Lattice-based Cryptography

A survey on the security of the lattice-based NIST finalists

Koen de Boer[1] and Wessel van Woerden[2]

[1]Mathematical Institute, Leiden University, The Netherlands
[2]Univ. Bordeaux, CNRS, Inria, Bordeaux INP, IMB, Talence, France
[2]PQShield, `wessel.vanwoerden@pqshield.com`

February 20, 2025

Contents

Preface

This survey, mostly written in the years 2022-2023, is meant as an as short as possible description of the current state-of-the-art lattice attacks on lattice-based cryptosystems, without losing the essence of the matter.

The main focus is the security of the NIST finalists and alternatives that are *based on lattices*, namely CRYSTALS-Kyber, CRYSTALS-Dilithium and Falcon (see Table 1). Instead of going through these cryptosystems case by case, this survey considers attacks on the underlying hardness assumptions: in the case of the mentioned lattice-based schemes, these are (variants of) LWE (Learning With Errors) and NTRU.

NIST: selected algorithms for standardization	**Assumption**
CRYSTALS-Kyber	Module-LWE
CRYSTALS-Dilithium	Module-LWE
Falcon	NTRU
SPHINCS$^+$	Hash-based
NIST: candidates moved to the fourth round	**Assumption**
BIKE	QC-MDPC code-based
Classic McEliece	Code-based
HQC	QC-MDPC code-based
SIKE	Isogeny-based

Table 1: The NIST algorithms, selected for standardization and fourth round finalists, and their underlying cryptographic assumptions. In this survey we will consider only the first three, as those are based on lattices.

Overview of this survey

As fields of expertise differ from reader to reader, we would like to give a quick overview of the topics that are treated in this survey. Additionally this overview is also meant to navigate within the survey; it allows to skim certain parts and to concentrate more actively on other parts.

Chapter 1 is an introduction that gives a rough, intuitive idea how lattice-based cryptography works and where it is built on.

Part I In Chapter 2, Sections 2.1 to 2.3 are about lattices, bases, fundamental domains, relevant computational problems, projection and orthogonalization. These sections can be considered as preliminaries.

Sections 2.5 to 2.9 are about so-called *basis reduction algorithms*, whose, given a basis of a lattice, attempt to find a *new* basis of such lattice with better qualities. Finding such a good quality basis is one of the ingredients for attacks on lattice-based cryptosystems.

These basis-reduction algorithms in a rather large-dimensional basis of a lattice require the ability to find *exact* short vectors in lower dimensional lattices. How to find such short vectors is treated in Sections 2.10 to 2.12. To summarize, in Chapter 2 essentially all attacks on lattice-based cryptography are roughly treated.

Chapter 3 is about structured lattices, which are lattices based on number rings. These number rings are succinctly treated in Section 3.2 and ideal lattices and module lattices are defined in Sections 3.3 and 3.4.

Next, in Chapter 4, the computational problems NTRU and LWE are introduced. The 'plain' variants of these are explained in Section 4.2 (LWE) and Section 4.4 (NTRU), where the precise definitions are in Sections 4.2.5 and 4.4.5. The other sections (Sections 4.3 and 4.5) show the structured variants of LWE and the 'unstructured' variant of NTRU; and shows how to transform structured LWE into unstructured LWE.

Part II: Unstructured Attacks In Chapter 5 we concentrate on asymptotic estimates of the most efficient attacks on cryptosystems. We start by showing that LWE and NTRU can indeed be adequately phrased as lattice problems (Section 5.1) and proceed by describing the variety of attacks in Section 5.2: the primal, dual and hybrid attack; and the dense sublattice attack. Various estimates and quick summaries are given. A small text about Arora-Ge and BKZ is added as well (Section 5.2.5). We finish this chapter with showing how the NIST candidates get their 'beta' estimate (Section 5.3).

In Chapter 6 is zoomed in on the *concrete costs* of the attacks, how (non-asymptotic) costs are generally measured on hardware and quantum computers, the concrete costs of enumeration, sieving and BKZ (Sections 6.1 to 6.3). We finish by a text about the concrete estimates of the NIST candidates (Section 6.4).

Part III: Structured Attacks In Chapter 7 an explanation is given of an attack on ideal lattices and a reduction for module lattices. The attack on ideal lattices is first explained for cyclotomic fields (Section 7.3), and later for general fields (Section 7.4). In the last section, Section 7.5, is shown what role quantum computation plays in this attack. In Chapter 8, a reduction from higher rank modules to rank-2 modules is treated. For module lattices, no attacks better than the 'generic' attacks are known.

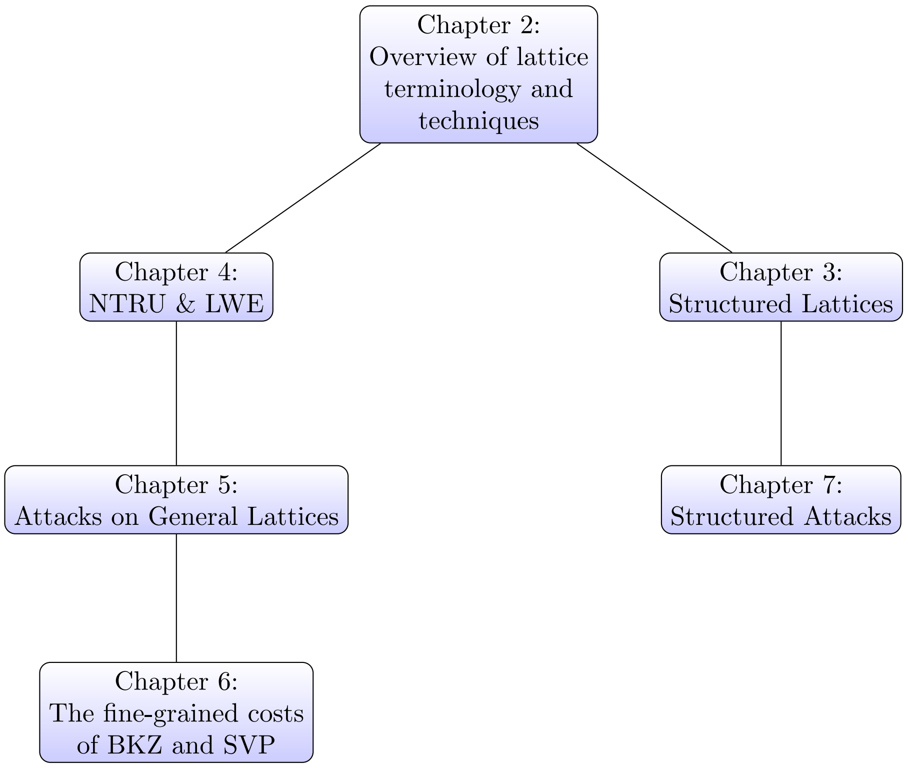

Chapter 2:
Overview of lattice terminology and techniques
Chapter 4:
NTRU & LWE
Chapter 3:
Structured Lattices
Chapter 5:
Attacks on General Lattices
Chapter 7:
Structured Attacks
Chapter 6:
The fine-grained costs of BKZ and SVP

Figure 1: In this diagram is depicted which two main reading routes can be followed; the left part is the 'unstructured route' which mainly treats general lattices and attacks thereof. The right part is the 'structured route', which is more focused on structured lattices (module and ideal lattices) and specific structured attacks.

Appendix In the appendix, a short explanation is given about how to use the state-of-the-art scripts that estimate the bit security of lattice-based cryptosystems by considering all known unstructured attacks.

Acknowledgments

The work of the first author was funded by the AIVD (the General Intelligence and Security Service of the Netherlands) with the aim of stimulating further work on the security foundations of post-quantum cryptography standards. The second author performed an initial part of this work while employed in the Cryptology Group at CWI in Amsterdam and the remainder while employed at the University of Bordeaux with support of the CHARM ANR-NSF grant (ANR-21-CE94-0003). We thank Ronald Cramer (CWI Cryptology Group and Mathematical Institute, Leiden University) for his contributions to the initiation of this project and to its scoping.

Part I

Lattice terminology

Chapter 1

Introduction

Lattices are, among others, one of the most promising mathematical objects to base cryptography on. Indeed, some of the NIST candidates for standardization for post-quantum cryptography are based on lattices.

In this introduction we wish to give you an intuition on how lattice-based cryptography works and what the underlying assumptions are. For simplicity, we choose to explain a simple public-key cryptosystem based on lattices.

Public-key cryptosystem

A public-key cryptosystem allows for an individual, say Alice, to securely send a message over an untrusted channel to another individual, say, Bob, in such a way that only Bob can read the message.

The message Alice wants to send is encrypted using a public key (pk) that Bob had announced publicly before. She then sends the encrypted message $Enc_{pk}(message)$ over the insecure channel to Bob, who then decrypts the encrypted message using his secret key (sk), see Figure 1.1.

Ignoring for the moment the inner workings of the encryption and decryption algorithms, it is of fundamental importance for the security of the cryptosystem

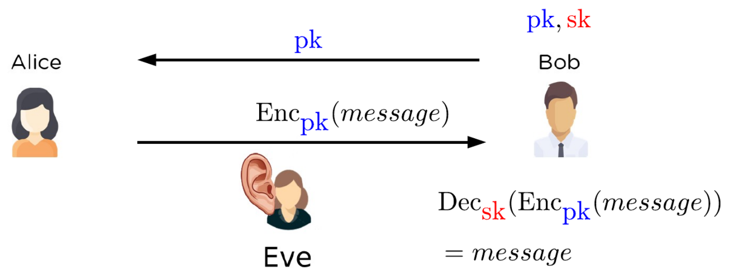


Figure 1.1: A public-key cryptosystem allows Alice to securely send a message to Bob over a untrusted channel which has a possible eavesdropper Eve.

