosing a vector $\mathbf{a}_i \in \mathbb{Z}_q^n$ uniformly at random, $e_i \in \mathbb{Z}_q$ according to χ. $\mathcal{A}_{\mathbf{s},\chi}$ outputs pairs of

$$(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \mod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q. \tag{2.15}$$

Additions are performed in $\mathbb{Z}_q$. In the case of $q = 2$ the LWE problem corresponds to the *Learning Parity with Noise* (LPN) problem. We distinguish between two versions of LWE.

Definition 2.5.2 (Search-LWE$_{n,q,m,\chi}$)
The Search-LWE$_{n,q,m,\chi}$ asks for the recovery of the secret vector $\mathbf{s}$ given m independent samples $(\mathbf{a}_i, z_i) \leftarrow \mathcal{A}_{\mathbf{s},\chi}$

Definition 2.5.3 (Decision-LWE$_{n,q,m,\chi}$)
Given m samples, the Decision-LWE$_{n,q,m,\chi}$ asks to distinguish whether the samples were drawn from $\mathcal{A}_{\mathbf{s},\chi}$ or from a uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Intuitively, Search-LWE is at least as hard as Decision-LWE as a solution to Search-LWE trivially solves Decision-LWE. The other direction, however, also holds true [Reg09] for a prime modulus $q = \text{poly}(n)$. The equivalence of the search and decision versions is convenient when we consider common attacks against LWE, some of which solving Decision-LWE and other Search-LWE (see Chapter 3).

LWE as a Decoding Problem

We request m samples $(\mathbf{a}_1, z_1), \ldots, (\mathbf{a}_m, z_m)$ where $z_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \in \mathbb{Z}_q$ and $\mathbf{s} \in \mathbb{Z}_q^n$. Let $A = [\mathbf{a}_1 \cdots \mathbf{a}_m] \in \mathbb{Z}_q^{n \times m}$, $\mathbf{z} = [z_1, \ldots, z_m]^\intercal$ and $e = [e_1, \ldots, e_n]^\intercal \in \mathbb{Z}_q^n$. Hence, we can formulate LWE as a decoding problem as in [GJS15]:

$$\mathbf{z} = \mathbf{A}^\intercal \mathbf{s} + \mathbf{e} \tag{2.16}$$

with generator matrix $\mathbf{A}$ for a linear code over $\mathbb{Z}_q$ and $\mathbf{z}$ as the received word (see Appendix A.1 for more details about linear codes). Finding the secret vector $\mathbf{z}$ is equivalent to finding the codeword $\mathbf{y} = \mathbf{A}^\intercal \mathbf{s}$ with minimum distance $||\mathbf{y} - \mathbf{z}||$.

We can transform an LWE$_{n,q,m,\chi}$ instance with a secret vector $\mathbf{s}$ chosen according to a uniform distribution into an LWE$_{n,q,m-n,\chi}$ instance with a secret vector $\hat{\mathbf{s}}$ chosen according to the error distribution χ at a loss of n samples as follows [GJS15]: Let $\mathbf{A}_0 = [\mathbf{a}_1 \cdots \mathbf{a}_n]$ where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are the first n columns of $\mathbf{A}$. We introduce new variables $\hat{\mathbf{s}} = \mathbf{A}_0^\intercal \mathbf{s} - [z_1, \ldots, z_n]^\intercal = [e_0, \ldots, e_n]^\intercal$ and $\hat{\mathbf{A}} = \mathbf{A}_0^{-1} \mathbf{A} = [\mathbf{I}\ \hat{\mathbf{a}}_{n+1} \cdots \hat{\mathbf{a}}_m]$ and compute $\hat{\mathbf{z}} = \mathbf{z} - \hat{\mathbf{A}}^\intercal\ [z_1, \ldots, z_n]^\intercal = [\mathbf{0}, \hat{z}_{n+1} \cdots \hat{z}_m]^\intercal$. Our new LWE instance has samples $(\hat{\mathbf{a}}_{n+1}, \hat{z}_{n+1}), \ldots, (\hat{\mathbf{a}}_m, \hat{z}_m)$.

LWE as a BDD Problem

Solving LWE also corresponds to solving the *Bounded Distance Decoding problem* (BDD) in the lattice $\Lambda(\mathbf{A}^\intercal) = \{\mathbf{x} \in \mathbb{Z}_q^m \mid \exists \mathbf{s} \in \mathbb{Z}_q^n : \mathbf{x} = \mathbf{A}^\intercal \mathbf{s} \mod q\}$, where the m columns of $\mathbf{A}$ correspond to the vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$ of m independent LWE samples $(\mathbf{a}_i, z_i) \leftarrow \mathcal{A}_{\mathbf{s},\chi}$ and the components z_i correspond to a perturbed lattice point in $\Lambda(\mathbf{A}^\intercal)$.

Hardness

The most important hardness results for LWE come from [Reg05] and [Pei09]. Regev showed that there exists a polynomial-time quantum reduction from worst-case GapSVP$_\gamma$ and SIVP$_\gamma$ in the ℓ_2-norm to the (average-case) search version of LWE. That means that LWE is quantumly at least as hard as GapSVP$_\gamma$ and SIVP$_\gamma$. A similar classical probabilistic polynomial-time reduction from worst-case GapSVP$_\gamma$ was later proposed by Peikert for a sufficiently large modulus $q \geq 2^n$ in any ℓ_p-norm for $p \geq 2$. The Shortest Vector Problem and its variants are well studied problems that are NP-hard for certain approximation problems (see Section 2.3.1. To this point, it is assumed that no efficient quantum algorithms exist that can solve NP-hard problems. It is thus conjectured that cryptosystems based on the worst-case to average-case reductions from Regev and Peikert are secure, even in the context of quantum computing. The currently best algorithms solve LWE in $2^{O(n)}$ time.

2.5.2 Short Integer Solution (SIS)

The dual problem to LWE is the *Short Integer Solution problem* (SIS). In SIS problem, we are given a set of uniformly random vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$ and asked to find an integer linear combination with small coefficients that sums to zero the zero vector (modulo q). A formal definition follows.

Definition 2.5.4 (SIS Problem, adapted from [LS15])
The problem $SIS_{n,q,m,\beta}$ is defined as follows: Given a uniformly random matrix $\mathbf{A}^{n \times m}$, find a vector $\mathbf{s} \in \mathbb{Z}^m$ such that $\mathbf{A} \cdot \mathbf{s} = \mathbf{0} \mod q$ and $0 < \|\mathbf{s}\| \leq \beta$.

Finding such a vector corresponds to solving SVP_γ in the scaled q-ary dual lattice $\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{Av} = \mathbf{0} \mod q\}$. Note that the problem becomes trivial for sufficiently large bounds β. If $\beta \geq q$ in any ℓ_p-norm we can simply choose $\mathbf{s} = [q, 0, \ldots, 0]^\top$. In some scenarios, the vector $\mathbf{s}$ is restricted to $\mathbb{Z}_q^m$. However, depending on the used norm for the length of $\mathbf{s}$, we can still efficiently find $\mathbf{s}$ with $\|\mathbf{s}\|_p$ by using some standard linear equation solver $\|\mathbf{s}\|_p$, if $\{\mathbf{s} \in \mathbb{Z}_q^m \mid \|\mathbf{s}'\|_p \leq \beta\} = \mathbb{Z}_q^m$. For example, in the ℓ_∞-norm we have that $\beta = q$ is large enough. In Section 4.2 we will define norm bound conversions to be able to express relationships between different ℓ_p-norms.

We obtain similar hardness results as for LWE. Micciancio and Regev show a reduction from SIVP_γ in the worst-case to average-case SIS problem [MR04].

2.5.3 Ring and Module Variants

In practice, the base variants of LWE and SIS are not used due to large key sizes required for secure cryptoschemes. Consider the matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$. We usually require $m \in \Omega(n)$ which gives us at least a quadratic space complexity. We therefore want to decrease our key sizes. A way to do this is by using rings and modules as the underlying structure of LWE and SIS. Explaining the details regarding the ring and module variants of both problems would go beyond the scope of this work. Hence we will only provide a rough intuition following [Reg10] and define the various problem variants of LWE and SIS. All of them are included in our tool.

Consider the following choice for $\mathbf{A}$. We set $n = 2^k$ for some $k > 0$ and choose the column vectors $\mathbf{a}$ of $\mathbf{A}$ in groups of size n. In each group we sample $\mathbf{a}_1 = [a_1, \ldots, a_n]^\top$ from the uniform distirbution over $\mathbb{Z}_q^n$. The remaining columns are rotations $\mathbf{a}_i = [a_i, \ldots, a_n, -a_1, \ldots, -a_{i-1}]^\top$ of the first vector in the group. Hence, a block of $n \times n$ only needs $O(n)$ memory reducing the size of $\mathbf{A}$ by a factor of n. In addition, it is possible to achieve speedups in the operations by means of the Fast Fourier Transformation.

In the ring variant, instead of vectors of the group $\mathbb{Z}_q^n$, the columns of $\mathbf{A}$ are chosen as elements of the ring $\mathbb{Z}_q[x]/\langle x^n + 1\rangle$ which we call $\mathcal{R}_q$. We ensure that the $x^n + 1$ is irreducible over the rationals by letting n be a power of two. The disadvantage of powers of two is that our key size is rather coarse-grained which may result in a larger key size for a secure instance than needed up to a factor of two. Modules present themselves as a solution to this problem. Let $K = \mathbb{Q}(\theta)$ be a number field, where θ is an algebraic number with degree n, as defined in [LS15]. A $\mathcal{R}$-module $\mathcal{M} \subseteq K^d$ with dimension or rank d is a generalization of rings and vector spaces and is closed under addition and multiplication by elements of $\mathcal{R}$. We only consider the module $\mathcal{R}^d$ with ring dimension n and module rank d. For more details on the mathematical background, please see [LS15].

Definition 2.5.5 (RSIS [LS15])
The Ring-SIS problem $RSIS_{n,q,m,\beta}$ is defined as follows: Given $a_1, \ldots, a_m \in \mathcal{R}_q$ chosen independently from the uniform distribution, find $s_1, \ldots, s_m \in \mathcal{R}$ such that $\sum_{i=1}^{m} \mathbf{a}_i \cdot s_i = 0 \mod q$ and $0 < \|\mathbf{s}\| \leq \beta$, where $\mathbf{s} = [s_1, \ldots, s_m]^\intercal \in \mathcal{R}^m$.

We can interpret a ring element $r \in \mathcal{R}$ as an n dimensional vector with coefficients r_i such that $r = \sum_{i=0}^{n-1} r_i x^i$. If we compare RSIS to standard SIS, each a_i in RSIS corresponds to a $n \times n$ block in the standard SIS matrix $\mathbf{A}$. The n columns are obtained by rotation as described above. An $\mathrm{RSIS}_{n,q,m,\beta}$ instance can thus be seen as an $\mathrm{SIS}_{n,q,m\cdot n,\beta}$ instance.

Definition 2.5.6 (MSIS [LS15])
The Module-SIS problem $MSIS_{n,d,q,m,\beta}$ is defined as follows: Given $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathcal{R}_q^d$ chosen independently from the uniform distribution, find $s_1, \ldots, s_n \in \mathcal{R}$ such that $\sum_{i=1}^{m} a_i \cdot s_i = 0 \mod q$ and $0 < \|\mathbf{s}\| \leq \beta$, where $\mathbf{s} = [s_1, \ldots, s_m]^\intercal \in \mathcal{R}^m$.

As for RSIS, we can view the vectors $\mathbf{a}_i$ in MSIS as blocks of $\mathbf{A}$ in SIS. Each vector $\mathbf{a}_i$ has d coefficients and corresponds to a $n \cdot d \times n$ block in $\mathbf{A}$. An $\mathrm{MSIS}_{n,d,q,m,\beta}$ instance thus becomes an $\mathrm{SIS}_{n\cdot d,q,m\cdot n,\beta}$ instance.

For LWE we will only list the definition of the ring and module variant. We are aware that we have not introduced the underlying math and refer the reader to [LS15]. Let us just note here that $\mathcal{R}^\perp$ referes to the dual of $\mathcal{R}$.

Definition 2.5.7 (Ring-LWE (RLWE) Distribution [LS15])
Let χ be the error distribution distribution on $\mathbb{T}_{\mathcal{R}^\perp} = K_\mathbb{R}/\mathcal{R}^\perp$ and $s \in \mathcal{R}^\perp$ be the secret. Then we define $\mathcal{A}_{q,s,\chi}^{(\mathcal{R})}$ as the RLWE distribution on $\mathcal{R}_q \times \mathbb{T}_{\mathcal{R}^\perp}$ obtained by choosing $a \in \mathbb{R}_q$ uniformly at random and an error term $e \in \mathbb{T}_{\mathcal{R}^\perp}$ according to χ, and returning samples $(a, (a \cdot s)/q + e)$.

The search version of $\mathrm{RLWE}_{n,q,m,\chi}$ asks us to find s given m samples from $\mathcal{A}_{q,s,\chi}^{(\mathcal{R})}$ with modulus $q \geq 2$ and n the degree of the polynomial of $\mathcal{R}$. The decision version asks us to distinguish between m samples from $\mathcal{A}_{q,s,\chi}^{(\mathcal{R})}$ and m independent samples from the uniform distirbution over $\mathcal{R}_q \times \mathbb{T}_{\mathcal{R}^\perp}$.

Definition 2.5.8 (Module-LWE (MLWE) Distribution [LS15])
Let χ be the error distribution distribution on $\mathbb{T}_{\mathcal{R}^\perp}$ and $\mathbf{s} \in (\mathcal{R}^\perp)^d$ be the secret vector. Then we define $\mathcal{A}_{q,\mathbf{s},\chi}^{(\mathcal{M})}$ as the MLWE distribution on $(\mathcal{R}_q)^d \times \mathbb{T}_{\mathcal{R}^\perp}$ obtained by choosing $\mathbf{a} \in (\mathbb{R}_q)^d$ uniformly at random and an error term $e \in \mathbb{T}_{\mathcal{R}^\perp}$ according to χ, and returning samples $(\mathbf{a}, \frac{1}{q}\langle \mathbf{a}, \mathbf{s} \rangle + e)$.

Analogously, the search version of $\mathrm{MLWE}_{n,d,q,m,\chi}$ asks us to find $\mathbf{s}$ given m samples from $\mathcal{A}_{q,\mathbf{s},\chi}^{(\mathcal{M})}$ with modulus $q \geq 2$, n the degree of the polynomial of $\mathcal{R}$ and modulus rank d. The decision version asks us to distinguish between m samples from $\mathcal{A}_{q,\mathbf{s},\chi}^{(\mathcal{M})}$ and m independent samples from the uniform distirbution over $\mathcal{R}_q^d \times \mathbb{T}_{\mathcal{R}^\perp}$.

We can construct a matrix $\mathbf{A}$ of the a_i of RLWE and $\mathbf{a}_i$ of MLWE as for standard LWE to formulate the problems as a decoding problem as in Section 2.5.1 and interpret $\text{RLWE}_{n,q,m,\chi}$ and $\text{MLWE}_{n,d,q,m,\chi}$ as instances of $\text{LWE}_{n,q,m\cdot d,\chi'}$ and $\text{RLWE}_{n\cdot d,q,m\cdot n,\chi'}$ respectively.

3 Algorithms and Estimates

3.1 Lattice Basis Reduction

In Section 2.5 we showed how LWE and SIS can be viewed as lattice problems. However, the corresponding lattice basis that we obtain is rather ugly lattice basis with long basis vectors. Solving lattice problems in such a basis is infeasible. Our goal is thus to find a better basis with shorter and more orthogonal basis vectors. A family of algorithms that achieve just this is called lattice reduction algorithms.

First of all, we need to define a measure to evaluate the quality of a given basis. The standard measure in the literature is the (root) Hermite factor.

Definition 3.1.1
Root Hermite Factor [LP11] Given a basis $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ *with* $\|\mathbf{b}_1\| \leq \cdots, \leq \|\mathbf{b}_n\|$*, then an n-dimensional lattice* $\Lambda(\mathbf{B})$ *has root Hermite factor* δ *if*

$$\|\mathbf{b}_1\| \approx \delta^n \det(\Lambda)^{1/n}. \tag{3.1}$$

We will later use a result that follows from the Geometric Series Assumption (GSA) as in [Göp16]. The GSA estimates the length of the Gram-Schmidt vectors $\tilde{\mathbf{b}}_i$ as follows [Sch03]:

$$\|\tilde{\mathbf{b}}_i\| \approx \alpha^{i-1} \|\mathbf{b}_1\|, \tag{3.2}$$

for $0 < \alpha < 1$. If we combine Equation (3.1) with Equation (3.2), we obtain $\|\tilde{\mathbf{b}}_i\| \approx \alpha^{i-1}\delta^n \det(\Lambda)^{1/n}$. Furthermore, we know that $\prod_{i-1}^{n} \|\tilde{\mathbf{b}}_i\| = \det(\Lambda)$ and get

$$\begin{aligned} \prod_{i-1}^{n} \|\tilde{\mathbf{b}}_i\| \approx \prod_{i-1}^{n} \alpha^{i-1}\delta^n \det(\Lambda)^{1/n} &\iff \det(\Lambda) \approx \delta^{2n} \det(\Lambda) \prod_{i-1}^{n} \alpha^{i-1} \\ &\iff \delta^{-2n} \approx \alpha^{\frac{n(n-1)}{2}} \\ &\iff \delta^{-2} \approx \alpha^{(n-1)/n} \end{aligned}$$

Hence, $\alpha \approx \delta^{-2}$ and

$$\|\tilde{\mathbf{b}}_i\| \approx \delta^{-2(i-1)+n} \det(\Lambda)^{1/n} \tag{3.3}$$

We can see that for smaller i the length of the Gram-Schmidt vectors $\tilde{\mathbf{b}}_i$ decreases whereas the length increases for larger i resulting in a long and skinny fundamental parallelipiped $\mathcal{P}_{1/2}(\tilde{\mathbf{B}})$.