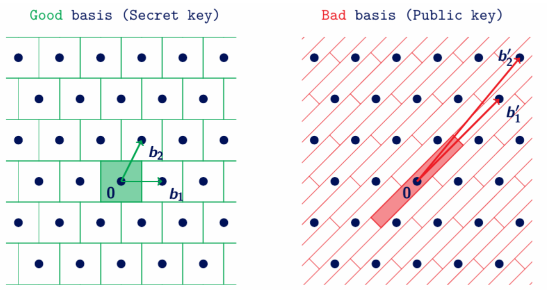


Figure 1.2: A good basis of a lattice is short and somewhat orthogonal, which allows to round a point nearby a lattice point successfully. A bad basis has long vectors that are not orthogonal and does not allow for rounding successfully.

that the secret key is known only to Bob and that it cannot be (computationally) recovered from the information that has been transported over the channel, e.g., the public key.

Good basis and bad basis

In the lattice-based public-key toy cryptosystem that is explained here, the public key will consist of a bad basis of a lattice, whereas the secret key will consist of a good basis of the same lattice.

Fundamental to the cryptosystem will be the fact that with a good basis one can 'round' a point nearby a lattice point successfully to that lattice point, whereas with a bad basis one cannot. More specifically, any point in a green square of the leftmost picture of Figure 1.2 (with the good basis) will be rounded successfully to the associated lattice point. Using the same rounding algorithm with a bad basis will round a point near a lattice point mostly to a far away lattice point; as the red squares in the rightmost picture of Figure 1.2 indicate.

Encryption

The fact that only a good basis allows for efficient and successful rounding suggests an encryption mechanism. Suppose Alice's message can be encoded in a lattice point[1]; then adding a small error to this lattice point hides this original lattice point, see Figure 1.3.

Decryption

As only a good basis can 'round' well, and Bob is the only one that knows the good basis, he can efficiently and successfully recover the original lattice point Alice intended to sent, see Figure 1.4.

[1]One can, for example, publicly agree beforehand to label some lattice points with letters and repeat the procedure to transfer a sentence.

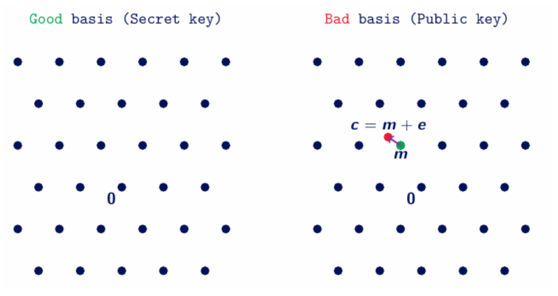


Figure 1.3: Using the public key, Alice can choose a lattice point to send to Bob, and adds a small error to hide the lattice point for eavesdroppers.

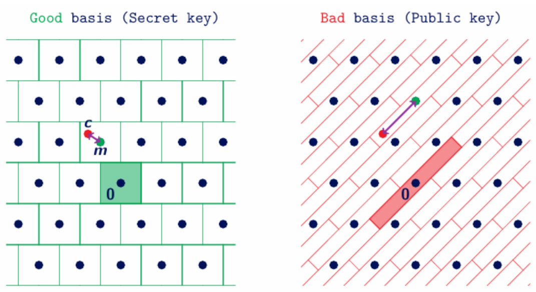


Figure 1.4: Using the secret key, the good basis, Bob can efficiently round the disturbed point to its original lattice point, and recovers the message Alice intended to sent. A bad basis does not allow to round successfully and thus someone other than Bob cannot recover Alice's message.

Underlying hardness assumption

As already noted, the security of this public-key cryptosystem relies on the hardness of recovering the secret key (i.e., the good basis) from the public key (i.e., the bad basis). In other words, in order for this cryptosystem to be secure, computing a short good basis of a lattice from a bad basis of a lattice should be hard. More generally, rounding successfully should be hard given only the bad basis.

We have not yet explained how one obtains such a key-pair. This is precisely where common hardness assumptions such as LWE and NTRU come in. Intuitively, these assumptions allow one to randomly create a lattice along with a (partial) good basis. The randomness of the lattice is required as otherwise an attacker could just repeat the generation procedure to obtain the good basis.

Chapter 2

Overview of lattice terminology and techniques

2.1 Notation

We denote by $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ the natural numbers, the integers, the rationals, the real numbers and the complex numbers respectively. For $k \in \mathbb{N}$ we will denote $[k] = \{1, \ldots, k\}$. We denote a vector space V in uppercase. Vectors $\mathbf{v}$ in V are always considered column-wise and denoted in lowercase bold, and matrices $\mathbf{M} = (\mathbf{m}_1, \ldots, \mathbf{m}_n)$ in uppercase bold with columns $\mathbf{m}_i$. We will consistently denote n for the lattice dimension, r for the module-lattice rank and t for the number field degree.

We use the standard Landau-notation $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$ for asymptotic behavior of functions, and occasionally the 'soft version' $\tilde{O}(\cdot)$ of the Big O, in which polylogarithmic factors of the argument are hidden.

2.2 Lattices

2.2.1 Introduction

Intuitively, lattices can be considered as 'discrete' analogues of vector spaces[1]. So, as real vector spaces V can be seen as the $\mathbb{R}$-span of certain basis vectors $\{\mathbf{b}_j \mid j \in [n]\}$,

$$V = \Big\{ \sum_{j=1}^{n} c_j \mathbf{b}_j \mid c_j \in \mathbb{R} \text{ for all } j \in [n] \Big\},$$

lattices $\mathcal{L} \subseteq V$ can be considered as the $\mathbb{Z}$-span of such basis vectors:

$$\mathcal{L} = \Big\{ \sum_{j=1}^{n} c_j \mathbf{b}_j \mid c_j \in \mathbb{Z} \text{ for all } j \in [n] \Big\}. \tag{2.1}$$

[1]Throughout this survey, we will only consider finite-dimensional vector spaces.

Mostly (especially in cryptography) one is particularly interested in certain *geometric properties* of the lattice, for example the length of the shortest non-zero vector in $\mathcal{L}$. For this reason, one requires a well-behaved *length notion* on the vector space V, which is then to be inherited by the lattice $\mathcal{L}$. More precisely, one would like the vector space to have an *inner product*; such vector spaces of finite dimension over the real numbers enriched with a inner product are called *Euclidean vector spaces.*

2.2.2 Lattices and Euclidean vector spaces

So, before we introduce lattices, we need to define their ambient space, which is a Euclidean vector space.

Definition 1 (Euclidean space)**.** *A Euclidean vector space V is a finite-dimensional inner product space over the real numbers together with an inner product $\langle \cdot, \cdot \rangle$, i.e., a positive definite symmetric bi-linear form.*

Such an inner product space has an induced norm, given by $\|\mathbf{v}\| := \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$, which gives a length notion on V. Throughout this text we will always denote V for the Euclidean vector space where the lattices live in, and its associated norm with $\|\cdot\|$.

The norm on V defines a metric (a length notion) on V, in which we can speak of discreteness. A subset $S \subseteq V$ is called uniformly discrete if there exists some distance $\delta > 0$ so that every two different points $s, s' \in S$ have at least this distance δ away from each other. More formally,

$$\inf\{\|s - s'\| \mid s, s' \in S, s \neq s'\} > \delta > 0.$$

In other words, a uniformly discrete set in V has a notion of 'minimum distance'. Later, this notion comes back for lattices in the form of the 'minimum length'.

Definition 2 (Lattice)**.** *A lattice $\mathcal{L}$ is a (uniformly) discrete*[2] *additive subgroup of a Euclidean vector space V.*

To elaborate on this definition, a lattice $\mathcal{L} \subseteq V$ is thus a uniformly discrete subset of V (i.e., has a 'minimum distance'). Moreover, since the vector space V has an addition operation (adding vectors), we require $\mathcal{L}$ to be a subgroup of V with respect to this inherited addition. Concretely, we demand $\mathbf{x}, \mathbf{y} \in \mathcal{L} \Rightarrow \mathbf{x} - \mathbf{y} \in \mathcal{L}$ (and thus $\mathbf{0} \in \mathcal{L}$).

2.2.3 Lattice Bases

A lattice $\mathcal{L}$ is free[3] as a $\mathbb{Z}$-module; this precisely means that it has a $\mathbb{Z}$-basis, exactly as in Equation (2.1). More precisely:

[2] For a subgroup of V, uniformly discrete and discrete are equivalent, by the homogeneity of the space V.

[3] This follows from the fact that lattices have no torsion: there are no elements $\mathbf{v} \in \mathcal{L}$ or $n \in \mathbb{N}_{>0}$ such that $\underbrace{\mathbf{v} + \ldots + \mathbf{v}}_{n \text{ times}} = 0$.

Lemma 1 (Basis)**.** *Every lattice $\mathcal{L}$ is a free module over $\mathbb{Z}$, and therefore has a $\mathbb{Z}$-basis $\boldsymbol{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$. In other words,*

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} c_i \mathbf{b}_i \mid c_i \in \mathbb{Z} \right\}.$$

In particular, due to the discreteness of the lattice $\mathcal{L}$, the vectors $(\mathbf{b}_1, \dots, \mathbf{b}_n)$ are $\mathbb{R}$-linearly independent. The number of basis vectors n is called the rank $\operatorname{rk}(\mathcal{L})$ of the lattice.

We call a lattice $\mathcal{L} \subset V$ full-rank if its rank $\operatorname{rk}(\mathcal{L})$ equals the dimension of the Euclidean space V.

Lemma 1 implies that we could alternatively define lattices by means of a basis. Bases play a central role in lattice algorithms, as they are the main tool to compute with. So, from now on, we always describe lattices in terms of a basis. A basis of a lattice is not unique; in fact, if $\mathbf{B}$ is a basis of $\mathcal{L}$, then also $\mathbf{BU}$ is a basis of $\mathcal{L}$, for any unimodular $\mathbf{U} \in \mathcal{GL}_n(\mathbb{Z}) = \{\mathbf{U} \in \mathbb{Z}^{n\times n} \mid \det(\mathbf{U}) = \pm 1\}$.

Given a basis $\mathbf{B}$ of a lattice $\mathcal{L}$, many algorithms on lattices attempt to find 'better' bases in the set $\{\mathbf{B} \cdot \mathbf{U} \mid \mathbf{U} \in \mathcal{GL}_n(\mathbb{Z})\}$ of all bases of the lattice $\mathcal{L}$. These algorithms are called 'basis reduction algorithms' and will be treated in Section 2.6 and later sections.

2.2.4 Fundamental domains

Any basis $\mathbf{B}$ defines a fundamental domain

$$\mathcal{P}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in [-1/2, 1/2) \right\},$$

also called the 'centered parallelepiped' of the basis $\mathbf{B}$. An example of such a fundamental domain for two bases of the same lattice can be seen in Figure 2.1.

This centered parallelepiped $\mathcal{P}(\mathbf{B})$ is a set of representatives of the quotient group $\operatorname{span}_{\mathbb{R}}(\mathcal{L})/\mathcal{L}$. Concretely, this means that one can uniquely decompose any $\mathbf{x} \in \operatorname{span}_{\mathbb{R}}(\mathcal{L})$ into $\mathbf{x} = \mathbf{v} + \mathbf{f}$, where $\mathbf{v} \in \mathcal{L}$ and $\mathbf{f} \in \mathcal{P}(\mathbf{B})$.

As will be made precise later, the shape of this fundamental domain $\mathcal{P}(\mathbf{B})$ gives an intuitive measure on the 'quality' of the basis $\mathbf{B}$. A basis $\mathbf{B}$ for which $\mathcal{P}(\mathbf{B})$ is reasonably concentrated around the origin and whose shape is not skewed is considered 'good', like the green basis in Figure 2.1.

Covolume of a Lattice

The volume $\operatorname{Vol}(\mathcal{P}(\mathbf{B})) = \operatorname{Vol}(\operatorname{span}(\mathcal{L})/\mathcal{L})$ of the fundamental domain $\mathcal{P}(\mathbf{B})$ does not depend on the basis $\mathbf{B}$ of $\mathcal{L}$, and is called the covolume $\operatorname{Vol}\mathcal{L}$ of $\mathcal{L}$. This is a rough measure of the 'sparsity' of the lattice $\mathcal{L}$.

Definition 3 (Covolume)**.** *The covolume* $\operatorname{Vol}\mathcal{L}$ *of a lattice $\mathcal{L}$ is defined as*

$$\operatorname{Vol}\mathcal{L} := \operatorname{Vol}(\operatorname{span}(\mathcal{L})/\mathcal{L}) = |\det(\langle \mathbf{b}_i, \mathbf{b}_j \rangle_{ij})|^{1/2},$$

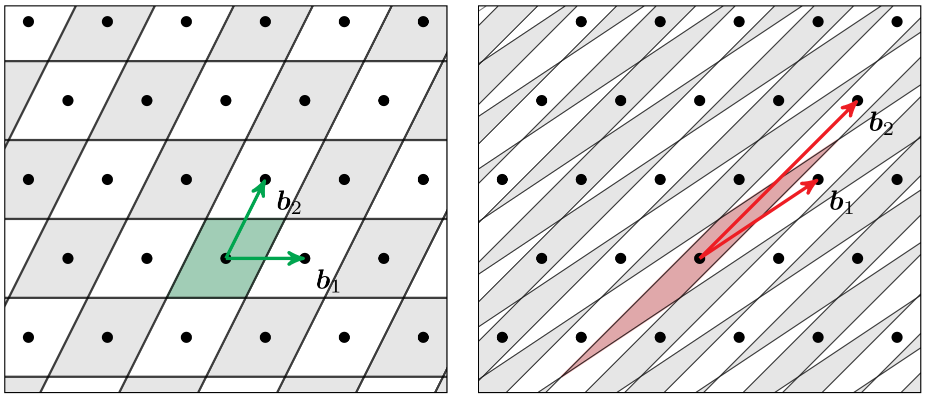


Figure 2.1: In this picture two bases (green, red) of the same lattice is given, in combination with their respective fundamental domains $\mathcal{P}(\mathbf{B})$ (the red and green parallelepiped).

which is independent of the basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ *of* $\mathcal{L}$*. Here, we mean by* $\langle \mathbf{b}_i, \mathbf{b}_j \rangle_{ij}$ *the matrix whose* ij*-th coordinate is given by* $\langle \mathbf{b}_i, \mathbf{b}_j \rangle$*.*

The First Minimum

The covolume is a measure of sparseness of a lattice, and therefore arises in many inequalities regarding geometric quantities of lattices. We will later see such an inequality (Minkowski's inequality, Equation (2.2)) for one of the most important geometric invariants of a lattice $\mathcal{L}$, the first minimum $\lambda_1(\mathcal{L})$. This is also known as the minimum distance or as the first successive minimum of $\mathcal{L}$.

Definition 4 (First minimum)**.** *The first minimum* $\lambda_1(\mathcal{L}) \in \mathbb{R}_{>0}$ *of the lattice* $\mathcal{L}$ *is defined as follows*

$$\lambda_1(\mathcal{L}) = \min\{\|\mathbf{y}\| \mid \mathbf{y} \in \mathcal{L} \backslash \{0\}\}. \tag{2.2}$$

This first minimum $\lambda_1(\mathcal{L})$ equals the length of a shortest non-zero vector of $\mathcal{L}$, and is well-defined by the discreteness of the lattice $\mathcal{L}$. Equivalently, it is the minimum distance between any two distinct lattice points. Minkowski proved in 1889 that the minimum distance and the covolume of a lattice are related in the following way.

Theorem 1 (Minkowski's theorem)**.** *For every lattice* $\mathcal{L}$ *of rank* n *we have the following bound on the first minimum.*

$$\lambda_1(\mathcal{L}) \leq 2 \cdot \frac{\mathrm{Vol}(\mathcal{L})^{1/n}}{\mathrm{Vol}(\mathcal{B}_n)^{1/n}} \leq \sqrt{n} \cdot \mathrm{Vol}(\mathcal{L})^{1/n},$$

where $\mathrm{Vol}(\mathcal{B}_n) = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)}$ *is the volume of an* n*-dimensional unit* $\mathbb{R}$*-ball, and* Γ *denotes the standard Gamma function.*

Intuitively, one can think about Minkowski's inequality in the following way. A lattice $\mathcal{L}$ whose shortest (non-zero) vector is long (i.e., $\lambda_1(\mathcal{L})$ is large), cannot have a small covolume ($\operatorname{Vol}(\mathcal{L})$). In other words, the two numbers $\lambda_1(\mathcal{L})$ and $\operatorname{Vol}(\mathcal{L})$ both give a measure for the sparsity of a lattice. The notion $\operatorname{Vol}(\mathcal{L})$ is more rough and more 'average', so that a large covolume can still mean that a lattice has a small first minimum $\lambda_1(\mathcal{L})$.

Occasionally, $\operatorname{Vol}(\mathcal{L})$ is considered as a measure for 'global sparsity' whereas $\lambda_1(\mathcal{L})$ is considered as a measure for 'local sparsity' of the lattice $\mathcal{L}$. The reason for these notions is the difference in scale where $\operatorname{Vol}(\mathcal{L})$ and $\lambda_1(\mathcal{L})$ have impact: For a very large box B inside V, the number of lattice points in $B \cap \mathcal{L} \approx \operatorname{Vol}(B)/\operatorname{Vol}(\mathcal{L})$ depends on the covolume of the lattice. For a small box B inside V the number of points $B \cap \mathcal{L}$ depends heavily on the smallest vectors in V and hence on $\lambda_1(\mathcal{L})$.

2.2.5 Computational problems

In lattice-based cryptography, (among many others) the following computational problems are of fundamental importance. Lattice-based cryptographic protocols rely on the hardness of these problems.

These computational problems occur so often that they are often denoted by their acronym: the Shortest Vector Problem (SVP), the Closest Vector Problem (CVP) and Bounded Distance Decoding (BDD) (see Figure 2.2).