In the following, we will focus on two related methods for lattice reduction. First, we define some reduction criterias following [ABLR21]. A basis $\mathbf{B} = [\mathbf{b}_1 \cdots \mathbf{b}_n]$ is *size-reduced* if its Gram-Schmidt coefficients (see Section 2.3) satisfy $|\mu_{i,j}| \leq \frac{1}{2}$ for all $0 \leq j < i < n$. If the first basis vector $\mathbf{b}_1$ of $\mathbf{B}$ is the shortest lattice vector, i.e. $\|\mathbf{b}_1\| = \lambda_1(\Lambda(\mathbf{B}))$, we call $\mathbf{B}$ *SVP-reduced*. If a basis $\mathbf{B}$ is size-reduced and in addition each block $\{\mathbf{b}_i, \dots, \mathbf{b}_n\}$ for $i = 1, \dots, n$ of basis vectors is SVP-reduced then $\mathbf{B}$ is *HKZ-reduced.* We will see in the next section that size-reduction is closely related to the LLL algorithm and a special case of the BKZ reduction outputs an HKZ-reduced basis.

3.1.1 The LLL Algorithm

The LLL algorithm was proposed by Lenstra, Lenstra and Lovász [LLL82] and can be considered as a generalization of the two dimensional Lagrange reduction. The lagrange reduction reduces a basis of two basis vectors such that output basis satisfies $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$ and $\frac{|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle|}{\|\mathbf{b}_1\|} = |\mu_{2,1}| \leq \frac{1}{2}$). Intuitively, a multiple of the shorter vector $\mathbf{b}_1$ is subtracted from the longer vector $\mathbf{b}_2$ such that the resulting vector $\mathbf{b}'_2$ is as orthogonal to $\mathbf{b}_0$ as possible, i.e. $\mathbf{b}'_1 = \mathbf{b}_1 - \lfloor \mu_{1,0} \rceil \mathbf{b}_0$. We set $\mathbf{b}_2 = \mathbf{b}'_2$ and repeat until nothing changes. Note that δ here does not refer to the root Hermite factor.

A δ-LLL reduced basis ensures two criterias [LLL82]:

1. Size reduced: $|\mu_{i,j}| \leq \frac{1}{2}$ for $1 \leq i \leq n$ and $j < i$
2. Lovász condition: $\delta \|\tilde{\mathbf{b}}_i\|^2 > \|\mu_{i+1,i} \tilde{\mathbf{b}}_i + \tilde{\mathbf{b}}_{i+1}\|^2$ for $1 \leq i < n$

Recall the definition of the Gram-Schmidt coefficients $\mu_{i,j} = \frac{\langle \tilde{\mathbf{b}}_j, \mathbf{b}_i \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle}$. The LLL algorithm shown in Algorithm 1 follows the notation in [Reg04a]. We start by computing the Gram-Schmidt orthogonalization of the input basis (Line 2) and continue with a reduction step in which we update every basis vector $\mathbf{b}_i$ by pairwisely comparing and subtracting lower indexed basis vector just as in the Lagrange reduction (Line 5) to ensure Criteria 1. Finally, vectors violating the Lovász condition are swapped (Line 7ff) and the process is repeated until nothing changes. The LLL algorithm can be used to find short vectors of at most $2^{n/2}\lambda_1(\Lambda)$ in polynomial time. Several floating-point variants have been suggested that can significantly speed up the runtime of LLL. For example, L^2 runs in $O(n^2 \log^2 B)$, where B is a bound on the norm of the input basis vectors [NS05].

Algorithm 1: The δ-LLL algorithm [LLL82]

```
1  function $\delta$-LLL($\mathbf{B} \in \mathbb{Z}^{m \times n}$)
2      Compute $\tilde{\mathbf{B}}$
3      for $i = 2, \dots, n$ do
4          for $j = i - 1, \dots, 1$ do
5              $\mathbf{b}_i = \mathbf{b}_i - \lfloor \mu_{i,j} \rceil \mathbf{b}_j$
6      if $\exists i$ such that $\delta \|\tilde{\mathbf{b}}_i\|^2 > \|\mu_{i+1,i} \tilde{\mathbf{b}}_i + \tilde{\mathbf{b}}_{i+1}\|^2$ then
7          Swap $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$
8          Return $\delta$-LLL($\mathbf{B}$)
9      else
10         Return $\mathbf{B}$
```

The proven upper bound of the output quality of the LLL algorithm is $\delta^n = (4/3)^{\frac{n-1}{4}}$ or $\delta \approx 1.075$ [LLL82]. In practice, on average we get much better results of $\delta \approx 1.021$ [Che13]. Given an input basis in which the length of all basis vectors is bounded by B, the runtime of LLL is proven to be in $O\left(n^4 m \log B(n + \log B)\right)$ [NS05] and heuristically known to be $O\left(n^3 \log^2 B\right)$ [APS15].

3.1.2 The BKZ Algorithm

The Block Korkin-Zolotarev (BKZ) algorithm was proposed by Schnorr in 1987 and adapted by Schnorr and Euchner in [SE91] and represents a family of lattice reduction algorithm. Essentially, BKZ iteratively divides the input basis into blocks of a lower dimension k and calling an SVP oracle on each block. The output of the oracle is then used to obtain a basis of improved quality.

Algorithm 2 presents the main concept of BKZ and follows the description in [CN11] with some adjustments. Initially, we run an LLL reduction on the input basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ and update the basis. In each jth iteration, we consider a block of k basis vectors $\mathbf{b}_j, \ldots, \mathbf{b}_{j+k-1}$. The vectors of the current block are projected onto the orthogonal complement of the span of vectors from previous iterations span $(\{\mathbf{b}_i \mid i \in [j-1]\})$ (Line 6 - 9, we skip this step if the span is empty). Note that the orthogonal complement $A^\perp$ of a subspace A is defined as the set of all vectors that are orthogonal to every vector in A. We then run an SVP oracle on the projected block to obtain a shortest vector $\mathbf{b}'_{\text{new}}$ in the projected lattice (Line 12) and reconstruct a lattice vector $\mathbf{b}_{\text{new}}$ of which $\mathbf{b}'_{\text{new}}$ is a projection Line 13. Note that in practice, the SVP oracle should include this step. If $\mathbf{b}_{\text{new}}$ is a new vector we insert it in our list of basis vectors before $\mathbf{b}_j$. Otherwise as nothing changed, we increment a counter z. Finally, we run LLL on all basis vectors up to index $j + i$ (including the possibly newly added vector). If no new lattice vectors can be found in n iterations, the reduction terminates. After n iterations, j is reset to start over at the first block. The ouput of the algorithm is a BKZ$_k$-reduced basis. For $k = 2$ we obtain an LLL-reduced basis in polynomial time and for $k = n$ an optimally HKZ-reduced basis in at least exponential time.

Several improvements have been suggested. The total number of rounds until termination is unknown and can be quite large. Hanrot *et al.* [HPS11] show an *early termination* of BKZ still yields a very good output basis quality and propose $\frac{n^2}{k^2} \log n$ rounds as a bound.

Local preprocessing increases the quality of the current block basis by recursively calling BKZ with smaller block size. A variant known as *progressive BKZ* applies the recursion globally [AWHT16].

If enumeration is used as an SVP oracle, the size of the search space can be reduced by means of *pruning* techniques. Nodes closer to the edges of the enumeration tree are less likely to represent in short lattice vectors. For more details on enumeration, we refer to Section 3.1.3. Gamma *et al.* show that applying a variant of this known as *extreme pruning* can reduce the running time by a much larger factor than the success probability. Repeating the search yields the desired speedup [GNR10].

In addition, [CN11] optimizes the *enumeration radius* by using experimental results. BKZ 2.0 incorporates a number of these techniques [CN11].

Algorithm 2: The BKZ algorithm [SE91]

```
function BKZ($\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}, k \in [n] \setminus \{1\}$)
    z = 0; j = 0
    $\mathbf{B}$ = LLL($\mathbf{B}$)
    while $z < n - 1$ do
        $j = (j \mod (n-1)) + 1; l = \min(j + k - 1, n); h = \min(l + 1, n)$
        $A = \text{span}(\{\mathbf{b}_i | i \in [j-1]\})$
        for $i \in \{j, ..., l\}$ do
            if $A \neq \emptyset$ then
                $\mathbf{b}'_i = \pi_{A^\perp}(\mathbf{b_i})$
            else
                $\mathbf{b}'_i = \mathbf{b}_i$
        $\mathbf{b}'_{\text{new}}$ = SVP-Oracle($\mathbf{b}'_j, \dots, \mathbf{b}'_l$)
        Reconstruct $\mathbf{b}_{\text{new}} = \sum_{i=j}^{l} \alpha_i \mathbf{b}_i$ with $\alpha_i \in \mathbb{Z}$ such that $\mathbf{b}'_{\text{new}} = \pi_{(\text{span}(\mathbf{b}_j,\dots,\mathbf{b}_l))^\perp}(\mathbf{b}_{\text{new}})$
        if $\mathbf{b}'_{new} \neq \tilde{\mathbf{b}}_j$ then
            $z = 0; \{\mathbf{b}_j, \dots, \mathbf{b}_h\} = \text{LLL}(\{\mathbf{b}_j, \dots, \mathbf{b}_{j-1}, \mathbf{b}_{\text{new}}, \mathbf{b}_j, \dots, \mathbf{b}_h\})$
        else
            $z = z + 1; \{\mathbf{b}_j, \dots, \mathbf{b}_h\} = \text{LLL}(\{\mathbf{b}_j, \dots, \mathbf{b}_h\})$
```

It is difficult to find hard runtime bounds for BKZ. The upper bound on the number of rounds is superexponential in the dimension n for a fixed block size [GN08; HPS11] before BKZ terminates given that no early termination strategy is used. In addition, calls to the SVP oracle in all dimensions $k' \leq k$ must be taken into account. Albrecht *et al.* ignore these intricacies and estimate the cost of BKZ in clock cycles as $\rho \cdot n \cdot t_k$ where ρ is the number of rounds needed and t_k is the cost (in block cycles) of calling the SVP oracle on a block of dimension k Albrecht et al. [APS15]. The value ρ is set to 8 in the *estimator* and is derived from experiments in [Che13] that indicate that the most significant progress happens in the first $7 - 9$ rounds.

The output quality of BKZ is closely related to the used block size k. The *estimator* uses a limiting value from [Che13] to estimate the root Hermite factor δ for a given k:

$$\lim_{n \to \infty} \delta \approx \left(\frac{k(\pi k)^{\frac{1}{k}}}{2\pi e} \right)^{\frac{1}{2(k-1)}} \tag{3.4}$$

For smaller block sizes $k \leq 40$ the *estimator* uses fixed experimental values for δ.

3.1.3 Cost Models for Lattice Reduction

In this section, we will look at various high level ideas to realize an SVP solver that can be used as a subroutine in BKZ and present up-to-date cost models from the literature. SVP is known to be NP-complete even for large constant approximation factors [Ajt98; Kho05]. An exponential approximation factor can be achieved in polynomial time but is mostly insufficient for practical purposes [LLL82]. We will mainly focus on two classes of (nearly) exact SVP solvers, namely, enumeration algorithms and sieving algorithms. Enumeration algorithms can solve SVP in a lattice of dimension k in $2^{O(k \log k)}$ time and polynomial space. Sieving algorithms only need $2^{O(k)}$ time, however, at the cost of exponential memory complexity. Only recently, progress in sieving strategies has given rise to BKZ implementations relying on sieving (e.g. the General Sieve Kernel (G6K) impelmentation [ADH+19; DSW21]) that outperform enumeration implementations already in relatively small dimensions $\gtrsim 70$ in the classical setting [ABLR21]. On the other hand, quantum speedups for enumeration are greater than for sieving. Aono *et al.* show a quadratic cost reduction for enumeration [ANS18], while the cost sieving only decreases by a factor of $2^{0.027}$ with idealized assumptions [Laa16]. The authors of [ADPS16] argue that due of a lower bound $2^{0.2075k}$ on the required size of the building lists future quantum sieving algorithms are not expected to achieve an asymptotic runtime below $2^{0.2075k}$.

A selection of the most relevant cost models for cryptographic purposes is shown Table 3.2. All these cost models are supported in our tool.

Enumeration

Enumeration aims to find the shortest vector by enumerating all lattice vectors within some bounded region. In general, we start with reducing the lattice basis to improve the basis quality. We then define a bound and iteratively project the lattice to the span of its Gram-Schmidt vectors beginning from $\tilde{\mathbf{b}}_n$ until we arrive at the lowest level of a one-dimensional subspace. We continue by enumerating all vectors of norm less than r in the projected lattice and "lift" each of these vectors to the level above and repeat this process until we arive at the level from which we started. The search space can be thought of as a large tree of (projected) vectors on which we apply depth-first search. Note that the root of the tree here is at the lowest level and the leafs are the lattice vectors in our target lattice. The low memory cost of enumeration is due to its similarities to depth-first search.

A very early but very efficient variant was suggested by Kannan [Kan83] with a proven worst-case runtime of $2^{O(k \log k)}$. BKZ$_k$ using Kannan's enumeration algorithm as SVP oracle yields a short lattice vector of norm approximately $\left(k^{1/(2k)}\right)^n \cdot \mathrm{Vol}(\Lambda)^{1/n}$ or equivalently achieves a root Hermite factor of $k^{1/(2k)}$ [ABF+20; HS07].

In [ABF+20] we find a more concrete experimental cost model of $\mathrm{poly}(n) \cdot 2^{(k \log k)/(2e) - 0.995k + 16.25}$ for BKZ 2.0 (see Section 3.1.2), where $\mathrm{poly}(n)$ is the number of calls to the enumeration subroutine. BKZ 2.0 achieves a root Hermite factor of $\left(k/(2\pi e) \cdot (\pi k)^{1/k}\right)^{1/(2(k-1))}$ [Che13].

The FastEnum algorithm in Albrecht et al. [ABF+20] incorporates an idea called "extended preprocessing" and simulations achieve a root Hermite factor of $k^{(1+o(1))/(2k)}$ in $\mathrm{poly}(n) \cdot 2^{0.125k \log k - 0.050k + 56}$ time. The corresponding quantum algorithm reduces the runtime from $2^{(k \log k)/8 + o(k)}$ to $2^{(k \log k)/16 + o(k)}$. In extended preprocessing, instead of preprocessing the current projected basis block of size k, the BKZ-reduction is applied to a block of higher dimension $\lceil (1+c) \cdot k \rceil$ for some constant c. Enumeration is faster on the first basis vectors as their Gram-Schmidt norms closely follow the Geometric Series Assumption [MW16].

A tradeoff of runtime and success probability for "relaxing" the approximation and extreme pruning turns out to exponentially speed up the search [LN20] and was combined with extended preprocessing by Albrecht *et al.* to further reduce the experimental runtime of BKZ to $\mathrm{poly}(n) \cdot 2^{(k \log k)/8-0.654k+25.84}$ for a root Hermite factor of $k^{1/(2k)}$ [ABF+20].

Sieving

The second group of SVP solvers are sieving algorithms [ADH+19; BDGL16; BGJ15; BLS16; HK17; MV10; NV08]. In sieving, initially, we create a long list of randomly selected lattice points. The points in the list are then combined or "reduced" in some way to find points of smaller length. One way to achieve this is by finding a minimimal sublist of "center" points in the initial list such that spheres centered at these points cover all list points. Subtracting the center points yields short lattice points. ListSieve [MV10] uses a smaller initial list to divide the space into two half-spaces, one closer to the center and one closer to the respective point. The list is then used to reduce the length of newly sampled points as much as possible by subtracting each list vectors such that the result is located in the half-space closer to the center respectively. Once two points with a distance less than the target distance are found, they are subtracted and the result is returned.

Algorithm	Runtime complexity	Memory complexity
List Sieve [MV10]	$2^{0.3199n+o(n)}$	$2^{0.1325+o(n)}$
NV-sieve [ADH+19; NV08]	$2^{0.415n+o(n)}$	$2^{0.2075n+o(n)}$
NV-sieve (quantum) [ADH+19; NV08]	$2^{0.311n+o(n)}$	$2^{0.2075n+o(n)}$
Gauss sieve [HK17; MV10]	$2^{0.415n+o(n)}$	$2^{0.2075n+o(n)}$
BGJ-sieve [BGJ15]	$2^{0.311n+o(n)}$	$2^{0.2075n+o(n)}$
3-sieve [BLS16; HK17]	$2^{0.3962n+o(n)}$	$2^{0.1887n+o(n)}$
BDGL-sieve [BDGL16]	$2^{0.292n+o(n)}$	$2^{0.2075n+o(n)}$
BDGL-sieve (quantum) [BDGL16]	$2^{0.265n+o(n)}$	$2^{0.2075n+o(n)}$

Table 3.1: Overview of popular sieving algorithms

Table 3.1 presents a list of currently best sieving algorithms. Note that some runtime and space complexities are only conjectured and not proven yet.

In the Nguyen-Vidick sieve [NV08], we iteratively reduce a pair of list points whose combined length is smaller than the longest list vector. The longest vector is then replaced by the result. The list length is fixed. In the Gauss sieve [MV10], we start with an empty list and a stack. In each step, a new point is either sampled or taken from the stack. We then attempt to reduce the new point with all points in the list. If a reduction is successful, the longer vector of the pair is replaced. If the longer vector was the list point, the replacement is inserted in the stack. If no reduction is possible, the stack points are moved back to the list. If the stack is empty, all list points are reduced pairwisely. In practice, the Gauss sieve outperforms Nguyen-Vidick sieve. The Becker-Gama-Joux sieve [BGJ15] exploits coding theory to find vectors that are likely to be nearest neighbors. Similar vectors are stored in the same bucket to speed up the search for reduction candidates. The 3-sieve [BLS16; HK17] reduces the required list size by using triples instead of pairs of points for combination. Finally, the Becker-Ducas-Gama-Laarhoven sieve [BDGL16] applies locality sensitive hashing to create buckets of points in near neighborhood similar to the Becker-Gama-Joux sieve.