The Shortest Vector Problem

Problem 1 (Shortest Vector Problem (SVP_γ)). *Given as input a basis $\boldsymbol{B}$ of a lattice $\mathcal{L}$ and a $\gamma \in \mathbb{R}_{\geq 1}$, the* γ-shortest vector problem *is the computational task of finding a non-zero lattice vector $\mathbf{x} \in \mathcal{L}$ that satisfies $\|\mathbf{x}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$.*

The task of finding a short vector in a lattice (given by a 'bad' basis $\mathbf{B}$), as we will elaborate on later, seems to be very hard to solve, even for quantum computers. For this problem to be hard, it is of essential importance that the dimension (or rank) $\operatorname{rk}(\mathcal{L})$ of the lattice is large; for cryptographic purposes, lattices of rank well over 300 are the rule. For lattices of smaller dimensions (below 80) the shortest vector problem is considered easy.

Remark 1. *Note that for $\gamma = 1$, one gets the 'original' shortest vector problem; the output must then be a shortest vector of the lattice $\mathcal{L}$. This γ in above definition, generally called the* approximation factor*, is often written as a function in the dimension n of the lattice. In the context of cryptography, it is mostly sufficient to find a 'really short' vector, but not a* shortest *vector per se. For example, for cryptographic purposes, $\gamma = \sqrt{n}$ is considered very short, and $\gamma = poly(n)$ is considered very interesting. We will see that the shortest vector problem for $\gamma = 2^n$ can be solved efficiently by means of the LLL-algorithm, and is therefore considered 'easy'.*

In the literature, actually another variant of the shortest vector problem is used, called the *Hermite variant.* In this variant, the quantity $\lambda_1(\mathcal{L})$ is replaced by the root determinant $\det(\mathcal{L})^{1/n}$, where n is the dimension of the lattice. In this text, we will use SVP and 'the shortest vector problem' for both versions, where it depends on the context which one we mean.

Problem 2 (Hermite Shortest Vector Problem (SVP_γ))**.** *Given as input a basis* $\boldsymbol{B}$ *of an n-dimensional lattice* $\mathcal{L}$ *and a* $\gamma \in \mathbb{R}_{>0}$*, the* γ-Hermite shortest vector problem *is the computational task of finding a non-zero lattice vector* $\mathbf{x} \in \mathcal{L}$ *that satisfies* $\|\mathbf{x}\| \leq \gamma \cdot \det(\mathcal{L})^{1/n}$*.*

The Closest Vector Problem

The closest vector problem is of a slightly different flavor than SVP; instead of just a basis $\mathbf{B}$ as input, also an additional vector $\mathbf{t} \in \operatorname{span}(\mathcal{L})$ is given, often called the *target* (which is generally not a lattice vector). The task is to 'round' the target $\mathbf{t}$ to the *nearest* lattice vector $\mathbf{v} \in \mathcal{L}$, see the right panel of Figure 2.2. This problem can be formally phrased as follows.

Problem 3 (Closest Vector Problem (CVP_γ))**.** *Given as input a basis* $\boldsymbol{B}$ *of a lattice* $\mathcal{L}$*, a target vector* $\mathbf{t} \in \operatorname{span}(\mathcal{L})$ *and* $\gamma \in \mathbb{R}_{>0}$*, the* closest vector problem *is the computational task of finding a lattice vector* $\mathbf{x} \in \mathcal{L}$ *that satisfies* $\|\mathbf{x}-\mathbf{t}\| \leq \gamma$*.*

Remark 2. *By putting* $\gamma = \min_{\mathbf{x}\in\mathcal{L}} \|\mathbf{x} - \mathbf{t}\|$*, one obtains the 'original' closest vector problem, as the computational task is then to find* $\mathbf{x} \in \mathcal{L}$ *that minimizes the distance* $\|\mathbf{x} - \mathbf{t}\|$*.*

Note that if $\gamma \in \mathbb{R}_{>0}$ *is too small, there might not even exist a* $\mathbf{x} \in \mathcal{L}$ *such that* $\|\mathbf{x}-\mathbf{t}\| \leq \gamma$*. In real-life settings, often is assumed that* $\gamma \geq \min_{\mathbf{x}\in\mathcal{L}} \|\mathbf{t}-\mathbf{x}\|$*, which implies that the problem always has a solution. The ratio* $\gamma / \min_{\mathbf{x}\in\mathcal{L}} \|\mathbf{t} - \mathbf{x}\|$ *is (also) often called the* approximation factor*, and within cryptography this is usually polynomial in the dimension of the lattice.*

As with SVP, this problem, with polynomial approximation factors, also seems to be hard to solve for (quantum) computers, and likewise, lattices of rank well over 300 are the rule.

Bounded Distance Decoding

The following computational problem, Bounded Distance Decoding, is a variant on the closest vector problem, with the difference that it only considers target vectors $\mathbf{t} \in \operatorname{span}(\mathcal{L})$ *sufficiently close* to the lattice $\mathcal{L}$.

Problem 4 (Bounded distance decoding (BDD))**.** *Given as input a basis* $\boldsymbol{B}$ *of a lattice* $\mathcal{L}$*, an error* $\epsilon < \lambda_1(\mathcal{L})/2$ *and a target vector* $\mathbf{t} \in \operatorname{span}(\mathcal{L})$ *satisfying* $\operatorname{dist}(\mathcal{L}, \mathbf{t}) \leq \epsilon$*, the* bounded distance decoding problem *is the computational task of finding the (unique) closest lattice vector* $\mathbf{x} \in \mathcal{L}$ *to* $\mathbf{t} \in \operatorname{span}(\mathcal{L})$*, i.e., the lattice vector* $\mathbf{x} \in \mathcal{L}$ *minimizing* $\|\mathbf{x} - \mathbf{t}\|$*.*

Due to the 'promise' that $\mathbf{t}$ is sufficiently close to $\mathcal{L}$, BDD is sometimes seen as a promise-problem version of CVP.

Remark 3. *Note that for increasing* γ*, (Hermite)-SVP*$_\gamma$ *and CVP*$_\gamma$ *becomes easier, whereas for increasing* ϵ*, BDD*$_\epsilon$ *becomes harder.*

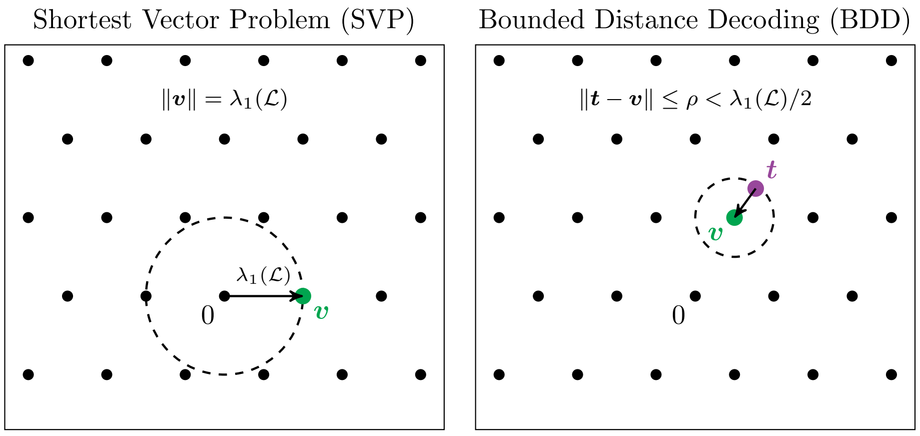

Shortest Vector Problem (SVP)
$\|\boldsymbol{v}\| = \lambda_1(\mathcal{L})$
$\lambda_1(\mathcal{L})$
0
$\boldsymbol{v}$
Bounded Distance Decoding (BDD)
$\|\boldsymbol{t} - \boldsymbol{v}\| \leq \rho < \lambda_1(\mathcal{L})/2$
$\boldsymbol{t}$
$\boldsymbol{v}$
0

Figure 2.2: A two-dimensional visualization of the most important lattice problems for cryptography.

2.3 Projections and orthogonalization

Introduction

Among the most important lattice algorithms are *basis reduction algorithms*, like LLL, BKZ and HKZ, which attempt to improve the quality of a basis $\mathbf{B}$ of a lattice $\mathcal{L}$ by multiplying it by a suitable $\mathbf{U} \in \mathcal{GL}_n(\mathbb{Z})$, yielding a better basis $\mathbf{BU}$. This multiplication by $\mathbf{U}$ is precisely the same as replacing the basis $\mathbf{B}$ by a new basis $\mathbf{B}'$ where each new basis vector is an adequate integer linear combination of the former basis vectors.

A key idea that is almost universal in lattice reduction techniques is that most lattice problems are generally much easier to solve *in lower dimensions*. So, many lattice reduction techniques (LLL, BKZ) improve the quality of the input basis $\mathbf{B}$ by solving lattice problems (SVP, CVP) in *low-dimensional lattices derived from* $\mathbf{B}$.

One way of obtaining low-dimensional lattices in $\mathcal{L} = \mathcal{L}(\mathbf{B})$ is by taking *sublattices*, for example, those generated by only a few columns of $\mathbf{B}$. Another way of obtaining low-dimensional lattice in $\mathcal{L} = \mathcal{L}(\mathbf{B})$ is by *projecting* the lattice onto a lower-dimensional subspace of $\text{span}(\mathcal{L})$.

Gram-Schmidt orthogonalization

Let V be a Euclidean vector space and $W \subseteq V$ a linear subspace. We can define the orthogonal complement $W^\perp$ of W:

$$W^\perp = \{\mathbf{v} \in V \mid \langle \mathbf{v}, \mathbf{w} \rangle = 0 \text{ for all } \mathbf{w} \in W\}$$

The *projection map* $\pi_W : V \to V$ is the unique linear map that is equal to the identity map id_W on W and sends $W^\perp$ to 0. When a basis $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ of W

is given, the space $W^\perp$ is sometimes written as $(\mathbf{w}_1, \dots, \mathbf{w}_k)^\perp$.