Name	Reference	Cost model
Sieving		
Q-Sieve (paranoid lower bound)	[ADPS16]	$2^{0.2075k}$
Q-Sieve	[ADPS16; AGPS20; Laa16]	$2^{0.265k}$
Q-Sieve + O(1)	[SAL+17]	$2^{0.265k+16}$
Q-Sieve (min space)	[SHRS17]	$2^{0.2975k}$
Sieve	[ADPS16; AGPS20; BDGL16]	$2^{0.292k}$
Sieve + O(1)	[SAL+17]	$2^{0.292k+16}$
Sieve (min space)	[SHRS17]	$2^{0.368k}$
Enumeration		
Lotus	[ACD+18; PHAM17]	$2^{0.125k \log k - 0.755k + 2.254}$
Enum + O(1)	[ACD+18; Che13; SHRS17]	$2^{0.187k \log k - 1.019k + 16.1}$
Q-Enum + O(1)	[ACD+18; Che13; SHRS17]	$2^{0.0936k \log k - 0.51k + 8.05}$
BCLV-Enum (quadratic fit) + O(1)	[BCLV17]	$2^{0.000784k^2 + 0.366k + 0.875}$
BKZ2.0-Enum	[ABF+20; Che13; CN11]	$2^{0.184k \log k - 0.995k + 16.25}$
ABF-Enum	[ABF+20]	$2^{0.125k \log k}$
ABF-Enum + O(1)	[ABF+20]	$2^{0.125k \log k - 0.547k + 10.4}$
Q-ABF-Enum	[ABF+20]	$2^{0.0625k \log k}$
ABLR-Enum + O(1)	[ABLR21]	$2^{0.125k \log k - 0.654k + 25.84}$

Table 3.2: SVP cost models overview (based on [ACD+18, Table 4])

3.2 Algorithms for Solving LWE

In this section, we present a number of popular attacks on LWE. The *estimator* includes cost estimate algorithms for each of these attacks. In our treatise, we will ignore special cases of LWE that can be exploited to obtain more efficient attack variants and only present the main algorithms. For more details we refer the reader to [APS15] and [BBGS19].

3.2.1 Lattice Problems

Some of the following attacks rely on a reduction of LWE to different lattice problems. We first define three important lattice problems in the context of LWE.

Definition 3.2.1 (HSVP$_\delta$)
Given a basis $\mathbf{B}$ *of a lattice* $\Lambda(\mathbf{B}) \in \mathbb{R}^m$, *the (approximate) Hermite Shortest Vector Problem (HSVP$_\delta$) is the problem of finding a nonzero lattice vector* $\mathbf{v} \in \Lambda$ *such that* $\|\mathbf{v}\| \leq \delta \cdot det(\Lambda)^{\frac{1}{n}}$.

Definition 3.2.2 (uSVP$_\gamma$)
Given a lattice Λ *such that* $\lambda_2(\Lambda) > \gamma\lambda_1(\Lambda)$, *the (approximate) Unique Shortest Vector Problem (uSVP$_\gamma$) is the problem of finding the shortest nonzero vector in* $\mathbf{v} \in \Lambda$ *with* $\|\mathbf{v}\| = \lambda_1(\Lambda)$.

Definition 3.2.3 (BDD$_\gamma$)
Given a lattice $\Lambda \subset \mathbb{R}^m$ and a target vector $\mathbf{t} \in \mathbb{R}^m$ such that dist$(\mathbf{t}, \Lambda) < \gamma\lambda_1(\Lambda)$, the (approximate) Bounded Distance Decoding (BDD$_\gamma$) is the problem of finding the closest lattice vector $\mathbf{v} \in \Lambda$, i.e. $\mathbf{v} = \arg\min_{\mathbf{v}' \in \Lambda} \|\mathbf{v}' - \mathbf{t}\|$.

3.2.2 Approaches

Figure 3.1 gives an overview of the three main approaches to solving LWE. The most natural approach is to recover the secret in LWE directly, e.g., by exhaustively searching the search space. In general, however, the search space is quite large resulting in high attack costs. The *estimator* incorporate two direct attacks, the Meet-in-the-Middle attack [BG14] and an attack due to Arora and Ge [AG11].

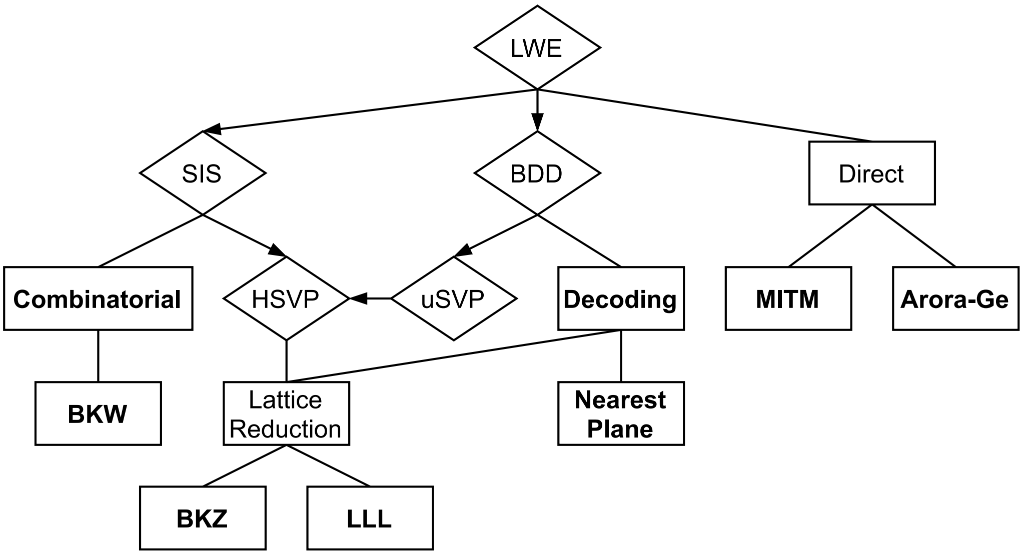

Figure 3.1: An overview of the problems and algorithms used in this work. The diamond shaped boxes represent problems and arrows reductions between problems. Rectangular boxes represent approaches, classes of algorithms or algorithms. A connection between a problem and a rectangular box indicates that the latter can be used as a solver for the respective problem. Bold names indicate algorithms that can be used in our tool.

Primal Attack

The second approach aims to solve LWE in the "primal" LWE lattice by a reduction to the BDD problem. Consider an LWE instance with samples $\mathbf{A}, \mathbf{z}$ and the corresponding lattice $\Lambda_q(\mathbf{A}^\top)$. We know that for the secret vector $\mathbf{s}$ and error vector $\mathbf{e}$ it holds that $\mathbf{z} = \mathbf{A}^\top\mathbf{s} + \mathbf{e} \mod q = \mathbf{A}^\top\mathbf{s} + \mathbf{e} + q\mathbf{x}$ for some vector $\mathbf{x} \in \mathbb{Z}^m$. Obviously, $\mathbf{A}^\top\mathbf{s} + q\mathbf{x}$ is a lattice vector in the lattice $\Lambda_q(\mathbf{A}^\top)$. We know that dist$(\mathbf{z}, \Lambda_q(\mathbf{A}^\top) = \|\mathbf{e}\|$ and in general it holds that $\|\mathbf{e}\| < \gamma\lambda_1(\Lambda_q(\mathbf{A}^\top))$. We thus have a reduction from LWE to BDD. By finding closest lattice point to $\mathbf{z}$ or, in other words, solving BDD in the lattice $\Lambda_q(\mathbf{A}^\top)$ we obtain $\mathbf{A}^\top\mathbf{s} + q\mathbf{x}$ and can recover the secret $\mathbf{s}$.

Dual Attack

Finally, we can take advantage of the close relationship of LWE and SIS and solve LWE by finding a short vector in the dual SIS lattice [LP11]. More precicely, we want to find a short nonzero vector $\mathbf{v} \in \mathbb{Z}_q^m$ in the scaled q-ary lattice $\Lambda_q(\mathbf{A}^\top)^\perp = \{\mathbf{y} \in \mathbb{Z}^m \mid \mathbf{Ay} = \mathbf{0} \mod q\}$. Obviously, we have that $\langle \mathbf{v}, \mathbf{z} \rangle = \langle \mathbf{v}, \mathbf{A}^\top \mathbf{s} \rangle = \langle \mathbf{v}, \mathbf{e} \rangle \mod q$ and $\langle \mathbf{v}, \mathbf{z} \rangle$ roughly corresponds to a sample drawn from a Gaussian with parameter $s' = \|\mathbf{v}\| \cdot s$ given that $\mathbf{e}$ is distributed according to a Gaussian with parameter s. We then test whether $\langle \mathbf{v}, \mathbf{z} \rangle \mod q$ approximates the Gaussian. If $\mathbf{z}$ is drawn from a uniform distribution instead, the test accepts exactly with probability 0.5. If s' is not much larger than q, the advantage of distinguishing uniform from LWE samples is very close to $\exp(-\pi(\|\mathbf{v}\| s/q)^2)$. An optimal attack cost can be achieved by balancing the advantage with the computational effort required to solve SVP_γ on the SIS lattice, where $\gamma = \|\mathbf{v}\|$.

3.2.3 Decoding Attack [LP11]

The decoding attack falls into the regime of primal attacks that solve LWE by solving BDD in the primal LWE lattice (as described in Section 3.2.2). In the decoding attack, we first apply a reduction algorithm to the input basis $\mathbf{A}^\top$ to improve the basis quality. We then enumerate a number of candidate lattice points close to our target $\mathbf{z}$ by running a variant of Babai's Nearest Planes algorithm and simply choose the closest point.

We begin with the original Nearest Planes algorithm due to Babai [Bab85] as shown in Algorithm 3. Our algorithm follows the notation in[Reg04b]. The goal of the algorithm is to find a lattice vector relatively close to target vector. The procedure used is similar to the procedure in the inner loop of LLL. We first project $\mathbf{t}$ to the span of all basis vectors $\mathrm{span}(\mathbf{B})$, where B denotes an LLL-reduced basis, to eliminate irrelevant dimensions and obtain a projected vector which we call $\mathbf{b}$. We then iterate over $i = n, \ldots, 1$. In each step, we want to find the closest hyperplane $c_i \tilde{\mathbf{b}}_i + \mathrm{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ to the projected vector $\mathbf{b}$. We can compute c_i by a simple projection of b onto the Gram-Schmidt vector $\tilde{\mathbf{b}}_i$ and dividing by the square length of dividing by the length of $\tilde{\mathbf{b}}_i$. We subtract $c_i \mathbf{b}_i$ from b and replace $\mathbf{b}$ with the result. Notice that in the last step the hyperplane is simply a point. We return $\mathbf{t} - \mathbf{b}$. If we indeed found the closest lattice vector for target $\mathbf{z}$, the vector $\mathbf{b}$ will be our error vector $\mathbf{e}$ and $\mathbf{t} - \mathbf{b}$ will be the desired lattice vector $\mathbf{A}^\top \mathbf{s} + q\mathbf{x}$ from which we can recover our secret.

Algorithm 3: Babai's Nearest Planes algorithm [Bab85]

```
function NearestPlanes(LLL-reduced basis $\mathbf{B} \in \mathbb{R}^{m \times n}$, target $\mathbf{t} \in \mathbb{R}^m$)
    $\mathbf{b} = \pi_{\mathrm{span}(\mathbf{B})}(\mathbf{t})$
    for $i = n, \ldots, 1$ do
        $c_i = \mathrm{round}(\langle \mathbf{b}, \tilde{\mathbf{b}}_i \rangle / \langle \tilde{\mathbf{b}}_i, \tilde{\mathbf{b}}_i \rangle)$
        $\mathbf{b} = \mathbf{b} - c_i \mathbf{b}_i$
    output $\mathbf{t} - \mathbf{b}$
```

In general, however, the output is a lattice vector $\mathbf{v} \in \Lambda(\mathbf{B})$ in the fundamental parallelipiped of the Gram-Schmidt basis $\mathcal{P}_{1/2}(\tilde{\mathbf{B}})$ and we only get the guarantee of $\|\mathbf{v} - \mathbf{t}\| \leq 2^{n/2} \mathrm{dist}(\mathbf{t}, \Lambda(\mathbf{B}))$. To apply the algorithm on LWE either $\mathbf{e}$ must be in $\mathcal{P}_{1/2}(\tilde{\mathbf{B}})$ or we must increase our search radius.

The former case is unlikely as the last Gram-Schmidt vectors in a reduced basis because of the long and skinny structure of $\mathcal{P}_{1/2}(\tilde{\mathbf{B}})$ (see Section 3.1. The latter case is exactly what the generalized algorithm by [LP11] does.

Instead of choosing only the nearest plane in each iteration step, the Lindner and Peikert's variant that can be seen in Algorithm 4 selects a variable amount d_k of distinct planes in each step. As a consequence, the fundamental parallelipiped of the Gram-Schmidt basis is stretched in the direction of $\tilde{\mathbf{b}}_k$. The values of $\mathbf{d}$ should be chosen such that the covered area is approximately the same in each direction (i.e. by maximizing $\min_i(d_i \cdot \|\tilde{\mathbf{b}}_i\|)$). In particular this implies that the d_k are larger for larger k as the Gram-Schmidt vectors have a smaller length. Compared to Algorithm 3 the runtime increases by a factor $\prod_{i\in\{1,\ldots,d_k\}} d_i$, however, the recursion step can be fully parallelized.

Algorithm 4: Generalized Nearest Planes algorithm [LP11]

```
function GeneralizedNearestPlanes($\mathbf{B} \in \mathbb{R}^{m\times k}, \mathbf{t} \in \mathbb{R}^m, \mathbf{d} \in (\mathbb{Z}^+)^k$)
    if $k = 0$ then
        Return $\mathbf{0}$
    else
        Compute projection $\mathbf{v}$ of $\mathbf{t}$ onto span($\mathbf{B}$)
        Compute the $d_k$ distinct integers $c_1, \ldots, c_{d_k}$ closest to $\langle \mathbf{v}, \tilde{\mathbf{b}}_k\rangle / \langle \tilde{\mathbf{b}}_k, \tilde{\mathbf{b}}_k\rangle)$
        Return $\bigcup_{i\in\{1,\ldots,d_k\}}(c_i \cdot \mathbf{b}_k +$
          GeneralizedNearestPlanes($\{\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}\}, (d_1, \ldots, d_{k-1}), \mathbf{v} - c_i \cdot \mathbf{b}_k$))
```

It is evident that a lower quality of the reduced input basis can be compensated for by increasing the values of $\mathbf{d}$. The input parameters of the lattice reduction and the generalized Nearest Planes algorithm should hence be adjusted such that the overall runtime is minimized. Success probability of the Generalized Nearest Planes algorithm:

$$\Pr\left[\mathbf{e} \in \mathcal{P}_{\frac{1}{2}}(\tilde{\mathbf{B}} \cdot \operatorname{diag}(\mathbf{d}))\right] = \prod_{i=1}^{m-1} \operatorname{erf}\left(\frac{d_i \|\tilde{\mathbf{b}}_i\| \sqrt{\pi}}{2\alpha q}\right) \tag{3.5}$$

3.2.4 Primal-uSVP [ADPS16; BG14]

In the Primal uSVP attack, we again view the $\mathrm{LWE}_{n,q,m,\chi}$ instance $(\mathbf{A}, \mathbf{z})$ as a BDD instance in the q-ary lattice $\Lambda(\mathbf{A}^\top) = \{\mathbf{y} \mid \exists \mathbf{x} \in \mathbb{Z}_q^n : \mathbf{y} = \mathbf{A}^\top\mathbf{x} \mod q\}$ Section 2.5.1 generated by rows of LWE instance. The target vector is $\mathbf{z}$.

Recall the γ-uSVP problem. Given a lattice Λ where $\lambda_2(\Lambda) > \gamma\lambda_1(\Lambda)$, we are asked to find shortest nonzero vector in Λ. In the primal attack, instead of directly solving BDD, we reduce BDD to uSVP, i.e., we reduce a BDD instance to a γ-uSVP instance. By solving γ-uSVP, we obtain a solution to BDD. To do this we apply Kannan's embedding technique [Kan87]. Kannan's embedding creates a lattice with uSVP structure. We know that $\mathbf{A}^\top\mathbf{s} \mod q$ is the closest vector to the target $\mathbf{z} = \mathbf{A}^\top\mathbf{s} + \mathbf{e}^\top \mod q$ in $\Lambda(\mathbf{A}^\top)$. We now add a linearly independent basis vector $(\mathbf{z}, \mu)$ and append a zero coefficient to each basis vector of the original lattice (i.e. the rows of $\mathbf{A}$). Thereby, we ensure that the new lattice contains the vector $[-\mathbf{e}, -\mu]^\top$ as $[\mathbf{A} \mid \mathbf{0}]^\top\mathbf{s} - 1 \cdot [\mathbf{z}^\top, \mu] = [-\mathbf{e}, -\mu]^\top$.

More formally, let $\mathbf{B} \in \mathbb{Z}^{m \times m}$ be the basis of an m dimensional lattice derived from the LWE instance and $\mu = \operatorname{dist}(\mathbf{z}, \Lambda(\mathbf{B})) = \|\mathbf{z} - \mathbf{s}\|$ be the embedding factor, where $\mathbf{s}$ is the secret vector of the LWE instance. For more details on how to compute $\mathbf{B}$, we refer to [AFG13].

We now embed $\Lambda(\mathbf{B})$ into $\Lambda(\mathbf{B}')$ with γ-uSVP structure as follows:

$$\mathbf{B}' = \begin{pmatrix} \mathbf{B} & \mathbf{z} \\ \mathbf{0}^\top & \mu \end{pmatrix} \tag{3.6}$$

If $\gamma \geq 1$ and $\mu < \frac{\lambda_1(\Lambda(\mathbf{B})}{2\gamma}$ (or equivalently, $(\Lambda(\mathbf{A}^\top), \mathbf{z})$ a $\mathrm{BDD}_{1/(2\gamma)}$-instance), then $\Lambda(\mathbf{B}')$ contains a γ-unique shortest vector $\mathbf{z}' = [(\mathbf{A}^\top\mathbf{s} - \mathbf{z})^\top, -\mu]^\top = [-\mathbf{e}^\top, -\mu]^\top$. The statement can be proven by showing by contradiction that all vectors $\mathbf{v} \in \Lambda(\mathbf{B}')$ that are independent of $\mathbf{z}'$ satisfy $\|\mathbf{v}\| \geq \lambda_1\Lambda(\mathbf{B})/\sqrt{2} > \sqrt{2}\gamma\mu = \gamma\|\mathbf{z}'\|$ (see [LM09, Section 4] for more details). Note that the reduction can be done in polynomial time [LM09, Theorem 4.1]. The length of $\mathbf{z}'$ is given by $\|\mathbf{z}'\| = \sqrt{\|\mathbf{e}\|^2 + |\mu|^2} = \sqrt{m\alpha^2q^2/(2\pi) + |\mu|^2}$ [BBGS19]. From $\mathbf{z}'$ we can recover the error vector $\mathbf{e}$ and thereby the secret vector $\mathbf{s} = \mathbf{z} - \mathbf{e} \mod q$.

A solution to γ-uSVP can be found by reducing it to δ-HSVP where $\gamma = \delta^2$ [APS15]. Various algorithms, in particular, lattice reduction algorithms, exist to solve δ-HSVP. If we are able to solve a linear number of δ-HSVP instances that correspond to a δ^2-approximate SVP instance, we can construct a solution of the latter (see [Lov87, Section 1.2.21] for more details). Consider any lattice with uSVP structure. In exactly one direction, that is, in the direction of its unique shortest vector, the lattice has vectors that are significantly smaller than in other directions. A lattice reduction algorithm that yields a sufficiently good output basis quality, therefore, must return some small vector in the desired direction. Let $\mathbf{v}$ be a solution to SVP_δ^2, i.e. $\|\mathbf{v}\| \leq \delta^2\lambda_1(\Lambda(\mathbf{B}'))$. All other vectors $\mathbf{w} \in \Lambda(\mathbf{B}')$ that are not multiples of a shortest vector have length $\|\mathbf{w}\| \geq \lambda_2(\Lambda(\mathbf{B}')) > \delta^2\lambda_1(\Lambda(\mathbf{B}'))$. Thus, we obtain a solution to γ-uSVP and, as shown above, we can reconstruct the secret vector to solve LWE.

The attack is successful with high probability if $\lambda_2(\Lambda(\mathbf{B}'))/\lambda_1(\Lambda(\mathbf{B}')) \geq \tau\delta^m$ [AFG13]. We assume that the determinant of $\Lambda_q(\mathbf{A}^\top)$ is given by q^{m-n} (see Equation (2.11)). The length of the error is $\|\mathbf{e}\| \approx \sqrt{m \cdot \left(\alpha q/(\sqrt{2\pi})\right)^2} \approx \sqrt{m/(2\pi)}\alpha q$. Furthermore, in practice $\mu = 1$ is used [APS15]. We then apply the simplified Gaussian heuristic as in [Göp16] on $\lambda_2(\Lambda(\mathbf{B}')) = \lambda_2(\Lambda(\mathbf{A}^\top))$ (see Equation (2.8)) to obtain

$$\begin{aligned} \tau\delta^m &\leq \frac{\sqrt{\frac{m}{2\pi e}}\det(\Lambda(\mathbf{A}^\top))}{\lambda_1(\Lambda(\mathbf{B}'))} \approx \frac{\sqrt{m}q^{1-\frac{n}{m}}}{\sqrt{2\pi e}\lambda_1(\Lambda(\mathbf{B}'))} \leq \frac{\sqrt{m}q^{1-\frac{n}{m}}}{\sqrt{2\pi e}\|\mathbf{e}\|} \\ &\approx \frac{q^{1-\frac{n}{m}}}{\sqrt{2\pi e}\frac{\alpha q}{\sqrt{2\pi}}} \approx \frac{q^{1-\frac{n}{m}}}{\sqrt{e}\alpha q} \end{aligned}$$

The success probability of the attack is thus non-negligible if

$$\delta \leq \left(\frac{q^{1-\frac{n}{m}}}{\tau\sqrt{e}\alpha q}\right)^{\frac{1}{m}} \tag{3.7}$$

Experiments show that $\tau \leq 0.4$ achieves a success rate of about $\epsilon = 0.1$ [AFG13]. In order to increase the success probability to some fixed target $\epsilon' > \epsilon$, we can simply repeat the algorithm ρ times and obtain a success probability of $\epsilon' \leq 1 - (1 - \epsilon)^\rho$ [BBGS19]. Consequently, we need at least $\rho \geq \log(1 - \epsilon')/\log(1 - \epsilon)$ rounds for a successful attack.

3.2.5 The BKW Algorithm [BKW03]

The Blum, Kalai and Wasserman (BKW) algorithm was originially designed to solve the Learning Parity with Noise problem (LPN) [BKW03]. In Section 2.5.1 we pointed out that LPN is a subproblem of LWE and Albrecht *et al.* adapted BKW to LWE in [ACF+15]. The runtime and memory complexity of BKW is in $2^{O(n)}$ for an LWE instance with secret dimension n prime modulus $q \in \mathrm{poly}(n)$. The number of samples m must be sufficiently large (in $\otimes(n \log n)$).

BKW falls into the regime of dual attacks, that is, it solves LWE by finding a short vector $\mathbf{s}$ in the scaled dual lattice $\Lambda(\mathbf{A}^\top)^\perp$.

Albert *et al* divide the algorithm into three stages [ACF+15], namely, sample reduction, hypothesis testing and back substitution.

Sample Reduction. Algorithm 5 shows the sample reduction part of the BKW algorithm. The notation is inspired by the textual description in [GJS15] with minor adjustments.

For the algorithm, we use the matrix notation of LWE as in Equation (2.16), i.e. $\mathbf{z} = \mathbf{A}^\top \mathbf{s} + \mathbf{e}$. BKW consists of a series of BKW steps that iteratively reduce the dimension of input matrix $\mathbf{A}$ by finding collisions of its column vectors in the currently examined block of b entries. We start from the last b entries of $\mathbf{A}^{(1)} = \mathbf{A}$. In every step i, we maintain a collision table $\mathbf{T}^{(i)}$ and loop over the columns $\mathbf{a}_k^{(i)}$ of $\mathbf{A}^{(i)}$ and distinguish between the following cases: (1) If $\mathbf{a}_k^{(i)}$ only has zero entries in the examined block, pass $\mathbf{a}_k^{(i)}$ and $z_k^{(i)}$ to the next step, (2) if no match of $\mathbf{a}_k^{(i)}$ or the negation of $\mathbf{a}_k^{(i)}$ can be found in the collision table, add $\mathbf{a}_k^{(i)}$ to the colission table, and (3) if a match $\mathbf{a}_l^{(i)}$ is found, compute $\mathbf{a}_l^{(i)} + \mathbf{a}_k^{(i)}$ or in the case of a negation match $\mathbf{a}_l^{(i)} - \mathbf{a}_k^{(i)}$ (in $\mathbb{Z}_q$) such that the last b nonzero entries cancel out. By exploiting the symmetry of $\mathbb{Z}_q$ in this way, in every step we obtain at most $(q^b - 1)/2$ columns with distinct coefficients in the current b entries. We also make note of "observed symbols" $z_j^{(i)}$ that represent the combination of two samples given their respective matching columns (see lines 20, 24 for more details).

In each BKW step, the number of columns (and samples) decreases by at least $(q^b - 1)/2$ (size of the collision set) and the variance of the error distribution σ^2 increases by a factor of two. The algorithm terminates after $t = \lceil b/(n-d) \rceil$ steps returns a a set of observed symbols $\mathbf{z}^{(t)}$ and a corresponding reduced matrix $\mathbf{A}^{(t)}$ in which only the first d rows have nonzero entries. Parameter d should be set to 1, as in the original BKW algorithm, or 2 for the best performance [ACF+15].

The remaining part $\mathbf{s}'$ of the secret vector $\mathbf{s}$ is then guessed by means of hypothesis testing. After t steps the error term $\left(\mathbf{z}_j^{(t)} - \left\langle \mathbf{s}', \mathbf{a}_j^{(t)} \right\rangle\right)$ with $j \in [m']$ of the m' remaining observed symbols follows a Gaussian distribution χ with noise $\sigma'^2 = 2^t \cdot \sigma^2$ (for more details, see [ACF+15, Lemma 1]). We can test the noise of the error term for all $\mathbf{s}'' \in \mathbb{Z}_q^d$ against the hypothesized noise σ'^2 by means of the log-likelihood ratio (for details we again refer to [ACF+15]) and are thus able to determine $\mathbf{s}'$ given sufficiently many samples m'.

Finally, we can apply back substitution to recover all elements of $\mathbf{s}$. We again apply a similar procedure as in Algorithm 5 to reduce a number of columns from the colission tables computed in the Sample Reduction step and obtain m' columns with $d + d'$ nonzero entries and their corresponding "observed symbols". We then substitute the part of s that was recovered in the previous steps and recover the next part of $\mathbf{s}$ by hypothesis testing and repeat the process until we have found $\mathbf{s}$.

Algorithm 5: BKW (Sample Reduction)

```
function BKW($\mathbf{A} \in \mathbb{Z}^{n\times m}, \mathbf{z} \in \mathbb{Z}^m, b \in \mathbb{Z}, d \in \mathbb{Z}$)
    $i = 1$
    $\mathbf{A}^{(i)} = \mathbf{A}$
    $\mathbf{z}^{(i)} = \mathbf{z}$
    while the last $n - d$ coefficients of the columns of $\mathbf{A}^{(i)}$ are nonzero do
        // BKW step
        $j = 1$
        $\mathbf{T}^{(i)} = []$                                         // Collision table
        for $k = 1, \ldots, m^{(i)}$ do
            // $m^{(i)}$ is number of columns in $\mathbf{A}^{(i)}$
            if last $(i \cdot b)$ coefficients of $\mathbf{a}_k^{(i)}$ are zero then
                $\mathbf{a}_j^{(i+1)} = \mathbf{a}_k^{(i)}$
                $z_j^{(i+1)} = z_k$
                $j = j + 1$
            else if no match for $\mathbf{a}_k^{(i)}$ in $\mathbf{T}$ then
                $\mathbf{T} = \mathbf{T} + \left[\mathbf{a}_k^{(i)}\right]$                  // append to collision set
            else if match $\mathbf{a}_l^{(i)}$ for $\mathbf{a}_k^{(i)}$ is found then
                if $\mathbf{a}_l^{(i)}$ matches $\mathbf{a}_k^{(i)}$ in the last $(i \cdot b)$ components then
                    $\mathbf{a}_j^{(i+1)} = \mathbf{a}_k^{(i)} - \mathbf{a}_l^{(i)}$;       // last $i \cdot b$ coefficients of $\mathbf{a}_j^{(i+1)}$ are now zero
                    $z_j^{(i+1)} = z_k^{(i)} - z_l^{(i)} = y_j^{(i)} + e_j^{(i)}$, where $y_j^{(i)} = \left\langle \mathbf{s}, \mathbf{a}_j^{(i)} \right\rangle$ and $e_j^{(i)} = e_k^{(i)} - e_l^{(i)}$
                    $j = j + 1$
                else if the negation of $\mathbf{a}_l^{(i)}$ in $\mathbb{Z}_q^n$ matches $\mathbf{a}_k^{(i)}$ in the last $(i \cdot b)$ components then
                    $\mathbf{a}_j^{(i+1)} = \mathbf{a}_k^{(i)} + \mathbf{a}_l^{(i)}$
                    $z_j^{(i+1)} = z_k^{(i)} + z_l^{(i)} = y_j^{(i)} + e_j^{(i)}$, where $y_j^{(i)} = \left\langle \mathbf{s}, \mathbf{a}_j^{(i)} \right\rangle$ and $e_j^{(i)} = e_k^{(i)} + e_l^{(i)}$
                    $j = j + 1$
        $i = i + 1$
        // Calculate input for next BKW step
        $\mathbf{A}^{(i)} = (\mathbf{a}_1^{(i)} \cdots \mathbf{a}_{j-1}^{(i)})$
        $\mathbf{z}^{(i)} = (z_1^{(i)}, \ldots, z_{j-1}^{(i)})$
    Return $(\mathbf{A}^{(i)}, \mathbf{z}^{(i)})$
```

Theorem 1 (BKZ Complexity [ACF+15, Corollary 2])
Let $(\mathbf{a}_i, z_i)$ *be samples following* $\mathcal{A}_{\mathbf{s},\chi}$*, set* $a = \lfloor \log_2(1/(2\alpha)^2) \rceil$*,* $b = n/a$ *and* q *a prime. Let* d *be a small constant* $0 < d < \log_2(n)$*. Assume* α *is such that* $q^b = q^{n/a} = q^{n/\lfloor \log_2(1/(2\alpha)^2) \rceil}$ *is superpolynomial in n. Then, given these parameters the cost of the BKW algorithm to solve Search-LWE is*

$$\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)\right)+\left\lceil\frac{q^b}{2}\right\rceil\cdot\left(\left\lceil\frac{n}{d}\right\rceil+1\right)\cdot d\cdot a+poly(n)\approx(a^2n)\cdot\frac{q^b}{2} \tag{3.8}$$

operations in $\mathbb{Z}_q$*. Furthermore,*

$$a\cdot\left\lceil\frac{q^b}{2}\right\rceil+poly(n)\text{ calls to }\mathcal{A}_{\mathbf{s},\chi}\text{ and storage of }\left(a\cdot\left\lceil\frac{q^b}{2}\right\rceil\cdot n\right)\text{ elements in }\mathbb{Z}_q\text{ are needed.} \tag{3.9}$$

The first summand of Equation (3.8) roughly corresponds to the cost of creating the colission tables and the second summand is the cost of backsubstitution. For a more detailed cost analysis, see [ACF+15, Theorem 2].

Coded-BKW [GJS15]

The *estimator* uses a more efficient variant of BKW, called Coded-BKW [GJS15], that makes use of coding theory. We will only present the high level idea of Coded-BKW.

The main advantage of the algorithm comes from an improved "coded" BKW step. The improved step removes more column entries at the cost of an additional noise term. To avoid that the noise increases too much, the final algorithm combines original BKW steps with the "coded" BKW step. Let I be an index set and $\mathbf{x}_I$ the part of $\mathbf{x}$ with entries indexed by I.

In a "coded" step i, we set I as the set of b positions to be removed and fix some q-ary linear $[N_i, b]$ code C_i with q^b codewords. We then search for the closest codeword $\mathbf{c}_I \in C$ for every input vector $\mathbf{a}_I$ such that $\mathbf{a}_I = \mathbf{c}_I + \mathbf{e}_I$, where the error part $\mathbf{e}_I \in \mathbb{Z}_q^{N_i}$ is minimized by a decoding procedure. Finally, we subtract two vectors and their correpsonding samples and pass the result to the next BKW step. Consider the inner product $\langle \mathbf{s}_I, \mathbf{a}_I \rangle = \langle \mathbf{s}_I, \mathbf{c}_I \rangle + \langle \mathbf{s}_I, \mathbf{e}_I \rangle$. In the subtraction, only the error part $\langle \mathbf{s}_I, \mathbf{e}_I \rangle$ remains.

3.2.6 Dual Attack [MR09]

The dual attack [MR09] falls into the regime of solving LWE in the dual SIS lattice (see [**sec:dual-attacks**]). We will present the attack and show how it can be applied to LWE in Section 3.3.1.

3.2.7 Other Approaches

In the previous subsections we gave an overview the most relevant algorithms in practice. The list is by no means exhaustive. We indicated that many improvements and variants have been suggested within the last two decades. In particular, for special cases LWE in which the components of the secret vector are sampled from a small set $\mathcal{S} \subset \mathbb{Z}$ it is possible to speed up the algorithms (for example by using a technique called modulus switching for BKW [AFFP14]). Small secret variants also give rise to a different approach due to the fact of the drastically decreasing search space. In the Meet-In-The-Middle attack, we create a sorted list of $\mathbf{A}^\intercal \mathbf{s}'$ for for all $\mathbf{s}' \in \left\{\mathbf{v} \in \mathcal{S}^n \mid v_i = 0 \text{ for } \frac{n}{2} < i \leq n\right\}$. We then iterate over all $\mathbf{s}'' \in \left\{\mathbf{v} \in \mathcal{S}^n \mid v_i = 0 \text{ for } 0 \leq i \leq \frac{n}{2}\right\}$ and check if $\mathbf{z} - \mathbf{A}^\intercal \mathbf{s}''$ matches any value in the list. The basic version of this algorithm has a runtime and memory complexity of about $|\mathcal{S}|^{n/2}$.

The *estimator* also includes estimates for an algorithm due to Arora and Ge [AG11] that has sub-exponential runtime for a sufficiently narrow Gaussian distribution. In practical cryptographic scenarios however, the algorithm has a much higher cost than other algorithms and is thus irrelevant for our purposes.

3.3 Algorithms for Solving SIS

Recall that the $\text{SIS}_{n,q,m,\beta}$ problem asks to find a short vector $\mathbf{s} \in \mathbb{Z}_q^m$ of norm $\|\mathbf{s}\| \leq \beta$ such that such that $\mathbf{A} \cdot \mathbf{s} = 0 \bmod q$ for some uniformly distributed matrix $\mathbf{A}^{n\times m}$. Solving SIS is equivalent to finding a short vector in the dual lattice $\Lambda(\mathbf{A}^\intercal)^\perp = \{\mathbf{y} \in \mathbb{Z}^m \mid \mathbf{Ay} = \mathbf{0} \mod q\}$.

3.3.1 Lattice Reduction

MR variant [MR09]

Our first approach to solving SIS, sometimes referred to as the "dual attack", follows quite naturally. Given $\mathbf{A}$, we can efficiently compute the basis $\mathbf{B}(\mathbf{B}^\intercal\mathbf{B})^{-1}$ of the dual lattice $\Lambda(\mathbf{A}^\intercal)^\perp$ in polynomial time using Gauss-Jordan elimination or other more modern algorithms.