This map $\pi_W : V \to V$ then *projects* the space V onto W. It is not generally true that $\pi_W(\mathcal{L}) \subseteq W$ is a lattice if $\mathcal{L} \subseteq V$ is a lattice. For this, W is required to be compatible with the lattice in some sense; this will be answered in Definition 6. To identify what is required from W, we need the notion of Gram-Schmidt orthogonalization of a basis $\mathbf{B}$.

Example 1. *Take the projection map $\pi_1 : \mathbb{R}^2 \to \mathbb{R}$ where $(x, y) \mapsto x$. In terms of the discussion above, this corresponds to taking $V = \mathbb{R}^2$ and $W = \mathbb{R} \times \{0\}$. Note that*

$$\mathcal{L} = \left\{ n_1 \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + n_2 \cdot \begin{bmatrix} \sqrt{2} \\ 0 \end{bmatrix} \;\middle|\; n_1, n_2 \in \mathbb{Z} \right\}$$

is a lattice in $\mathbb{R}^2$, but that $\pi_1(\mathcal{L}) = \{n_1 + n_2\sqrt{2} \mid n_1, n_2 \in \mathbb{Z}\}$ is not a lattice in $\mathbb{R}$ because it is not (uniformly) discrete; we can get the number $n_1 + n_2\sqrt{2}$ to be arbitrarily close to 0.

Definition 5 (Gram-Schmidt orthogonalization)**.** *For a basis $\boldsymbol{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ of the lattice Λ, we define the Gram-Schmidt orthogonalization $\boldsymbol{B}^* = (\mathbf{b}_1^*, \dots, \mathbf{b}_n^*)$ of $\boldsymbol{B}$ as follows: $\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$, where $\pi_i = \pi_{(\mathbf{b}_1,\dots,\mathbf{b}_{i-1})^\perp}$. In other words, the projection map π_i projects to the orthogonal complement of the space generated by $(\mathbf{b}_1, \dots, \mathbf{b}_{i-1})$.*

A different way of defining the Gram-Schmidt orthogonalization is by means of matrix decomposition; $\boldsymbol{B}$ is decomposed as a matrix product $\boldsymbol{B}^* \cdot \mu$, where $\boldsymbol{B}^*$ is orthogonal and μ is upper triangular with ones on the diagonal. One can compute μ and $\boldsymbol{B}^*$ inductively by the following formulae:

$$\mu_{ij} = \frac{\langle \mathbf{b}_i^*, \mathbf{b}_j \rangle}{\langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle};$$

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ji} \mathbf{b}_j^*.$$

Projections

It turns out that the only good way to project lattices $\mathcal{L} = \mathcal{L}(\mathbf{B})$ in such a way that its projected image is *also* a lattice, is by projecting it orthogonally to a vector space spanned by some of the basis vectors[4].

Definition 6 (Projected lattices)**.** *Let $\mathcal{L}$ be a lattice with basis $\boldsymbol{B}$. The projected lattices $\mathcal{L}_i$ with respect to the basis $\boldsymbol{B}$ are defined as follows.*

$$\mathcal{L}_i = \pi_i(\mathcal{L}) \subseteq (\mathbf{b}_1, \dots, \mathbf{b}_{i-1})^\perp,$$

where $\pi_i = \pi_{(\mathbf{b}_1,\dots,\mathbf{b}_{i-1})^\perp}$ projects to the orthogonal complement of the space generated by $(\mathbf{b}_1, \dots, \mathbf{b}_{i-1})$

[4]More specifically, for a given lattice $\mathcal{L} \subseteq V$ and $W \subseteq V$, the projected image $\pi_W(\mathcal{L})$ is a lattice in W if and only if $W = (\mathbf{b}_1, \dots, \mathbf{b}_j)^\perp$ where $\mathbf{b}_k$ are vectors in some basis $\mathbf{B}$ of $\mathcal{L}$.

One can see that $\mathcal{L}_i$ is indeed a lattice because it has $(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_n))$ as a basis. This gives rise to the definition of a *projected basis*.

Definition 7 (Projected bases)**.** *Let* $\mathbf{B}$ *be a basis of a lattice* $\mathcal{L}$*. We denote by* $\mathbf{B}_{[i:j]}$ *the projected basis*

$$\mathbf{B}_{[i:j]} := (\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j)),$$

where $\pi_i = \pi_{(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^\perp}$*.*

It is clear that $\mathbf{B}_{[i:n]}$ is a basis of the lattice $\pi_i(\mathcal{L})$. These bases will play an important role in BKZ and HKZ, two important reduction algorithms.

The Profile of a Basis

Another measure for the quality of the basis $\mathbf{B}$ of a lattice is called the *profile* of the basis. This is a vector consisting of the lengths of the Gram-Schmidt vectors $\mathbf{b}_i^*$ for $i \in [n]$.

Definition 8. *The profile of a basis* $\mathbf{B}$ *is defined as n-tuple consisting of the Gram-Schmidt norms*

$$(\|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_n^*\|).$$

A basis is generally considered to be good if $\|\mathbf{b}_j^*\|$ *does not decrease too quickly* for increasing j. By plotting the logarithms of the Gram-Schmidt norms $(\log\|\mathbf{b}_j^*\|)_{j \in [d]}$ one often gets a quick insight on this decrease (and hence of the quality of the basis). For an example of different profiles see Figs. 2.7 and 2.8. Generally, a very negative slope (say -0.02 per step) indicates a bad basis, whereas an only slightly negative slope (say $-.003$ per step) indicates a better basis.

The profile of the projected basis $\mathbf{B}_{[i:j]}$ is given by $(\|\mathbf{b}_i^*\|, \ldots, \|\mathbf{b}_j^*\|)$.

2.4 Babai's fundamental domain

Given a basis $\boldsymbol{B}$ of the lattice $\mathcal{L}$, and let $\boldsymbol{B}^*$ be its Gram-Schmidt orthogonalization. Then we define Babai's fundamental domain as the parallelepiped $\mathcal{P}(\boldsymbol{B}^*) = \{\sum_{i=1}^n x_i \mathbf{b}_i^* \mid x_i \in [-1/2, 1/2)\}$.

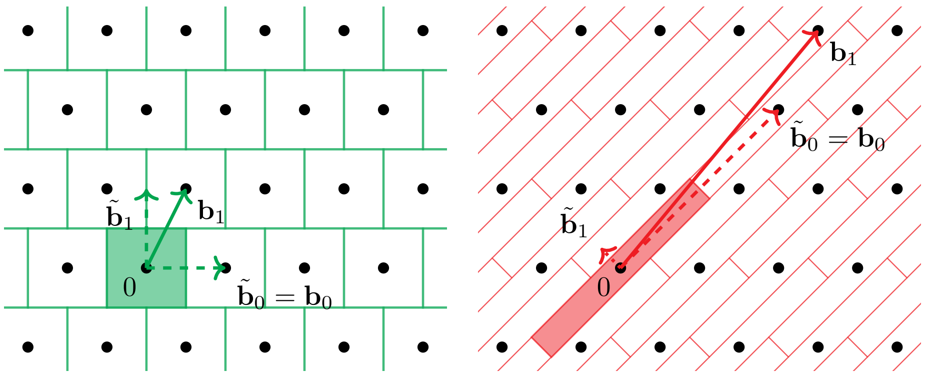
This fundamental domain occurs in *Babai's nearest-plane algorithm*, that uniquely decomposes any target $\mathbf{t} \in \operatorname{span}(\mathcal{L})$ into $\mathbf{t} = \mathbf{v} + \mathbf{e}$, where $\mathbf{v} \in \mathcal{L}$ and $\mathbf{e} \in \mathcal{P}(\boldsymbol{B}^*)$ lies in the fundamental domain (see Figure 2.3).

The fact that $\mathcal{P}(\boldsymbol{B}^*)$ is a fundamental domain of $\mathcal{L}$, gives us the following important identity.

$$\operatorname{Vol}(\mathcal{L}) = \operatorname{Vol}(\mathcal{P}(\boldsymbol{B}^*)) = \prod_{j=1}^n \|\mathbf{b}_j^*\|. \tag{2.3}$$

Algorithm 1: The Babai nearest-plane algorithm

```
Input :
    • A basis $\boldsymbol{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice $\mathcal{L}$ with Gram-Schmidt
      orthogonalization $\boldsymbol{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$,
    • a target $\mathbf{t} \in \text{span}(\mathcal{L})$.
Output: $(\mathbf{v}, \mathbf{e})$ such that $\mathbf{v} + \mathbf{e} = \mathbf{t}$, with $\mathbf{v} \in \mathcal{L}$ and $\mathbf{e} \in \mathcal{P}(\boldsymbol{B}^*)$.
$\mathbf{e} := \mathbf{t}$
$\mathbf{v} := \mathbf{0}$
for $i = n$ down to 1 do
    $k := \lceil \frac{\langle \mathbf{e}, \mathbf{b}_i^* \rangle}{\|\mathbf{b}_i^*\|^2} \rfloor$
    $\mathbf{e} := \mathbf{e} - k\mathbf{b}_i$
    $\mathbf{v} := \mathbf{v} + k\mathbf{b}_i$
end
return $(\mathbf{v}, \mathbf{e})$
```



$\tilde{\mathbf{b}}_1$
$\mathbf{b}_1$
0
$\tilde{\mathbf{b}}_0 = \mathbf{b}_0$
$\mathbf{b}_1$
$\tilde{\mathbf{b}}_0 = \mathbf{b}_0$
$\tilde{\mathbf{b}}_1$
0

Figure 2.3: Babai's fundamental domain for a good (left) and a bad basis (right) of the same lattice.