We can then apply a lattice reduction algorithm and obtain a basis with root Hermite factor δ. The first basis $\mathbf{b}_1$ vector of the reduced basis has length $\|\mathbf{b}_1\| = \delta^m \text{det}(\Lambda(\mathbf{A}^\intercal)^\perp)^{1/m}$. We can see that δ depends on the subdimension m which we want to be ideal in order to minimize the cost of the lattice reduction by relaxing δ.

We further assume that $\det(\Lambda(\mathbf{A}^\intercal)^\perp) = \text{Vol}(\Lambda(\mathbf{A}^\intercal)^\perp) = q^n$ (see Equation (2.12)). Our first equation then becomes

$$\|\mathbf{b}_1\| = \delta^m q^{\frac{n}{m}}, \tag{3.10}$$

which is minimal for $m = \sqrt{n \log q / \log \delta}$.

Theorem 2 (Optimal subdimension *m* [MR09])
Given a q-ary scaled dual lattice $\Lambda(\mathbf{A}^\top)^\perp$ defined by a matrix $\mathbf{A} \in \mathbb{Z}^{n\times m}$ with m sufficiently larger than n and a prime q. Then a lattice reduction algorithm yields an optimal output if performed in subdimension

$$m' = \sqrt{\frac{n \log q}{\log \delta}}. \tag{3.11}$$

Higher dimension increase the complexity of the reduction algorithms and lower dimensions may cause a lack of sufficiently short lattice vectors [MR09]. In contexts in which Equation (3.10) does not hold, we may still choose m as in Equation (3.11) heuristically. Removing columns from $\mathbf{A}$ does not greatly impact our results since we can just set the corresponding components of the secret vector s to zero. We reformulate Equation (3.10) a bit:

$$\begin{aligned} \|\mathbf{b}_1\| = \delta^m q^{\frac{n}{m}} &\iff \log\beta = m\log\delta + \frac{n\log q}{m} \\ &\iff \log\delta = \frac{\log\beta}{m} - \frac{n\log q}{m^2} \end{aligned} \tag{3.12}$$

We continue by plugging Equation (3.11) into Equation (3.12):

$$\begin{aligned} \log\delta = \frac{\log\beta}{\sqrt{\frac{n\log q}{\log\delta}}} - \frac{n\log q}{\left(\sqrt{\frac{n\log q}{\log\delta}}\right)^2} &\iff \log\delta = \frac{\log\beta}{\sqrt{\frac{n\log q}{\log\delta}}} - \log\delta \\ &\iff 2\log\delta = \frac{\log\beta}{\sqrt{\frac{n\log q}{\log\delta}}} \\ &\iff \log\delta = \frac{\log^2\beta}{4n\log q} \end{aligned} \tag{3.13}$$

Theorem 3 (Optimal subdimension *m* [MR09])
Given a q-ary scaled dual lattice $\Lambda(\mathbf{A}^\top)^\perp$ defined by a matrix $\mathbf{A} \in \mathbb{Z}^{n\times m}$ with m sufficiently larger than n and a prime q. Then a lattice reduction algorithm performed in its optimal subdimension achieves a log root Hermite factor of

$$\log\delta = \frac{\log^2\beta}{4n\log q}. \tag{3.14}$$

To estimate the cost of the lattice reduction for SIS, we call a function from the *estimator* to find the required block size k such that BKZ achieves root Hermite factor δ and apply a cost model with the optimal subdimension m' and block size k.

Note that for LWE we have α, q as input parameters instead of a bound. The advantage of distinguishing $\langle \mathbf{v}, \mathbf{e}\rangle$ from random is given by $\epsilon = e^{-\pi(\|\mathbf{v}\|\alpha)^2}$. We can thus convert α to a required bound $\beta = \frac{1}{\alpha}\sqrt{\ln(\frac{1}{\epsilon})/\pi}$ such that the success probability of solving an LWE instance is given by ϵ [APS15, Corollary 2]. The *estimator* uses a rinse and repeat strategy to find the best tradeoff between runtime and success probability.

RS variant [RS10]

A similar approach is described in [RS10]. The optimal subdimension and required root Hermite factor are given by a slightly different expression. Apart from that the attack works as described in Section 3.3.1.

Theorem 4 (Optimal subdimension *m* [RS10, Conjecture 2]))
For every $n \geq 128$, constant $c \geq 2, q \geq n^c, m = \Omega(n \log_2(q))$ and $\beta < q$, the best known approach to solve SIS with parameters (n, m, q, β) involves solving δ-HSVP in dimension $m' = \min(x : q^{2n/x} \leq \beta)$ with $\delta = \sqrt{d}\beta / q^{n/m'}$.

We reformulate the expression for m'

$$\begin{aligned} q^{2n/m'} &\leq \beta \\ \frac{2n}{m' \log(q)} &\leq \beta \\ m' &\geq \frac{2n \log(q)}{\log(\beta)} \end{aligned} \tag{3.15}$$

and obtain $m' = \left\lceil \frac{2n \log(q)}{\log(\beta)} \right\rceil$. If $m' > m$, we take $m' = m$.

The root Hermite factor δ must be larger than 1 for the reduction to be tractable. From $\delta = \sqrt{d}\beta / q^{n/m'} \geq 1$ it follows that we need $m' \geq n \log_2(q) / \log_2(\beta)$.

3.3.2 Combinatorial Attack [MR09]

Micciancio and Regev also describe a combinatorial method for solving SIS [MR09] that is similar to BKW.

Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and the dual lattice $\Lambda(\mathbf{A}^\intercal)^\perp$, we want to find a lattice vector $\mathbf{v} \in \Lambda(\mathbf{A}^\intercal)^\perp$ with coefficients bounded by β. Expressed differently, we want to find $\mathbf{v}$ such that $\mathbf{Av} = 0 \mod q$ and $\|\mathbf{v}_i\| \leq b$ for all $i \in [m]$.

We begin by dividing the columns of $\mathbf{A}$ into 2^k for some k. Each set contains $m/2^k$ column vectors. We now compute all linear combinations $c_1\mathbf{b}_1 + \cdots + c_{m/2^k}\mathbf{b}_{m/2^k}$, where $\mathbf{b}_i$ denote the indexed column vectors in each set, such that $|c_i| \leq b$ and obtain 2^k new sets $\mathbf{A}_j^{(k)}$ of $L = (2b+1)^{m/2^k}$ vectors, $j \in \left[2^k\right]$. Remember that each c_i represents a coeffient of the lattice vector. By means of these new sets we satisfy the shorteness criteria of the output vector.

We then continue iteratively for $i = k$ to $i = 0$ as follows. In each step we merge pairs of two sets $\mathbf{A}_j^{(i)}, \mathbf{A}_{j+1}^{(i)}$. If a vector $\mathbf{x}$ in first set of the pair can be combined with each vector $\mathbf{y}$ in second set such that the first $\log_q L$ components in $\mathbf{x} \pm \mathbf{y}$ are zero, we put the result in combined set in $\mathbf{A}_j^{(i-1)}$. The size of the combined sets is at most L as we consider a part of the vectors with $\log_q L$ components that can take at most $q^{\log_q L} = L$ different values. We start from $j = 0$ and increment j by 2 after each merge for $j < 2^i$. After the merge now 2^{i-1} sets.

We choose k such that

$$n \approx (k+1)\log_q L = (k+1)\log_q (2\beta+1)^{m/2^k} \iff \frac{2^k}{k+1} \approx \frac{m\log(2\beta+1)}{n\log(q)} \tag{3.16}$$

After k steps the first $k \log_q L$ entries of the columns in the result set are cancelled out. We expect that of the remaining $\approx \log_q L$ entries in result set that we should find at least one zero vector as at there again are most L different vectors. The zero vector represents the linear combination with entries bounded by β and we can easily reconstruct the short lattice vector $\mathbf{v}$ with $\|\mathbf{v}\|_\infty \leq \beta$.

To find an optimal k, we iterate over k starting from $k = 1$ and calculate diff as follows:

$$\frac{2^k}{k+1} \approx \frac{m\log(2\beta+1)}{n\log(q)} \tag{3.17}$$

$$\text{diff} = \text{abs}\left(\frac{2^k}{k+1} - \frac{m\log(2\beta+1)}{n\log(q)}\right). \tag{3.18}$$

When diff does not decrease for 10 iteration steps, we stop and take the current k.

In our tool we include two estimates of the cost of the algorithm. The overall runtime is dominated by the size of the sets L. Variants of the algorithm may speed up various steps and hence in our first, more conservative, estimate algorithm `algorithms.combinatorial_conservative()` we neglect the cost of single operations in the algorithm and just set the cost to the list size L. We obtain a more realistic estimate by considering the number of operations needed to create the initial sets. Each of the 2^k lists contains L vectors. The cost for any operation on a list element is at least $\log_2(q) \cdot n$. Hence, the total cost is $2^k \cdot L \cdot \log_2(q) \cdot n$. This second estimate is included in our tool in `algorithms.combinatorial()`.

4 Lattice Parameter Estimation Tool

The main goal of the thesis is the creation a tool that combines the algorithm estimates for LWE and SIS that we introduced above and can be configured without much knowledge of the workings of the underlying algorithms. To our knowledge there has not been a tool that includes the function of generically searching for secure parameters for both LWE and SIS instances as well as for ring and module variants. Some schemes (e.g. commitment schemes, see Section 5.1) depend on the statistical security of either LWE or SIS. We hence include classes respectively to find parameters satisfying this criteria. Furthermore, we provide a set of utility classes and methods for most commonly used distributions and norms. A configuration class allows for a substantial customization of the estimation process. The tool can either estimate the bit security level of fixed parameter sets or generically search for parameter sets that satisfy a certain bit security level.

4.1 Supported Distributions

The secret distribution used in encryption schemes based on LWE or SIS can either distributed according to a Gaussian or a uniform distribution. The error distribution in LWE must be a Gaussian distribution with parameter α. In general, if both error and secret are sampled in LWE are sampled according to Gaussian distributions their parameters may differ. Currently, however, the *estimator* only supports secrets that follow a uniform distributions or a Gaussian distribution with the same paramter as the error distribution.

4.1.1 Gaussian Distribution

In some applications we receive a Gaussian distribution as input but require a bound in some norm in order to estimate the hardness of an SIS instance. Hence, we need to transform a Gaussian with parameter $s = \sqrt{2\pi}\sigma$ into a bound β given some security parameter sec. Note that a n-dimensional Gaussian $D_{\mathbf{Z}^n,s}$ can be sampled by combining samples from n independent one-dimensional Gaussians $D_{\mathbf{Z},s}$ [GJS15].

For a Gaussian distribution and a random variable X with $X \sim D_{\mathbf{Z},s}$, the following holds:

$$\Pr\left[|X| \geq \beta\right] \leq 2e^{-\pi \|\mathbf{v}\|^2/s^2} \tag{4.1}$$

We demand $2e^{-\pi\beta^2/s^2} \approx 2^{-\mathsf{sec}}$ with $\beta = \|\mathbf{v}\|$ and obtain

$$\begin{aligned}
2e^{-\pi\beta^2/s^2} &\approx 2^{-\mathsf{sec}} \\
-\pi\frac{\beta^2}{s^2} &\approx (-\mathsf{sec} - 1)\ln(2) \\
\beta &\approx s\sqrt{\frac{(\mathsf{sec}+1)\ln(2)}{\pi}}.
\end{aligned}$$

Theorem 5 (Gaussian to Bound)
Given a Gaussian distribution $D_{\mathbb{Z}^n,s}$ with width parameter $s = \sqrt{2\pi}\sigma$ and a security parameter sec, *we can compute a bound β such that a sample $\mathbf{v}$ drawn from $D_{\mathbb{Z}^n,s}$ satisfies $Pr\left[\|\mathbf{v}\| \geq \beta\right] \leq 2^{-\mathsf{sec}}$ as follows:*

$$\beta \approx s\sqrt{\frac{(\mathsf{sec}+1)\ln(2)}{\pi}}. \tag{4.2}$$

The resulting bound β is an ℓ_2-norm bound.

We provide the classes `GaussianAlpha`, `GaussianSigma` and `GaussianS` to allow for the most flexibility in specifying a Gaussian distribution.

4.1.2 Uniform Distribution

For uniform distributions, we support all distributions that are supported by the *estimator*, namely, uniform $\mod q$, uniform in the interval $[a, \ldots, b]$ and a sparse uniform distribution with parameters $((a, b), h)$, where exactly h components are in the interval $[a, \ldots, b] \setminus \{0\}$ and all other components are zero. Note that we only consider discrete distributions are over $\mathbb{Z}$.

We can estimate the corresponding standard deviation for a Gaussian distribution by computing the variance of the uniform distribution. Given an lower and upper bound (a, b), the variance for a discrete uniform distribution is defined as $\sigma^2 = \frac{(b-a+1)^2-1}{12}$. For a uniform distirbution $\mod q$ we set $a = -\left\lfloor \frac{q}{2} \right\rfloor, b = \left\lfloor \frac{q}{2} \right\rfloor$.

For a sparse discrete n-dimensional uniform distirbution $\mathcal{U}_h$ with an additional sparseness parameter h and a random variabel $X \sim \mathcal{U}_h$, we can compute the expected values $\mathbb{E}(X^2)$ and $\mathbb{E}(X)^2$ as follows [APS15]:

$$\mathbb{E}(X^2) = \frac{h}{n} \cdot \frac{2b^3 + 3b^2 + b - 2a^3 + 3a^2 - a}{6(b-a)} \tag{4.3}$$

$$\mathbb{E}(X)^2 = \frac{h}{n} \cdot \frac{b(b+1) - a(a-1)}{2(b-a)} \tag{4.4}$$

and obtain the variance $\mathrm{Var}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$.

4.2 Norms bound estimates

In practical scenarios, oftentimes we require that the bound of a result does not exceed a certain value to guarantee correctness. In addition, we have seen that estimation algorithms rely on bounds in different ℓ_p-norms. For example, the combinatorial attack in Section 3.3.2 needs a bound in ℓ_∞ norm, whereas the dual attack in Section 3.3.1 uses bounds in the Euclidean norm. We thus need a way to bound a value of a bound in one norm by a value in a different norm. In our tool, we define a norm class with parameter p and provide a method `to_Lp()` for conversions into all other norms mentioned here. We begin by the following Theorem.

Theorem 6
Let $f \in \mathcal{R}_q$ with $f = \sum_i f_i X^i$ as in in [BDL+18] and $p, q \in \mathbb{N}$ with $\infty \geq p \geq q \geq 1$. Then the following inequation holds:

$$\|f\|_p \leq \|f\|_q. \tag{4.5}$$

Let $p, q \in \mathbb{N}$ with $1 \leq p \leq q \leq \infty$. Then the following inequation holds:

$$\lim_{q' \to q} \|f\|_p \leq \lim_{q' \to q} n^{\frac{1}{p} - \frac{1}{q'}} \|f\|_q'. \tag{4.6}$$

It is easy to see from the definition of the norms in Section 2.2 that Equation (4.5) is true. Section 4.2 follows if we apply Hölder's inequality to finite vector spaces as in [Nor12].

In addition, oftentimes we consider norms in the canonical embedding. Let $\mathcal{O}_K$ be the ring of integers of a number field $K = \mathbb{Q}(\theta)$, where θ is an algebraic number and σ denote the canonical embedding as defined in [DPSZ12]. Then, according to [DPSZ12, Theorem 7] for $f \in \mathcal{O}_K$ the following inequations hold:

$$\|f\|_\infty \leq \|\sigma(f)\|_\infty, \tag{4.7}$$

$$\|\sigma(f)\|_\infty \leq \|f\|_1. \tag{4.8}$$

(We assume that the constant C_m used in the original Theorem is 1 for m a power of 2 [DPSZ12, Lemma 3].) If we combine Equations (4.7) and (4.10) respectively with we obtain the following theorem.

Theorem 7
Let $f \in \mathcal{O}_K$ as in in [DPSZ12]. Then, the following two inequations hold:

$$\|\sigma(f)\|_\infty \leq \|f\|_1 \leq n^{1-\frac{1}{p}} \|f\|_p, \text{ and} \tag{4.9}$$

$$\|f\|_p \leq n^{\frac{1}{p}} \|f\|_\infty \leq n^{\frac{1}{p}} \|\sigma(f)\|_\infty \tag{4.10}$$

We can thus bound a given bound in the norm ℓ_q for some q by a bound in some ℓ_p norm using the above inequalities and analogously in the embedding norm, which we call C_p. We combine both results in the method `to_Lp()` and `to_Cp()` in the norm classes `Lp` and `Cp` respectively. Note that the embedding norm C_p internally works like the ℓ_p norm as we define both for vector spaces of dimension n. We merely need the results in Theorem 7 for the conversion between both norms.

We also want to be able to add and multiply bounds from different norms. Note that the degree of the polynomial of the underlying ring to which we apply the norms must match. To add two bounds we simply bound the second addend by the used norm of the first as described in Theorems 6 and 7 and add both bounds to obtain a bound in the norm of the first addend. It is slightly more complicated for multiplication and we state the results in the next theorem.

Theorem 8 (Multiplication of Norm Bounds [BDL+18; DPSZ12])
Let f be defined as above and let $g \in \mathcal{R}_q$ where $g = \sum_i \overline{g}_i X^i$ where $g_i \in [-(q-1)/2, (q-1)/2]$ and $\overline{g}_i = g_i \mod q$ as in [BDL+18]. We then can define the following inequations for multiplication according to [BDL+18]:

- *If $\|f\|_\infty \leq \beta, \|g\|_1 \leq \gamma$ then $\|f \cdot g\|_\infty \leq \beta \cdot \gamma$.*
- *If $\|f\|_2 \leq \beta, \|g\|_2 \leq \gamma$ then $\|f \cdot g\|_\infty \leq \beta \cdot \gamma$.*

Let $x, y \in \mathcal{O}_K$. Then, the following inequation holds according to [DPSZ12]:

$$\begin{aligned} \|x \cdot y\|_\infty &\leq n^2 \cdot \|x\|_\infty \cdot \|y\|_\infty \\ \|\sigma(x \cdot y)\|_\infty &\leq \|\sigma(x)\|_\infty \cdot \|\sigma(y)\|_\infty. \end{aligned} \tag{4.11}$$

(Again, we assume that $C_m = 1$ in the original statement.)