Basis Quality

Given a target $\mathbf{t} \in \operatorname{span}(\mathcal{L})$ the close vector $\mathbf{v}$ that Babai's nearest plane algorithm returns depends on the basis. Preferably, one would like the error $\mathbf{e} = \mathbf{t} - \mathbf{v} \in \mathcal{P}(\boldsymbol{B}^*)$ to be as short as possible, so what properties should a basis have to minimize this? For any $\mathbf{e} \in \mathcal{P}(\boldsymbol{B}^*)$ we have the worst-case bound $\|\mathbf{e}\|^2 \leq \frac{1}{4}\sum_{i=1}^{n}\|\mathbf{b}_i^*\|^2$. Furthermore, if the target $\mathbf{t}$ is uniform over the cosets $\operatorname{span}(\mathcal{L})/\mathcal{L}$, then in particular $\mathbf{e}$ is uniform over $\mathcal{P}(\boldsymbol{B}^*)$. Its expected squared length $\mathbb{E}[\|\mathbf{e}\|^2]$ is then given by $\frac{1}{12}\sum_{i=1}^{n}\|\mathbf{b}_i^*\|^2$. Given the identity $\prod_{i=1}^{n}\|\mathbf{b}_i^*\| = \det(\mathcal{L})$ the worst-case and expected squared length is thus minimized when $\|\mathbf{b}_1^*\| = \ldots = \|\mathbf{b}_n^*\|$, i.e., Babai's nearest plane algorithm finds a closer vector on average if the basis has a well balanced Gram-Schmidt profile. Note that a perfect balanced basis with equal Gram-Schmidt norms might not always exist, but we can still use a basis that have a somewhat balanced profile, whose Gram-Schmidt norms do not differ too much. Informally we call such a basis 'good', and a basis with a very unbalanced profile 'bad'.

Secondly, for a bounded distance decoding problem Babai's nearest plane algorithm provably returns the unique closest vector if the target $\mathbf{t}$ lies at distance at most $\frac{1}{2}\min_i\|\mathbf{b}_i^*\|$ from the lattice. Again this quantity is maximized when $\|\mathbf{b}_1^*\| = \ldots = \|\mathbf{b}_n^*\|$, or, more generally, we expect it to be higher when the basis profile is more balanced. Summarizing, both problems CVP and BDD are easier to solve whenever the basis is balanced.

Size reduction

Babai's nearest-plane algorithm can in principle also be applied to targets $\mathbf{t} \notin \operatorname{span}(\mathcal{L})$ that do *not* lie in the span of the lattice $\mathcal{L}$. In that case, $\mathbf{t} = \mathbf{v} + \mathbf{e}$ is uniquely decomposed into $\mathbf{v} \in \mathcal{L}$ and $\mathbf{e} \in \mathcal{P}(\boldsymbol{B}^*) + \operatorname{span}(\mathcal{L})^\perp$. One can interpret Babai's nearest-plane algorithm as an attempt to reduce the target $\mathbf{t}$ optimally using the basis $\boldsymbol{B}$. If the target $\mathbf{t}$ were already reduced, i.e., if $\pi_{(\mathbf{b}_1,\ldots,\mathbf{b}_n)}(\mathbf{t}) \in \mathcal{P}(\boldsymbol{B}^*)$, we call $\mathbf{t}$ *size reduced.* This notion of size-reducedness can then be extended to an entire basis.

Definition 9 (Size reduced basis)**.** *A basis $\boldsymbol{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is said to be* size reduced *if for all $j \in \{2, \ldots, n\}$,*

$$\pi_{(\mathbf{b}_1^*,\ldots,\mathbf{b}_{j-1}^*)}(\mathbf{b}_j) \in \mathcal{P}(\mathbf{b}_1^*, \ldots, \mathbf{b}_{j-1}^*).$$

That is, if each basis element is size-reduced with respect to the previous basis vectors.

By applying Babai's nearest plane algorithm progressively on a given basis $\boldsymbol{B}$ of $\mathcal{L}$, one can always obtain a size-reduced basis of $\mathcal{L}$ efficiently.

This notion of size-reducedness is important during computations on bases, as it avoids a phenomenon called *coefficient explosion.* During repeated basis operations on rational bases (i.e., with entries in $\mathbb{Q}$) the numerators and denominators of the rational entries might increase drastically, if no care is taken.

These numbers can grow so large that they prevent an algorithm to be efficiently computable. Applying size-reduction sufficiently often in between these basis operations keeps the rational numbers 'small', and avoids this potential explosive growth of coefficients.

2.5 Lagrange Reduction Algorithm

In this section, we will explain an algorithm that finds a basis of a two-dimensional lattice consisting of vectors that attain the respective successive minima. Thus, the Lagrange reduction algorithm finds a shortest basis of a two-dimensional lattice.

Theorem 2 ('Wristwatch lemma')**.** *Let $\mathcal{L}$ be a two-dimensional lattice. Then there exists a basis $\boldsymbol{B} = (\mathbf{b}_1, \mathbf{b}_2)$ of $\mathcal{L}$ such that*

- $\mathbf{b}_1$ *is a shortest vector of $\mathcal{L}$, i.e., $\|\mathbf{b}_1\| = \lambda_1(\mathcal{L})$.*
- $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \frac{1}{2}\|\mathbf{b}_1\|^2$.

Furthermore, one can efficiently compute such a basis using Algorithm 2.

Proof. The proof of this theorem is 'by algorithm', as we will show that Algorithm 2 always computes a basis $\boldsymbol{B}$ satisfying the conditions in Theorem 2.

The output vectors $(\mathbf{b}_1, \mathbf{b}_2)$ of Algorithm 2 must form a basis of the lattice $\mathcal{L}$, as every occurring basis operation (swap, row-addition) is a unimodular transformation of the basis. The algorithm terminates because the norm $\mathbf{b}_1$ strictly decreases by a constant every repeat-loop, and has $\lambda_1(\Lambda)$ as a minimum.

It remains to prove that the output satisfies the conditions of Theorem 2. Write $\mathbf{b}_1, \mathbf{b}_2$ for the output basis, and write $\mathbf{b}_2^{\text{old}} = \mathbf{b}_2 + k\mathbf{b}_1$ for the previous version of $\mathbf{b}_2$ in the algorithm. Then

$$\begin{aligned}\langle \mathbf{b}_1, \mathbf{b}_2 \rangle = \langle \mathbf{b}_1, \mathbf{b}_2^{\text{old}} - k\mathbf{b}_1 \rangle &= \langle \mathbf{b}_1, \mathbf{b}_2^{\text{old}} \rangle - k\|\mathbf{b}_1\|^2 \\ &= \left(\frac{\langle \mathbf{b}_1, \mathbf{b}_2^{\text{old}} \rangle}{\|\mathbf{b}_1\|^2} - k \right) \|\mathbf{b}_1\|\end{aligned}$$

Since $k \in \mathbb{Z}$ is the rounded version of $\frac{\langle \mathbf{b}_1, \mathbf{b}_2^{\text{old}} \rangle}{\|\mathbf{b}_1\|^2}$, it follows that $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \frac{1}{2}\|\mathbf{b}_1\|^2$.

To show that $\|\mathbf{b}_1\| = \lambda_1(\mathcal{L})$, we pick any vector $\mathbf{v} \in \mathcal{L}$, and write it as $\mathbf{v} = \ell\mathbf{b}_1 + k\mathbf{b}_2$ with $k, \ell \in \mathbb{Z}$. Then, using $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \frac{1}{2}\|\mathbf{b}_1\|^2$,

$$\begin{aligned}\|\mathbf{v}\|^2 = \langle \ell\mathbf{b}_1 + k\mathbf{b}_2, \ell\mathbf{b}_1 + k\mathbf{b}_2 \rangle &= \ell^2\|\mathbf{b}_1\|^2 + 2k\ell\langle \mathbf{b}_1, \mathbf{b}_2 \rangle + k^2\|\mathbf{b}_2\|^2 \\ &\geq (\ell^2 + k^2 - k\ell)\|\mathbf{b}_1\|^2 \geq \|\mathbf{b}_1\|^2,\end{aligned}$$

as $\ell^2 + k^2 - k\ell \geq 1$ for all $k, \ell \in \mathbb{Z}$. □

Algorithm 2: Lagrange's reduction algorithm

```
Input  : A basis $(\mathbf{b}_1, \mathbf{b}_2)$ of a lattice $\mathcal{L}$.
Output: A basis $(\mathbf{b}_1, \mathbf{b}_2)$ such that $\|\mathbf{b}_1\| = \lambda_1(\mathcal{L})$ and
        $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \frac{1}{2}\|\mathbf{b}_1\|^2$, as in Theorem 2.

repeat
    swap $\mathbf{b}_1 \leftrightarrow \mathbf{b}_2$
    $k \leftarrow \lceil \frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle}{\|\mathbf{b}_1\|^2} \rfloor$
    $\mathbf{b}_2 \leftarrow \mathbf{b}_2 - k\mathbf{b}_1$
until $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$
```

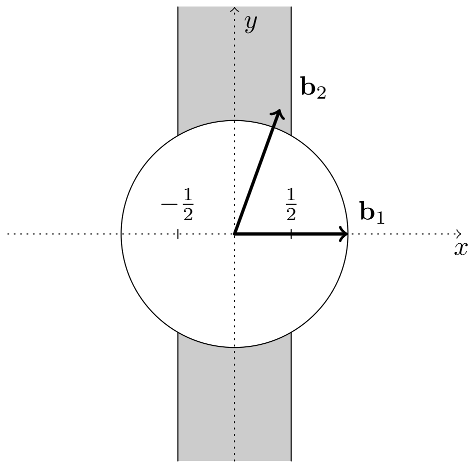


Figure 2.4: The 'wristwatch lemma' visualized. Given a basis vector $\mathbf{b}_1$, it is always possible to reduce another basis vector $\mathbf{b}_2$ with it, so that the reduction $\mathbf{b}_2 - k\mathbf{b}_1$ lands into the gray area (the wristwatch band).

2.6 The LLL algorithm

The LLL-algorithm, invented by Lenstra, Lenstra and Lovász [LLL82], is a basis reduction algorithm achieving an exponential approximation factor in the degree, while running in polynomial time. It is one of the most fundamental basis reduction algorithms.

The LLL-algorithm consists mainly of two ingredients. The first ingredient consists of *locally Lagrange-reducing the basis at every point.* In other words, we want every projected basis pair $\boldsymbol{B}_{[i:i+1]} = (\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}))$ to be Lagrange reduced. This has as a consequence that the Gram Schmidt norm sequence $(\mathbf{b}_i^*)_i$ does not decrease too quickly. The second ingredient consists of keeping the basis size-reduced in order to avoid coefficient explosion, which is needed for efficiency.

Algorithm 3: LLL-reduction algorithm

```
Input  : A basis $\boldsymbol{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ of a lattice $\Lambda$, $\delta \in (\frac{1}{4}, 1]$.
Output: An LLL-reduced basis $\boldsymbol{B}$
Size-reduce the basis $\boldsymbol{B}$
while $\exists i$ such that $\|\mathbf{b}_i^*\| > \delta \|\pi_i(\mathbf{b}_{i+1})\|$ do
    Find a unimodular matrix $\boldsymbol{U} \in \mathbb{Z}^{2\times 2}$ such that $\mathbf{B}_{[i:i+1]}\boldsymbol{U}$ is
     Lagrange reduced
    Put $(\mathbf{b}_i', \mathbf{b}_{i+1}') = (\mathbf{b}_i, \mathbf{b}_{i+1})\boldsymbol{U}$
    Put $\boldsymbol{B} = (\mathbf{b}_1, \dots, \mathbf{b}_{i-1}, \mathbf{b}_i', \mathbf{b}_{i+1}', \mathbf{b}_{i+2}, \dots, \mathbf{b}_n)$
    Size-reduce the new basis $\boldsymbol{B}$.
    Put $\pi_i = \pi_{(\mathbf{b}_1, \dots, \mathbf{b}_{i-1})^\perp}$.
end
Return $\boldsymbol{B}$
```

Theorem 3. *The LLL-reduction algorithm as in Algorithm 3 applied on an integral basis* $(\mathbf{b}_1, \dots, \mathbf{b}_m) = \boldsymbol{B} \in \mathbb{Z}^{m\times n}$ *with* $\delta \in (\frac{1}{4}, 1]$, *terminates within* $O(n^2 \log(\max_i \|\mathbf{b}_i\|)/\log(\delta))$ *iterations.*

Proof. The standard approach is to define the *potential* $P_{\boldsymbol{B}} = \prod_{j=1}^n \det(\mathcal{L}(\boldsymbol{B}_{[1:j]}))$ of the current basis. In order to prove that the number of iterations in Algorithm 3 is bounded by $O(n^2 \log(\max_i \|\mathbf{b}_i\|))$, we show that the potential is upper bounded by $(\max_i \|\mathbf{b}_i\|)^{\frac{n(n+1)}{2}}$, lower bounded by 1 and decreases with a multiplicative constant each iteration.

As $\det(\mathcal{L}(\boldsymbol{B}_{[1:j]})) = \prod_{i=1}^j \|\mathbf{b}_i^*\| \leq \prod_{i=1}^j \|\mathbf{b}_i\|$, we have

$$P_{\boldsymbol{B}} \leq \prod_{i=1}^n \|\mathbf{b}_i\|^{n-i} \leq (\max_i \|\mathbf{b}_i\|)^{\frac{n(n+1)}{2}},$$

which proves the upper bound. As $\det(\mathcal{L}(\boldsymbol{B}_{[1:j]})) = \mathrm{Vol}(\mathcal{L}(\mathbf{b}_1, \dots, \mathbf{b}_j))$ is the covolume of an integer lattice, it must be lower bounded by 1.

For the decrease of the potential, observe that size reduction does not change the Gram Schmidt norms, so it remains to concentrate on the Lagrange reduction step only. A single Lagrange reduction changes $\boldsymbol{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ into $\boldsymbol{B'} = (\mathbf{b}_1, \dots, \mathbf{b}_{j_0-1}, \mathbf{b}'_{j_0}, \mathbf{b}'_{j_0+1}, \dots, \mathbf{b}_n)$, which only changes the value $\det(\mathcal{L}(\boldsymbol{B}_{[1:j_0]}))$ among those of the $\det(\mathcal{L}(\boldsymbol{B}_{[1:j]}))$ in the definition of the potential $P_{\boldsymbol{B}}$. This implies

$$P_{\boldsymbol{B'}} = \frac{\det(\mathcal{L}(\boldsymbol{B}'_{[1:j_0]}))}{\det(\mathcal{L}(\boldsymbol{B}_{[1:j_0]}))} \cdot P_{\boldsymbol{B}} = \frac{\|\mathbf{b'}^*_{j_0}\|}{\|\mathbf{b}^*_{j_0}\|} \cdot P_{\boldsymbol{B}} < \sqrt{\delta} \cdot P_{\boldsymbol{B}}.$$

The last inequality follows from the fact that $\boldsymbol{B}$ at j_0 was not Lagrange reduced before the loop, i.e.,

$$\delta \|\mathbf{b}^*_{j_0}\|^2 = \delta \|\pi_{j_0}(\mathbf{b}_{j_0})\| > \|\pi_{j_0}(\mathbf{b}_{j_0+1})\|^2,$$

together with the fact that $\boldsymbol{B'}$ at j_0 is actually Lagrange reduced, which means that $\pi_{j_0}(\mathbf{b}'_{j_0})$ is the shortest vector in $\mathcal{L}(\boldsymbol{B})_{j_0:j_0+1}$, i.e.,

$$\|\pi_{j_0}(\mathbf{b}_{j_0+1})\|^2 \geq \|\pi_{j_0}(\mathbf{b}'_{j_0})\|^2 = \|\mathbf{b'}^*_{j_0}\|^2.$$

□

The LLL-algorithm always produces a basis that is LLL-reduced, which means that it satisfies the *Lovász condition* and that it is size-reduced.

Definition 10 (LLL reduced basis)**.** *A lattice basis $\boldsymbol{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ is called δ-LLL reduced, for $\delta \in (\frac{1}{4}, 1]$, if*

(i) $\delta\|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2$ for all $1 \leq i \leq n$, and (Lovász Condition)

(ii) it is size-reduced.

The parameter $\delta \in (\frac{1}{4}, 1]$ is known as the reduction parameter. Taking $\delta = 1$ corresponds with applying exact Lagrange reduction in the LLL-algorithm, but for this value we cannot show that the algorithm terminates in polynomial many iterations.

The intuition is that the Gram Schmidt norms of a δ-LLL reduced basis do not decrease too steeply, i.e.,

$$\|\mathbf{b}^*_{i+1}\| \geq \sqrt{\delta - \frac{1}{4}} \cdot \|\mathbf{b}^*_i\|,$$

for all $i \in \{1, \dots, n\}$. The basis being size-reduced implies bounds on the length of the basis vectors.