If the two norms bounds that we want to multiply are given in a different norm, we simply convert the bounds to the C_∞ norm as described in Equation (4.9). Analogously, we can multiply two bounds $\|\sigma(f)\|_p, \|\sigma(g)\|_q$ in the embedding norm by applying Theorem 8. In the case of , we have to recursively embedded $\sigma(f)$ and $\sigma(g)$. With Equation (4.10) we obtain

$$\begin{aligned} \|\sigma(f) \cdot \sigma(g)\|_\infty \leq \|\sigma(\sigma(f) \cdot \sigma(g))\|_\infty &\leq \|\sigma(\sigma(f))\|_\infty \cdot \|\sigma(\sigma(g))\|_\infty \\ &\leq n^{1-\frac{1}{p}} \|\sigma(f)\|_\infty \cdot n^{1-\frac{1}{q}} \|\sigma(f)\|_\infty. \\ &\leq n^{2-\frac{1}{p}-\frac{1}{q}} \|\sigma(f)\|_\infty \cdot \|\sigma(f)\|_\infty. \end{aligned}$$

4.3 Problem Classes

We now present the problem classes in `lattice_parameter_estimation/problem` (see Figure 4.1).

`LWE` and `SIS` inherit from the base class `BaseProblem` respectively. All instances provide a method `get_estimate_algorithms()` that returns a list of algorithm instances that can be executed by the function `estimate()`. Furthermore, any instance of `BaseProblem` can be compared to a bit security level (for more details, we refer the reader to the documentation). The `LWE` class is initialized by the secret dimension n, a modulus q, the number of samples m, a `secret_distribution` and a `error_distribution`. Both `secret_distribution` and `error_distribution` must be instances of the class `distributions.Distribution`. Instead of `secret_distribution` and `error_distribution`, a bound of type `norm.BaseNorm` must be set for `SIS`. Note that both distributions.Uniform and distributions.Gaussian are instances of `norm.BaseNorm` and can thus be used as a bound. We compute the bound for a given distribution instance as described in Section 4.1.

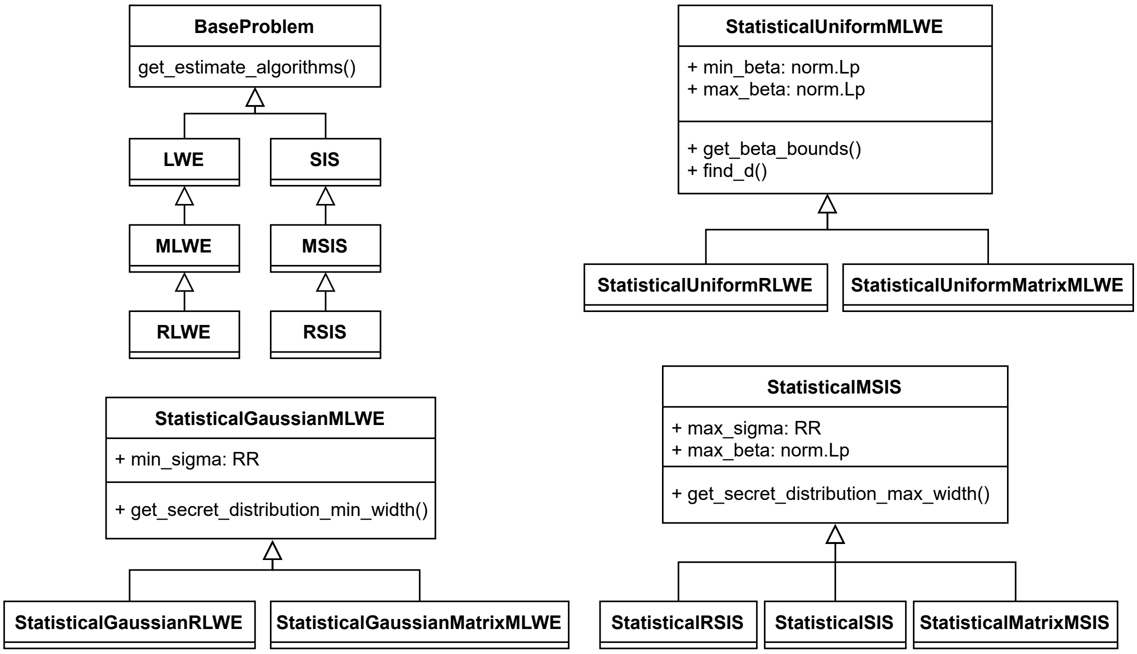

Figure 4.1: Problem classes

For ring and module variants `RLWE`, `RSIS` and `MLWE`, `MSIS` respectively n denotes the degree of the polynomial of the underlying ring $\mathcal{R}_q$. The module variants `MLWE` and `MSIS` take an addition parameter d for the rank of the module.

While there exist special cases where the ring structure of problem instances can be exploited in an attack on LWE or SIS, in general, the hardness of ring and Module variants is estimated by interpreting the coefficients of elements of $\mathcal{R}_q$ as vectors in $\mathbb{Z}_q^n$ [ACD+18]. If we take into account the considerations we presented in Section 2.5.3, we can thus reduce ring and Module instances as follows when calling `get_estimate_algorithms()` on the ring and module variant of `LWE` and `SIS`:

- $\text{RLWE}_{n,q,m,\chi} \longrightarrow \text{LWE}_{n,q,m\cdot n,\chi}$
- $\text{MLWE}_{n,d,q,m,\chi} \longrightarrow \text{LWE}_{n\cdot d,q,m\cdot n,\chi}$
- $\text{RSIS}_{n,q,m,\beta} \longrightarrow \text{SIS}_{n,q,m\cdot n,\beta}$
- $\text{MSIS}_{n,d,q,m,\beta} \longrightarrow \text{SIS}_{n\cdot d,q,m\cdot n,\beta}$

In addition to the base problem variants, we define some statistically secure variants for both LWE and SIS since some schemes depend on the unconditional hardness of either LWE or SIS. More precicely, we ask for parameters such an arbitrary powerful attacker can only break the scheme with probability less than $2^{-\text{sec}}$.

`StatisticalGaussianMLWE`. For LWE, we define a statistically secure variant over a Gaussian distribution and over a uniform distribution. `StatisticalGaussianMLWE` follows Corollary 7.5 and Theorem 7.4 in [LPR13]. The mapping of the parameters in [LPR13] to the usage in this work can be found in Table B.1. We obtain the following theorem:

Theorem 9 (Statistically Secure MLWE Over a Gaussian Distribution [LPR13])
Let $\mathcal{R}$ be the ring of integers in the m'th cyclotomic number field K of degree n, and $q \geq 2$ an integer. For positive integers $m \leq m + d \leq poly(n)$, let $\mathbf{A} = [\mathbf{I}_{[m]} \mid \bar{\mathbf{A}}] \in (\mathcal{R}_q)^{[m]\times[m+d]}$, where $\mathbf{I}_{[m]} \in (\mathcal{R}_q)^{[m]\times[m]}$ is the identity matrix and $\bar{\mathbf{A}} \in (\mathcal{R}_q)^{[m]\times[d]}$ is uniformly random. Then with probability $1 - 2^{-\Omega(n)}$ over the choice of $\bar{\mathbf{A}}$, the distribution of $\mathbf{Ax} \in (\mathcal{R}_q)^{[m]}$ where each coordinate of $\mathbf{x} \in (\mathcal{R}_q)^{[m+d]}$ is chosen from a discrete Gaussian distribution of parameter $s > 2n \cdot q^{m/(m+d)+2/(n(m+d))}$ over $\mathcal{R}$, satisfies that the probability of each of the q^{nm} possible outcomes is in the interval $(1 \pm 2^{-\Omega(n)})q^{-n}$ (and in particular is within statistical distance $2^{-\Omega(n)}$ of the uniform distribution over $(\mathcal{R}_q)^{[m]}$).

If a security parameter is passed and $\mathsf{sec} > n$, we raise an exepction. The resulting minimal standard deviation is stored in the instance variable `min_sigma` and the corresponding distribution can be obtained by calling `get_secret_distribution_min_width()` on the class instance.

`StatisticalUniformMLWE`. The authors of [BDL+18] describe statistically secure MLWE instances over a Uniform distribution with invertible elements. The samples $(\mathbf{A}', h_{\mathbf{A}'}(y))$ of the resulting MLWE instance are within statistical distance $2^{-\mathsf{sec}}$ of $(\mathbf{A}', \mathbf{u})$ for uniformly distributed $\mathbf{u}$.

We obtain the following theorem (for a mapping of the parameters, see Table B.2):

Theorem 10 (Statistically Secure MLWE Over a Uniform Distribution [BDL+18])
Let $1 < d_2 < n$ be a power of 2. If q is a prime congruent to $2d_2 + 1 \pmod{4d_2}$ and

$$q^{m/(m+d)} \cdot 2^{2\mathsf{sec}/((m+d)\cdot n)} \leq 2\beta < \frac{1}{\sqrt{d_2}} \cdot q^{1/d_2} \tag{4.12}$$

then any (all-powerful) algorithm $\mathcal{A}$ has advantage at most $2^{-\mathsf{sec}}$ in solving $DKS^{\infty}_{m,m+d,\beta}$, where DKS^{∞} is the decisional knapsack problem in ℓ_∞-norm.

Hence, we have:

$$\beta_{min} = \frac{q^{m/(m+d)} \cdot 2^{2\mathsf{sec}/((m+d)\cdot n)}}{2} \tag{4.13}$$

$$\beta_{max} = \frac{1}{2\sqrt{d_2}} \cdot q^{1/d_2} - 1 \tag{4.14}$$

The variable d_2 can be passed as an argument. If it is not passed, we try to find d_2 by iterating over all powers of 2 that are smaller than n. The resulting bounds are converted to ℓ_∞ and stored in the instance variables `min_beta` and `max_beta`. We also provide an instance method `get_beta_bounds()` to obtain a tuple of both.

For both statistically secure MLWE variants, we include the corresponding ring versions `StatisticalGaussianRLWE` and `StatisticalUniformRLWE` for $d = 1$ and matrix versions `StatisticalGaussianMatrixMLWE` and `StatisticalUniformMatrixMLWE` for which the width and height of the matrix $\mathbf{A}$ in [LPR13] can be passed instead of m and d.

`StatisticalMSIS`. We can find parameters for a statistically secure MSIS instance by following Section 4.1 of [DOTT21]. The mapping of the parameters in [DOTT21] is shown in Table B.3. More specifically, we ask to find a MLWE instance where the probability that non zero elements $\mathbf{r}$ in the Euclidean ball $B_m(0, 2B)$ satisfy $\hat{\mathbf{A}}_1 \cdot \mathbf{r} = \mathbf{0}$ is smaller than $2^{-\text{sec}}$.

The number of elements in $B_{m+d}(0, 2B)$ can be estimated from above as $|B_{m+d}(0, 2B)| \ll (2\pi e/((m+d)n))^{(m+d)n/2} \cdot (2B)^{(m+d)n}$. The scheme is statistically binding if the probability that non zero elements in $B_{m+d}(0, 2B)$ of radius $2B$ in $\mathcal{R}_q^{m+d}$ map to $\mathbf{0}$ in $\mathcal{R}_q^m$ is negligible. Hence, it must hold that $|B_{m+d}(0, 2B)|/q^{mn} \leq 2^{-\text{sec}}$ and we get:

$$\left(\sqrt{\frac{2\pi e}{(m+d) \cdot n}} \cdot 2B\right)^{(m+d) \cdot n} \leq 2^{-\text{sec}} \cdot q^{m \cdot n} \tag{4.15}$$

$$B \leq 2^{\frac{-\text{sec}}{(m+d) \cdot n} - 1} \cdot q^{\frac{m}{m+d}} \cdot \sqrt{\frac{(m+d) \cdot n}{2\pi e}} \tag{4.16}$$

We convert the bound B to a Gaussian over ℓ_2-norm by following the procedure described in Section 4.2:

$$s \approx x \sqrt{\frac{\pi}{(\text{sec} + 1)\ln(2)}} \tag{4.17}$$

The resulting parameters B and s can be accessed by the instance variables `max_sigma` and `max_beta` or by calling `get_secret_distribution_max_width()` on the class instance.

As for statistically secure MLWE we again include a matrix version `StatisticalMatrixMSIS` and a ring `StatisticalRSIS` by setting $d = 1$. In addition, the proof also applies to the base SIS variant and hence we include `StatisticalSIS`. Here the height of the matrix n becomes the rank of the modulus in the MSIS instance, i.e. $d = n$, and the degree of the polynomial is 1.

4.4 Parameter Search

The main functionality of our tool is to search for secure parameter sets given a set of problem instances and is encapsulated in the function `param_search.generic_search()`. It is also possible to directly estimate the cost of a list of parameter problems by calling the function `problem.estimate()`. For more details we refer to the documentation.

The high-level idea of the search is presented in Algorithm 6. We begin with an initial parameter set. We then create a list of problem instances generated by a `parameter_problem` function and estimate the cost of all instances in the list by calling `problem.estimate()`. If the list contains multiple SIS instances or multiple LWE instances, we first attempt to reduce the instances to the easiest problem instance respectively. The reduction is not exhaustive and is realized by a simple comparison of the parameters n, q, m and α or bound β according to the following intuitive hardness results. For LWE, we have that the larger n and α are and the smaller q and m are, the harder the problem becomes. For SIS, the problem becomes harder for increasing n and q and decreasing $beta$ and m.

The function then generates a list of estimation algorithm instances for each remaining problem instance and the resulting list is executed by an algorithm executor. The list is ordered according to expected runtime and output quality of the specified algorithms (see Section 4.5). If the estimation is configured to be parallel, the algorithms are split up into sublist and executed concurrently on multiple processors. Once we find an insecure problem instance (i.e. the estimated attack cost in clock cycles is smaller than 2^{sec}), we terminate the estimation procedure and `estimate()` returns a result that is labeled insecure. We then use the `next_parameters` function to generate a list L of (multiple) new parameter sets from our current parameter set and sort each of the new parameter sets into an ordered list with duplicate detection. The order is defined by a `parameter_cost` function. In the next step we retrieve the parameter set with the lowest cost from L and repeat the procedure until the cost estimation step returns a secure result. The result includes the estimates for all cost models and algorithms.

Algorithm 6: Generic search

```
function generic_search(sec, initial_params, next_parameters, parameter_cost,
  problem_instance
    )
L = OrderedList(initial_params)
while $L \neq \emptyset$ do
    current_params = L.pop()
    instances = parameter_problem(current_params)
    result = estimate(instances, sec)
    if result is secure then
        Return result
    else
        next_param_sets = next_parameters(current_params)
        forall param_set in next_param_sets do
            sort param_set into L according to parameter_cost function
```

4.5 Estimation Configuration Options

The configuration of the estimation can be customized via the class `algorithms.Configuration` which is passed as an optional argument of `param_search.generic_search()`.

We noted that the `estimate()` function can be configured to run in parallel which may speed up the search, in particular if many cost models need to be tested (e.g. with configuration setting `conservative=False` and long running algorithms like `ARORA_GB`, `CODED_BKW` and `PRIMAL_DECODE` are used. The list of used algorithms can be changed in the configuration. We recommend to include `PRIMAL_USVP` for LWE instances and `LATTICE_REDUCTION` for SIS instances to make full use of early termination since the estimate algorithms for these attacks have a short runtime and yield relatively low cost estimates.

Cost Models. Attacks that use BKZ for lattice reduction require a cost model to estimate the number of CPU cycles in the SVP subroutine. Default cost models are shown in Table 3.2. We distinguish between estimates for classical, quantum, sieving and enumeration and each of these categories can be deselected by setting the respective parameter to `False`. At least one of classical and quantum and of sieving and enumeration respectively must be selected to make use of the default cost models.

Note that classical and quantum cost models cannot be directly compared with each other. The number of operations per second that can be executed by a quantum computer may be significantly smaller than for classical computers.

If all are unselected custom cost models must be specified and passed as an argument. We included an option of taking the most conservative estimate for each category combination for a more efficient parameter search or estimation. Furthermore, we assigned a priority value on an ordinal scale to each cost model which enables us to first run cost models that yield a lower cost and thus terminate the estimation process earlier for an insecure parameter set. The priority values of the default cost models are derived from Figure 4.2. The number of BKZ rounds can be configured by passing a function with parameters `beta, d` where `beta` is the block size and `d` the lattice dimension. In the default configuration we use the more conservative Core-SVP model `algorithms.BKZ_SVP_repeat_core` [ADPS16] in which the polynomial factor of the runtime complexity of BKZ is completely ignored.

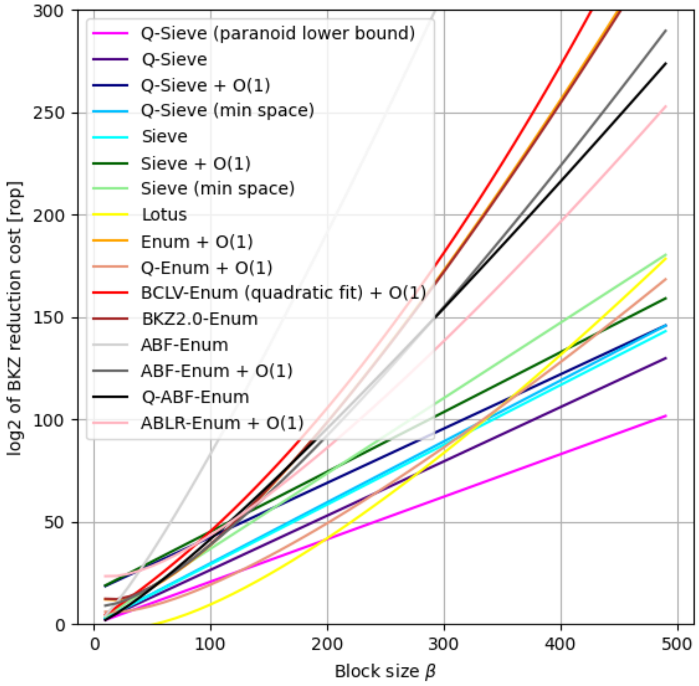

Figure 4.2: Cost models

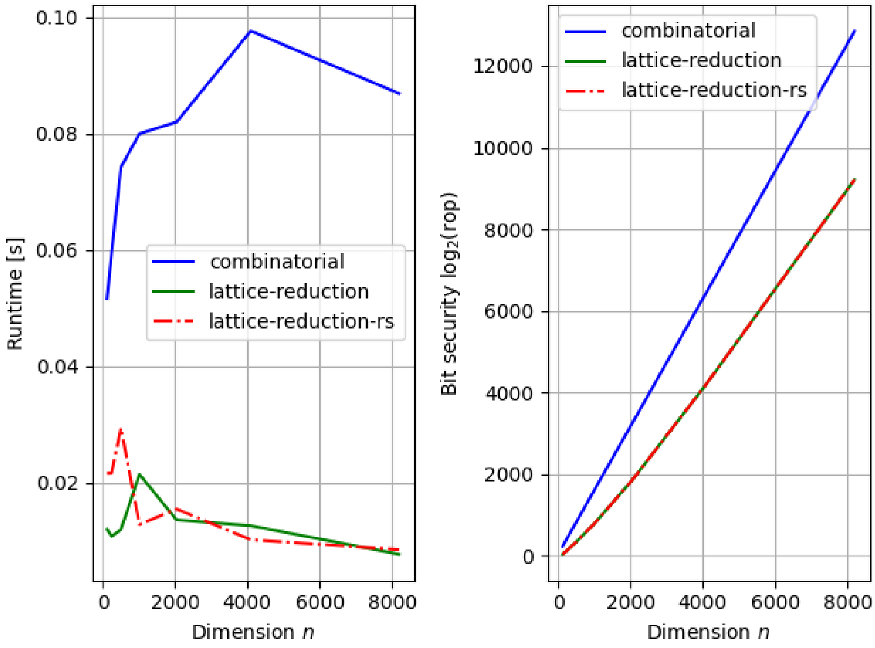

Figure 4.3: SIS instance with $\sigma = 2.828,\ m = n^2,\ 2^{2n} < q < 2^{2n+1}$

In addition, we provide a more realistic model `algorithms.BKZ_SVP_repeat_8d` for a BKZ cost of $8 \cdot d \cdot t_k$ BKZ rounds where d again referes to the lattice dimension and t_k is the number of clock cycles required for the SVP subroutine (see Line 17).

Algorithms. Figure 4.3, 4.4 and 4.5 show the plots of runtime and performance tests for the various algorithms that can be used in our tool. In accordance with the results, we assigned priority values on an ordinal scale to the estimation algorithms. In Table 4.1 and 4.2 we present the list of algorithms and their corresponding priority values and justify our choice. Algorithms with a smaller priority are expected to yield relatively good results quickly and can therefore be executed first. If the estimate result does not satisfy the specified security requirement we can terminate the estimation process early in order to maximize the efficiency of our search. Note that, while convenient, directly comparing the results for different algorithms is not always admissible as the compared algorithms may rely on different assumptions. Some algorithms may compute a more realistic cost while others may return a more conservative or even paranoid cost estimate (e.g. BKZ is used with the Core-SVP paranoid lower bound). The results have to be weighted carefully in order to guarantee the security of a given scheme in the forseable future while trying to keep the cost of using the scheme as low as possible.

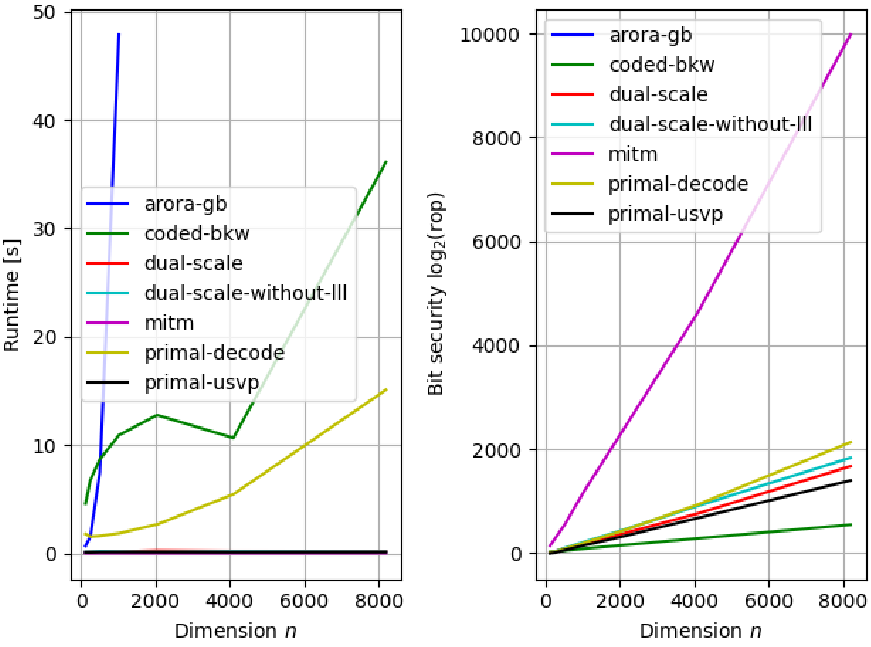

Figure 4.4: LWE instance with $\sigma = 0.125, \ m = \infty, \ 2^n < q < 2^{n+1}$

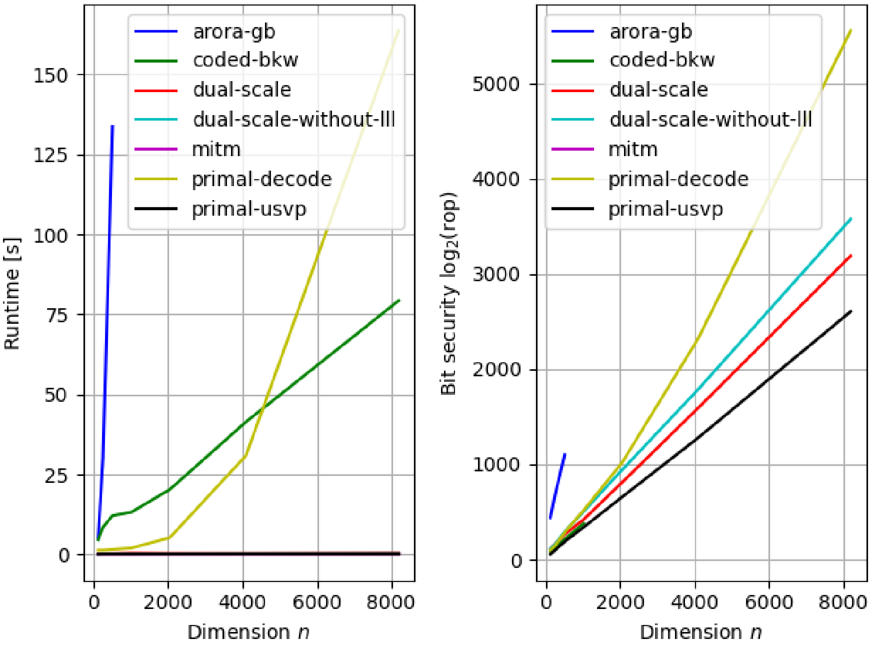

Figure 4.5: LWE instance with $\sigma = 2.828, \ m = \infty, \ 2^n < q < 2^{n+1}$

Algorithm	Priority	Justification
Meet-in-the-Middle	5	fastest, high cost estimate, as prefilter
Primal-uSVP	10	fast, low cost estimatate estimates
Dual Attack	20	fast, often higher estimates than primal-usvp
Dual Attack (without LLL)	30	fast, often higher estimates than dual
Coded-BKW	90	slow, somtimes very low cost estimate (for small stddev), does not always yield results
Decoding Attack	100	slow, often higher estimates than faster algorithms
Arora-Ge	200	extremely slow, often higher estimates, does not always yield results

Table 4.1: LWE estimate algorithm priorities

Algorithm	Priority	Justification
Lattice Reduction [MR09]	5	fastest, low cost estimates
Lattice Reduction [RS10]	7	same results as lattice-reduction, not always applicable
Combinatorial Attack	10	fast, often slightly higher cost results

Table 4.2: SIS estimate algorithm prioritie

5 Usage Examples

5.1 Two Problem Search

two problems lwe sis, what they are, how to solve it by the tool

5.2 TODO: find other schemes to apply

6 Conclusion

- short summary - tool - future work: add more algorithms, extension to specify own algorithms (already possible but not really conveniant), estimator may be adapted in near future => adapt tool - other things not part of my work but interesting

Bibliography

[ABF+20] M. R. Albrecht, S. Bai, P.-A. Fouque, P. Kirchner, D. Stehlé, W. Wen. "Faster Enumeration-Based Lattice Reduction: Root Hermite Factor $k^{1/(2k)}$ Time $k^{k/8+o(k)}$". In: *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*. Ed. by D. Micciancio, T. Ristenpart. Vol. 12171. Lecture Notes in Computer Science. Springer, 2020, pp. 186–212. URL: https://doi.org/10.1007/978-3-030-56880-1%5C_7 (cit. on pp. 31–33).

[ABLR21] M. R. Albrecht, S. Bai, J. Li, J. Rowell. "Lattice Reduction with Approximate Enumeration Oracles - Practical Algorithms and Concrete Performance". In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*. Ed. by T. Malkin, C. Peikert. Vol. 12826. Lecture Notes in Computer Science. Springer, 2021, pp. 732–759. DOI: 10.1007/978-3-030-84245-1_25. URL: https://doi.org/10.1007/978-3-030-84245-1%5C_25 (cit. on pp. 27, 31, 33).

[ACD+18] M. R. Albrecht, B. R. Curtis, A. Deo, A. Davidson, R. Player, E. W. Postlethwaite, F. Virdia, T. Wunderer. "Estimate All the {LWE, NTRU} Schemes!" In: *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*. Ed. by D. Catalano, R. D. Prisco. Vol. 11035. Lecture Notes in Computer Science. Springer, 2018, pp. 351–367. URL: https://doi.org/10.1007/978-3-319-98113-0%5C_19 (cit. on pp. 33, 49).

[ACF+15] M. R. Albrecht, C. Cid, J.-C. Faugère, R. Fitzpatrick, L. Perret. "On the complexity of the BKW algorithm on LWE". In: *Des. Codes Cryptogr.* 74.2 (2015), pp. 325–354. URL: https://doi.org/10.1007/s10623-013-9864-x (cit. on pp. 38, 40).

[ADH+19] M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, M. Stevens. "The General Sieve Kernel and New Records in Lattice Reduction". In: *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*. Ed. by Y. Ishai, V. Rijmen. Vol. 11477. Lecture Notes in Computer Science. Springer, 2019, pp. 717–746. URL: https://doi.org/10.1007/978-3-030-17656-3_25 (cit. on pp. 31, 32).

[ADPS16] E. Alkim, L. Ducas, T. Pöppelmann, P. Schwabe. "Post-quantum Key Exchange - A New Hope". In: *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*. Ed. by T. Holz, S. Savage. USENIX Association, 2016, pp. 327–343. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim (cit. on pp. 31, 33, 36, 53).

[AFFP14] M. R. Albrecht, J.-C. Faugère, R. Fitzpatrick, L. Perret. "Lazy Modulus Switching for the BKW Algorithm on LWE". In: *IACR Cryptol. ePrint Arch.* (2014), p. 19. URL: http://eprint.iacr.org/2014/019 (cit. on p. 41).

[AFG13] M. R. Albrecht, R. Fitzpatrick, F. Göpfert. "On the Efficacy of Solving LWE by Reduction to Unique-SVP". In: *Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers*. Ed. by H.-S. Lee, D.-G. Han. Vol. 8565. Lecture Notes in Computer Science. Springer, 2013, pp. 293–310. DOI: 10.1007/978-3-319-12160-4_18. URL: https://doi.org/10.1007/978-3-319-12160-4%5C_18 (cit. on p. 37).

[AG11] S. Arora, R. Ge. "New Algorithms for Learning in Presence of Errors". In: *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*. Ed. by L. Aceto, M. Henzinger, J. Sgall. Vol. 6755. Lecture Notes in Computer Science. Springer, 2011, pp. 403–415. DOI: 10.1007/978-3-642-22006-7_34. URL: https://doi.org/10.1007/978-3-642-22006-7%5C_34 (cit. on pp. 34, 41).

[AGPS20] M. R. Albrecht, V. Gheorghiu, E. W. Postlethwaite, J. M. Schanck. "Estimating Quantum Speedups for Lattice Sieves". In: *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*. Ed. by S. Moriai, H. Wang. Vol. 12492. Lecture Notes in Computer Science. Springer, 2020, pp. 583–613. URL: https://doi.org/10.1007/978-3-030-64834-3%5C_20 (cit. on p. 33).

[Ajt96] M. Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. Ed. by G. L. Miller. ACM, 1996, pp. 99–108. URL: https://doi.org/10.1145/237814.237838 (cit. on p. 17).

[Ajt98] M. Ajtai. "The Shortest Vector Problem in L_2 is *NP*-hard for Randomized Reductions (Extended Abstract)". In: *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*. Ed. by J. S. Vitter. ACM, 1998, pp. 10–19. DOI: 10.1145/276698.276705. URL: https://doi.org/10.1145/276698.276705 (cit. on p. 30).

[ANS18] Y. Aono, P. Q. Nguyen, Y. Shen. "Quantum Lattice Enumeration and Tweaking Discrete Pruning". In: *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*. Ed. by T. Peyrin, S. D. Galbraith. Vol. 11272. Lecture Notes in Computer Science. Springer, 2018, pp. 405–434. URL: https://doi.org/10.1007/978-3-030-03326-2%5C_14 (cit. on p. 31).

[APS15] M. R. Albrecht, R. Player, S. Scott. "On the concrete hardness of Learning with Errors". In: *J. Math. Cryptol.* 9.3 (2015), pp. 169–203. URL: http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml (cit. on pp. 15, 29, 30, 33, 37, 42, 46).

[AWHT16] Y. Aono, Y. Wang, T. Hayashi, T. Takagi. “Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator”. In: *Proceedings, Part I, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9665*. Berlin, Heidelberg: Springer-Verlag, 2016, pp. 789–819. ISBN: 9783662498897. URL: https://doi.org/10.1007/978-3-662-49890-3%5C_30 (cit. on p. 29).

[Bab85] L. Babai. “On Lovász’ Lattice Reduction and the Nearest Lattice Point Problem (Shortened Version)”. In: *STACS 85, 2nd Symposium of Theoretical Aspects of Computer Science, Saarbrücken, Germany, January 3-5, 1985, Proceedings*. Ed. by K. Mehlhorn. Vol. 182. Lecture Notes in Computer Science. Springer, 1985, pp. 13–20. DOI: 10.1007/BFb0023990. URL: https://doi.org/10.1007/BFb0023990 (cit. on p. 35).

[BBGS19] N. Bindel, J. Buchmann, F. Göpfert, M. Schmidt. “Estimation of the hardness of the learning with errors problem with a restricted number of samples”. In: *J. Math. Cryptol.* 13.1 (2019), pp. 47–67. DOI: 10.1515/jmc-2017-0040. URL: https://doi.org/10.1515/jmc-2017-0040 (cit. on pp. 19, 33, 37).

[BCLV17] D. J. Bernstein, C. Chuengsatiansup, T. Lange, C. van Vredendaal. “NTRU Prime: Reducing Attack Surface at Low Cost”. In: *Selected Areas in Cryptography - SAC 2017 - 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers*. Ed. by C. Adams, J. Camenisch. Vol. 10719. Lecture Notes in Computer Science. Springer, 2017, pp. 235–260. URL: https://doi.org/10.1007/978-3-319-72565-9%5C_12 (cit. on p. 33).

[BDGL16] A. Becker, L. Ducas, N. Gama, T. Laarhoven. “New directions in nearest neighbor searching with applications to lattice sieving”. In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. Ed. by R. Krauthgamer. SIAM, 2016, pp. 10–24. URL: https://doi.org/10.1137/1.9781611974331.ch2 (cit. on pp. 32, 33).

[BDL+18] C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, C. Peikert. “More Efficient Commitments from Structured Lattice Assumptions”. In: *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*. Ed. by D. Catalano, R. D. Prisco. Vol. 11035. Lecture Notes in Computer Science. Springer, 2018, pp. 368–385. URL: https://doi.org/10.1007/978-3-319-98113-0%5C_20 (cit. on pp. 17, 47, 48, 50, 73).

[BG14] S. Bai, S. D. Galbraith. “Lattice Decoding Attacks on Binary LWE”. In: *Information Security and Privacy - 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings*. Ed. by W. Susilo, Y. Mu. Vol. 8544. Lecture Notes in Computer Science. Springer, 2014, pp. 322–337. URL: https://doi.org/10.1007/978-3-319-08344-5%5C_21 (cit. on pp. 34, 36).

[BGJ15] A. Becker, N. Gama, A. Joux. “Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search”. In: *IACR Cryptol. ePrint Arch.* 2015 (2015), p. 522. URL: http://eprint.iacr.org/2015/522 (cit. on p. 32).

[BKW03] A. Blum, A. Kalai, H. Wasserman. “Noise-tolerant learning, the parity problem, and the statistical query model”. In: *J. ACM* 50.4 (2003), pp. 506–519. URL: https://doi.org/10.1145/792538.792543 (cit. on p. 38).

[BLP+13] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, D. Stehlé. "Classical hardness of learning with errors". In: *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. Ed. by D. Boneh, T. Roughgarden, J. Feigenbaum. ACM, 2013, pp. 575–584. URL: https://doi.org/10.1145/2488608.2488680 (cit. on p. 15).

[BLS16] S. Bai, T. Laarhoven, D. Stehlé. "Tuple lattice sieving". In: *IACR Cryptol. ePrint Arch.* (2016), p. 713. URL: http://eprint.iacr.org/2016/713 (cit. on p. 32).