Corollary 1 (LLL bounds on basis vectors)**.** *Let $\boldsymbol{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ be a δ-LLL reduced basis of the lattice $\mathcal{L}$, with $\delta \in (\frac{1}{4}, 1]$. Writing $\gamma = (\delta - \frac{1}{4})^{-1} \in (\frac{4}{3}, \infty)$ we have*

$$\|\mathbf{b}_1\| \leq \gamma^{(n-1)/2} \cdot \lambda_1(\mathcal{L})$$
$$\|\mathbf{b}_1\| \leq \gamma^{(n-1)/4} \cdot (\mathrm{Vol}(\mathcal{L}))^{1/n}$$

Proof. By combining (i) and (ii) of Definition 10, and writing $\pi_j(\mathbf{b}_{j+1}) = \mathbf{b}^*_{j+1} + cj\mathbf{b}^*_j$, with $c_j \in [-1/2, 1/2)$ (by size-reducedness), we obtain

$$\delta\|\mathbf{b}^*_j\|^2 \leq \|\pi_j(\mathbf{b}_{j+1})\|^2 = \|\mathbf{b}^*_{j+1}\|^2 + c_j^2\|\mathbf{b}^*_j\|^2.$$

Hence, $\|\mathbf{b}^*_j\| \leq (\delta - \frac{1}{4})^{-1/2} \cdot \|\mathbf{b}^*_{j+1}\| = \gamma^{1/2} \cdot \|\mathbf{b}^*_{j+1}\|$. Therefore,

$$\|\mathbf{b}_1\| = \|\mathbf{b}^*_1\| \leq \gamma^{1/2} \cdot \|\mathbf{b}^*_2\| \leq \ldots \leq \gamma^{\frac{n-1}{2}} \cdot \|\mathbf{b}^*_n\|.$$

In particular we have $\|\mathbf{b}_1\| \leq \gamma^{\frac{n-1}{2}} \cdot \min_i\|\mathbf{b}^*_i\|$, and the first statement follows from the fact that $\min_i\|\mathbf{b}^*_i\| \leq \lambda_1(\mathcal{L})$. For the second statement we compute

$$\det(\mathcal{L})^{1/n} = \left(\prod_{i=1}^{n}\|\mathbf{b}^*_i\|\right)^{1/n} \geq \left(\prod_{i=1}^{n}\|\mathbf{b}^*_1\| \cdot \gamma^{\frac{i-1}{2}}\right)^{1/n} = \|\mathbf{b}_1\| \cdot \gamma^{(n-1)/4}.$$

□

2.7 The HKZ algorithm

One of the strongest notion of reducedness of a lattice basis is being Hermite-Korkine-Zolotarev-reduced (HKZ-reduced). This notion follows in a natural way from trying to optimize the the so-called 'basis profile' $(\|\mathbf{b}^*_1\|, \ldots, \|\mathbf{b}^*_n\|)$, by greedily demanding $\|\mathbf{b}^*_1\| = \|\mathbf{b}_1\| = \lambda_1(\mathcal{L}(\boldsymbol{B}))$ and proceeding recursively in the projected lattice $\mathcal{L}(\boldsymbol{B}_{[2:n]})$. That is, demanding $\|\mathbf{b}^*_2\| = \lambda_1(\mathcal{L}(\boldsymbol{B}_{[2:n]}))$, $\|\mathbf{b}^*_3\| = \lambda_1(\mathcal{L}(\boldsymbol{B}_{[3:n]}))$ and so on. The result of such a process (if one includes size-reduction as well) is called a HKZ-reduced basis.

Definition 11 (Hermite-Korkine-Zolotarev reduced basis)**.** *A basis $\boldsymbol{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is called Hermite-Korkine-Zolotarev (HKZ) reduced if it is size-reduced and*

$$\|\mathbf{b}^*_i\| = \lambda_1(\mathcal{L}(\boldsymbol{B}_{[i:n]})) \text{ for all } i = 1, \ldots, n.$$

Any lattice has an HKZ-reduced basis; this follows from the fact that any shortest vector of $\mathcal{L}(\boldsymbol{B}_{[i:n]})$ is primitive[5]in $\mathcal{L}(\boldsymbol{B}_{[i:n]})$.

An alternative, recursive, definition would be as follows: a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is HKZ reduced if $\|\mathbf{b}_1\| = \lambda_1(\mathcal{L})$, and $\boldsymbol{B}_{[2:n]}$ is HKZ reduced (or equals $\{\mathbf{0}\}$). The formal algorithmic analogue of this recursive definition is given in Algorithm 4. This algorithm indeed obtains an HKZ reduced basis, as it computes a shortest vector, projects away from it, and repeats on the projected lattice.

As HKZ reduction is very strong, it could be interpreted as (close to) optimal for many purposes. For example, an HKZ reduction oracle would break almost all lattice-based cryptography. In reality, though, the HKZ reduction algorithm is very expensive, as it requires to solve an exact SVP instance in dimension n, which becomes unfeasible quickly for larger n.

[5]A vector $\mathbf{b} \in \mathcal{L}$ is called primitive if it can be extended to a basis of $\mathcal{L}$. I.e., $\mathbf{b} \in \mathcal{L}$ is called primitive if there exists $\mathbf{b}_2, \ldots, \mathbf{b}_n$ such that $(\mathbf{b}, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ is a basis of $\mathcal{L}$. It can be shown that $\mathbf{b} \in \mathcal{L}$ satisfying $\|\mathbf{b}\| = \lambda_1(\Lambda)$ are always primitive.

Algorithm 4: The HKZ algorithm.

```
Input  : A rank $n$ lattice $\mathcal{L}$ (represented by any basis $\boldsymbol{B}$).
Output: An HKZ-reduced basis $\boldsymbol{B}$
for $i = 1, \ldots, n$ do
    $\pi_i := \pi_{(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^\perp}$
    $\mathbf{w} \leftarrow$ a shortest vector in $\pi_i(\mathcal{L})$
    Lift $\mathbf{w}$ to a lattice vector $\mathbf{b}_i \in \mathcal{L}$ such that $\pi_i(\mathbf{b}_i) = \mathbf{w}$
    Size-reduce $\mathbf{b}_i$ with respect to $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$
end
return $\boldsymbol{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$
```

2.8 The BKZ algorithm

The two reduction algorithms in this text so far, the LLL-reduction and the HKZ-reduction algorithm, can be seen as the extreme cases of basis reduction. The HKZ-reduction algorithm gives an optimal reduced basis with the very high costs of solving exact SVP in lattices, whereas the LLL-reduction algorithm gives only a 'somewhat' reduced basis achieving exponential approximation factors while running in polynomial time.

The BKZ algorithm is a generalization of these two algorithms, and allows for a trade-off: by paying more computation power, one achieves a better reduction quality of a basis (see Figure 2.6).

A common trait of the HKZ and the LLL algorithm is that they both require a short vector in a projected sublattice. The HKZ algorithm needs an SVP oracle of dimension n, whereas the LLL-algorithm only needs such oracle for dimension 2. More specifically, the HKZ-algorithm asks for a shortest vector in 'blocks' $\mathcal{L}(\boldsymbol{B}_{[i:n]})$ of rank at most n, while the LLL does so for 'blocks' $\mathcal{L}(\boldsymbol{B}_{[i:i+1]})$ of rank 2, which can be solved efficiently by Lagrange reduction.

A natural generalization is an algorithm that asks for short vectors in 'blocks' $\mathcal{L}(\boldsymbol{B}_{[i:i+\beta-1]})$ of rank at most β. This is precisely what the Block-Korkine-Zolotarev [Sch87] reduction algorithm does; the parameter β, here, is called the *blocksize*.

Definition 12 (BKZ). *A basis $\boldsymbol{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ is called BKZ-β reduced if it is size-reduced and*

$$\|\mathbf{b}_i^*\| = \lambda_1(\mathcal{L}(\boldsymbol{B}_{[i:\min(i+\beta-1,n)]})) \text{ for all } i = 1, \ldots, n. \tag{2.4}$$

One sees that the BKZ algorithm is indeed truly a generalization of both the HKZ and the LLL algorithm, as the parameter instantiation $\beta = 2$ recovers the LLL algorithm (for $\delta = 1$), whereas putting $\beta = n$ reveals the HKZ algorithm. In the literature Equation (2.4) is often slightly relaxed by a small constant > 1, having a similar role as δ in the LLL algorithm (see Algorithm 3).

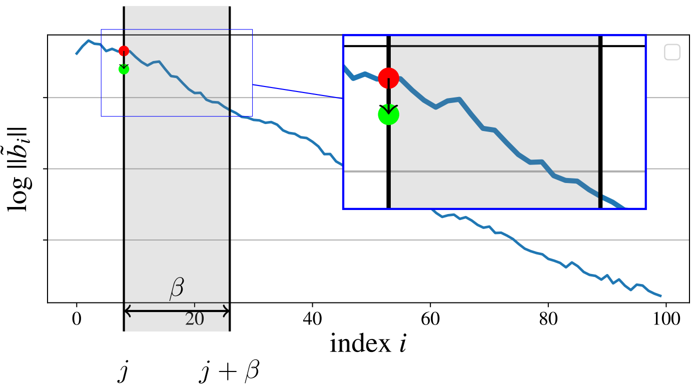


Figure 2.5: The BKZ algorithm repeatedly replaces the basis vector $\mathbf{b}_j$ by a lift of a shortest vector in the projected sublattice $\mathcal{L}(\boldsymbol{B}_{[j:j+\beta-1]})$.

2.8.1 BKZ algorithm

The BKZ algorithm (Algorithm 5) transforms any given basis into a BKZ-reduced basis. The algorithm, similarly to the HKZ algorithm, greedily attempts to satisfy the BKZ condition (Equation (2.4)) at each position i by replacing the basis vector $\mathbf{b}_i$ by a shortest vector in the block $\mathcal{L}(\boldsymbol{B}_{[i:\min(i+\beta-1,n)]})$. Though this has as a consequence that the basis is BKZ-β reduced at position i, it might also invalidate the BKZ-condition at other positions. This is the reason why just one loop of $i \in \{1,\ldots,n-1\}$, which is called a *'tour'*, is usually not enough to fully BKZ-β reduce a basis. The BKZ algorithm instead repeats such tours until the basis remains unchanged, which means that this resulting basis is BKZ-β reduced.

Algorithm 5: The BKZ algorithm.

```
Data: A lattice basis $\boldsymbol{B}$, blocksize $\beta$.
LLL reduce $\boldsymbol{B}$;
while $\boldsymbol{B}$ is not BKZ-$\beta$ reduced