[BS99] J. Blömer, J.-P. Seifert. "On the Complexity of Computing Short Linearly Independent Vectors and Short Bases in a Lattice". In: *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*. STOC '99. Atlanta, Georgia, USA: Association for Computing Machinery, 1999, pp. 711–720. ISBN: 1581130678. URL: https://doi.org/10.1145/301250.301441 (cit. on p. 20).

[Che13] Y. Chen. "Réduction de réseau et sécurité concrete du chiffrement completement homomorphe". PhD thesis. Paris 7, 2013 (cit. on pp. 29–31, 33).

[CN11] Y. Chen, P. Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates". In: *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*. Ed. by D. H. Lee, X. Wang. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 1–20. URL: https://doi.org/10.1007/978-3-642-25385-0%5C_1 (cit. on pp. 29, 33).

[DOTT21] I. Damgård, C. Orlandi, A. Takahashi, M. Tibouchi. "Two-Round n-out-of-n and Multi-signatures and Trapdoor Commitment from Lattices". In: *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I*. Ed. by J. A. Garay. Vol. 12710. Lecture Notes in Computer Science. Springer, 2021, pp. 99–130. URL: https://doi.org/10.1007/978-3-030-75245-3%5C_5 (cit. on pp. 51, 73).

[DPSZ12] I. Damgård, V. Pastro, N. P. Smart, S. Zakarias. "Multiparty Computation from Somewhat Homomorphic Encryption". In: *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. Ed. by R. Safavi-Naini, R. Canetti. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 643–662. URL: https://doi.org/10.1007/978-3-642-32009-5%5C_38 (cit. on pp. 47, 48).

[DSW21] L. Ducas, M. Stevens, W. P. J. van Woerden. "Advanced Lattice Sieving on GPUs, with Tensor Cores". In: *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*. Vol. 12697. Lecture Notes in Computer Science. Springer, 2021, pp. 249–279. DOI: 10.1007/978-3-030-77886-6_9. URL: https://doi.org/10.1007/978-3-030-77886-6%5C_9 (cit. on p. 31).

[GJS15] Q. Guo, T. Johansson, P. Stankovski. "Coded-BKW: Solving LWE Using Lattice Codes". In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*. Ed. by R. Gennaro, M. Robshaw. Vol. 9215. Lecture Notes in Computer Science. Springer, 2015, pp. 23–42. URL: https://doi.org/10.1007/978-3-662-47989-6%5C_2 (cit. on pp. 18, 21, 22, 38, 40, 45, 71).

[GN08] N. Gama, P. Q. Nguyen. “Predicting Lattice Reduction”. In: *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*. Ed. by N. P. Smart. Vol. 4965. Lecture Notes in Computer Science. Springer, 2008, pp. 31–51. URL: https://doi.org/10.1007/978-3-540-78967-3%5C_3 (cit. on p. 30).

[GNR10] N. Gama, P. Q. Nguyen, O. Regev. “Lattice Enumeration Using Extreme Pruning”. In: *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*. Ed. by H. Gilbert. Vol. 6110. Lecture Notes in Computer Science. Springer, 2010, pp. 257–278. URL: https://doi.org/10.1007/978-3-642-13190-5%5C_13 (cit. on p. 29).

[Göp16] F. Göpfert. “Securely Instantiating Cryptographic Schemes Based on the Learning with Errors Assumption”. PhD thesis. Darmstadt University of Technology, Germany, 2016. URL: http://tuprints.ulb.tu-darmstadt.de/5850/ (cit. on pp. 19, 27, 37).

[GPV08] C. Gentry, C. Peikert, V. Vaikuntanathan. “Trapdoors for hard lattices and new cryptographic constructions”. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*. Ed. by C. Dwork. ACM, 2008, pp. 197–206. URL: https://doi.org/10.1145/1374376.1374407 (cit. on p. 21).

[HK17] G. Herold, E. Kirshanova. “Improved Algorithms for the Approximate k-List Problem in Euclidean Norm”. In: *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I*. Ed. by S. Fehr. Vol. 10174. Lecture Notes in Computer Science. Springer, 2017, pp. 16–40. DOI: 10.1007/978-3-662-54365-8_2. URL: https://doi.org/10.1007/978-3-662-54365-8%5C_2 (cit. on p. 32).

[HPS11] G. Hanrot, X. Pujol, D. Stehlé. “Analyzing Blockwise Lattice Algorithms Using Dynamical Systems”. In: *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*. Ed. by P. Rogaway. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 447–464. URL: https://doi.org/10.1007/978-3-642-22792-9%5C_25 (cit. on pp. 29, 30).

[HS07] G. Hanrot, D. Stehlé. “Improved Analysis of Kannan’s Shortest Lattice Vector Algorithm”. In: *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*. Ed. by A. Menezes. Vol. 4622. Lecture Notes in Computer Science. Springer, 2007, pp. 170–186. DOI: 10.1007/978-3-540-74143-5_10. URL: https://doi.org/10.1007/978-3-540-74143-5%5C_10 (cit. on p. 31).

[Kan83] R. Kannan. “Improved Algorithms for Integer Programming and Related Lattice Problems”. In: *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*. Ed. by D. S. Johnson, R. Fagin, M. L. Fredman, D. Harel, R. M. Karp, N. A. Lynch, C. H. Papadimitriou, R. L. Rivest, W. L. Ruzzo, J. I. Seiferas. ACM, 1983, pp. 193–206. DOI: 10.1145/800061.808749. URL: https://doi.org/10.1145/800061.808749 (cit. on p. 31).

[Kan87] R. Kannan. “Minkowski’s Convex Body Theorem and Integer Programming”. In: *Math. Oper. Res.* 12.3 (1987), pp. 415–440. DOI: 10.1287/moor.12.3.415. URL: https://doi.org/10.1287/moor.12.3.415 (cit. on p. 36).

[Kho05] S. Khot. “Hardness of approximating the shortest vector problem in lattices”. In: *J. ACM* 52.5 (2005), pp. 789–808. DOI: 10.1145/1089023.1089027. URL: https://doi.org/10.1145/1089023.1089027 (cit. on pp. 20, 30).

[Laa16] T. Laarhoven. “Search problems in cryptography: from fingerprinting to lattice sieving”. English. Proefschrift. PhD thesis. Mathematics and Computer Science, Feb. 2016. ISBN: 978-90-386-4021-1 (cit. on pp. 31, 33).

[LLL82] A. Lenstra, H. Lenstra, L. Lovász. “Factoring Polynomials with Rational Coefficients”. In: *Mathematische Annalen* 261 (Dec. 1982). URL: https://doi.org/10.1007/BF01457454 (cit. on pp. 28, 29, 31).

[LM09] V. Lyubashevsky, D. Micciancio. “On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem”. In: *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*. Ed. by S. Halevi. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 577–594. DOI: 10.1007/978-3-642-03356-8_34. URL: https://doi.org/10.1007/978-3-642-03356-8%5C_34 (cit. on pp. 20, 37).

[LN20] J. Li, P. Q. Nguyen. “A Complete Analysis of the BKZ Lattice Reduction Algorithm”. In: *IACR Cryptol. ePrint Arch.* (2020), p. 1237. URL: https://eprint.iacr.org/2020/1237 (cit. on p. 32).

[Lov87] L. Lovasz. *An Algorithmic Theory of Numbers, Graphs and Convexity*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1987. ISBN: 9780898712032. URL: https://books.google.com/books?id=sJ3mBHTU55QC (cit. on p. 37).

[LP11] R. Lindner, C. Peikert. “Better Key Sizes (and Attacks) for LWE-Based Encryption”. In: *Topics in Cryptology - CT-RSA 2011 - The Cryptographers’ Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*. Ed. by A. Kiayias. Vol. 6558. Lecture Notes in Computer Science. Springer, 2011, pp. 319–339. URL: https://doi.org/10.1007/978-3-642-19074-2%5C_21 (cit. on pp. 27, 35, 36).

[LPR13] V. Lyubashevsky, C. Peikert, O. Regev. “A Toolkit for Ring-LWE Cryptography”. In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. Ed. by T. Johansson, P. Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 35–54. URL: https://doi.org/10.1007/978-3-642-38348-9%5C_3 (cit. on pp. 49, 50, 73).

[LS15] A. Langlois, D. Stehlé. “Worst-case to average-case reductions for module lattices”. In: *Des. Codes Cryptogr.* 75.3 (2015), pp. 565–599. URL: https://doi.org/10.1007/s10623-014-9938-4 (cit. on pp. 23, 24).

[Mar21] Marc. *Volume q^n of a dual q-ary lattice in MR09*. Cryptogrpahy Stack Exchange. (user: https://crypto.stackexchange.com/users/45690/mark). 2021. URL: https://crypto.stackexchange.com/a/95368 (visited on 09/30/2021) (cit. on p. 19).

[MG02] D. Micciancio, S. Goldwasser. *Complexity of lattice problems - a cryptograhic perspective*. Vol. 671. The Kluwer international series in engineering and computer science. Springer, 2002. ISBN: 978-0-7923-7688-0 (cit. on p. 18).

[MP13] D. Micciancio, C. Peikert. "Hardness of SIS and LWE with Small Parameters". In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*. Ed. by R. Canetti, J. A. Garay. Vol. 8042. Lecture Notes in Computer Science. Springer, 2013, pp. 21–39. URL: https://doi.org/10.1007/978-3-642-40041-4%5C_2 (cit. on p. 15).

[MR04] D. Micciancio, O. Regev. "Worst-Case to Average-Case Reductions Based on Gaussian Measures". In: *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*. IEEE Computer Society, 2004, pp. 372–381. URL: https://doi.org/10.1109/FOCS.2004.72 (cit. on p. 23).

[MR09] D. Micciancio, O. Regev. "Lattice-based Cryptography". In: *Post-Quantum Cryptography*. Ed. by D. J. Bernstein, J. Buchmann, E. Dahmen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 147–191. ISBN: 978-3-540-88702-7. URL: https://doi.org/10.1007/978-3-540-88702-7_5 (cit. on pp. 40–43, 56).

[MV10] D. Micciancio, P. Voulgaris. "Faster Exponential Time Algorithms for the Shortest Vector Problem". In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*. Ed. by M. Charikar. SIAM, 2010, pp. 1468–1480. URL: https://doi.org/10.1137/1.9781611973075.119 (cit. on p. 32).

[MW16] D. Micciancio, M. Walter. "Practical, Predictable Lattice Basis Reduction". In: *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*. Ed. by M. Fischlin, J.-S. Coron. Vol. 9665. Lecture Notes in Computer Science. Springer, 2016, pp. 820–849. URL: https://doi.org/10.1007/978-3-662-49890-3%5C_31 (cit. on p. 31).

[Nor12] Norbert. *Relations between p norms*. Mathematics Stack Exchange. (user: https://math.stackexchange.com/users/19538/norbert). 2012. URL: https://math.stackexchange.com/q/218129 (visited on 04/13/2017) (cit. on p. 47).

[NS05] P. Q. Nguên, D. Stehlé. "Floating-Point LLL Revisited". In: *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*. EUROCRYPT'05. Aarhus, Denmark: Springer-Verlag, 2005, pp. 215–233. ISBN: 3540259104. URL: https://doi.org/10.1007/11426639%5C_13 (cit. on pp. 28, 29).

[NV08] P. Q. Nguyen, T. Vidick. "Sieve algorithms for the shortest vector problem are practical". In: *J. Math. Cryptol.* 2.2 (2008), pp. 181–207. DOI: 10.1515/JMC.2008.009. URL: https://doi.org/10.1515/JMC.2008.009 (cit. on p. 32).

[NV10] P. Q. Nguyen, B. Vallée, eds. *The LLL Algorithm - Survey and Applications*. Information Security and Cryptography. Springer, 2010. ISBN: 978-3-642-02294-4. URL: https://doi.org/10.1007/978-3-642-02295-1 (cit. on p. 17).

[Pei09] C. Peikert. “Public-key cryptosystems from the worst-case shortest vector problem: extended abstract”. In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*. Ed. by M. Mitzenmacher. ACM, 2009, pp. 333–342. URL: https://doi.org/10.1145/1536414.1536461 (cit. on p. 22).

[Pei16] C. Peikert. “A Decade of Lattice Cryptography”. In: *Found. Trends Theor. Comput. Sci.* 10.4 (2016), pp. 283–424. URL: https://doi.org/10.1561/0400000074 (cit. on p. 18).

[PHAM17] L. T. Phong, T. Hayashi, Y. Aono, S. Moriai. “Lotus”. In: *Technical report, National Institute of Standards and Technology*. 2017. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions (cit. on p. 33).

[Reg04a] O. Regev. *Lecture notes in Lattices in Computer Science*. Fall 2004. URL: https://cims.nyu.edu/~regev/teaching/lattices%5C_fall_2004/ln/lll.pdf (cit. on p. 28).

[Reg04b] O. Regev. *Lecture notes in Lattices in Computer Science*. Fall 2004. URL: https://cims.nyu.edu/~regev/teaching/lattices%5C_fall%5C_2004/ln/cvp.pdf (cit. on p. 35).

[Reg05] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. Ed. by H. N. Gabow, R. Fagin. ACM, 2005, pp. 84–93. URL: https://doi.org/10.1145/1060590.1060603 (cit. on pp. 15, 18, 22).

[Reg09] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *J. ACM* 56.6 (2009), 34:1–34:40. URL: http://doi.acm.org/10.1145/1568318.1568324 (cit. on p. 22).

[Reg10] O. Regev. “The Learning with Errors Problem (Invited Survey)”. In: *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*. IEEE Computer Society, 2010, pp. 191–204. URL: https://doi.org/10.1109/CCC.2010.26 (cit. on pp. 21, 23).

[RS10] M. Rückert, M. Schneider. “Estimating the Security of Lattice-based Cryptosystems”. In: *IACR Cryptol. ePrint Arch.* 2010 (2010), p. 137. URL: http://eprint.iacr.org/2010/137 (cit. on pp. 43, 56).

[SAL+17] N. P. Smart, M. R. Albrecht, Y. Lindell, E. Orsini, V. Osheter, K. Paterson, G. Peer. “Lima”. In: *Technical report, National Institute of Standards and Technology*. 2017. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions (cit. on p. 33).

[Sch03] C.-P. Schnorr. “Lattice Reduction by Random Sampling and Birthday Methods”. In: *STACS 2003, 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27 - March 1, 2003, Proceedings*. Ed. by H. Alt, M. Habib. Vol. 2607. Lecture Notes in Computer Science. Springer, 2003, pp. 145–156. DOI: 10.1007/3-540-36494-3_14. URL: https://doi.org/10.1007/3-540-36494-3%5C_14 (cit. on p. 27).

[SE91] C.-P. Schnorr, M. Euchner. “Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems”. In: *Fundamentals of Computation Theory, 8th International Symposium, FCT ’91, Gosen, Germany, September 9-13, 1991, Proceedings*. Ed. by L. Budach. Vol. 529. Lecture Notes in Computer Science. Springer, 1991, pp. 68–85. URL: https://doi.org/10.1007/3-540-54458-5%5C_51 (cit. on pp. 29, 30).

[SFS08] N. Sommer, M. Feder, O. Shalvi. “Low-Density Lattice Codes”. In: *IEEE Trans. Inf. Theory* 54.4 (2008), pp. 1561–1585. DOI: 10.1109/TIT.2008.917684. URL: https://doi.org/10.1109/TIT.2008.917684 (cit. on p. 71).

[SHRS17] J. M. Schanck, A. Hulsing, J. Rijneveld, P. Schwabe. “NTRU-HRSS-KEM”. In: *Technical report, National Institute of Standards and Technology*. 2017. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions (cit. on p. 33).

[Van12] J. H. Van Lint. *Introduction to coding theory*. Vol. 86. Springer Science & Business Media, 2012 (cit. on p. 71).

All links were last followed on October 1, 2021.

A Additional Math

A.1 Linear Codes

Let $\mathbb{F}_q^n$ be the n-dimensional vector space over the field $\mathbb{F}_q$. A q-ary linear code C or $[n, k]$-code [Van12] is a k-dimensional linear subspace of $\mathbb{F}_q^n$ such that $\mathbf{0} \in C$, $\mathbf{x} + \mathbf{y} \in C$ for all $\mathbf{x}, \mathbf{y} \in C$ and $\gamma\mathbf{x} \in C$ for $\mathbf{x} \in C, \gamma \in \mathbb{F}_q$. There are q^k different codewords in C.

Let C be a q-ary linear $[n, k]$-code. The lattice over C [GJS15] is defined as

$$\Lambda(C) = \{\mathbf{x} \in \mathbb{R}^n \mid \exists \mathbf{y} \in C : \mathbf{x} = \mathbf{y} \mod q\} . \tag{A.1}$$

Similarly, for a lattice $\Lambda(\mathbf{B})$ a lattice code C defined by $\Lambda(\mathbf{B})$ and a shaping region $\mathcal{V} \subset \mathbb{R}^n$ (e.g. the Voronoi region, see Equation (2.5)) is a subspace of $\mathbb{R}^n$ such that all codewords are lattice vectors in $\Lambda(\mathbf{B})$ within the region $\mathcal{V}$ [SFS08]:

$$C = \{x \in \Lambda(\mathbf{B}) \mid x \in \mathcal{V}\} . \tag{A.2}$$

B Tables

B.1 Mapping of Parameters in Referenced Papers

Parameters in [LPR13]	Noation here	Represents
l	$m + d$	width of matrix $\mathbf{A}$
k	m	height of matrix $\mathbf{A}$

Table B.1: Parameter mapping from [LPR13]

Parameters in [BDL+18]	Noation Here	Represents
k	$m + d$	width of matrix $[\mathbf{I}_n\ \mathbf{A}']$
n	m	height of matrix $[\mathbf{I}_n\ \mathbf{A}']$
d	d_2	variable
N	n	degree of Ring polynomial

Table B.2: Parameter mapping from [BDL+18]

Parameters in [DOTT21]	Noation Here	Represents
m'	$m + d$	width of matrix $\hat{\mathbf{A}}_1$
m	m	height of matrix $\hat{\mathbf{A}}_1$
B	B	norm-bound of secret
s	s	Gaussian width (not stddev)
N	n	degree of polynomial

Table B.3: Parameter mapping from [DOTT21]

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature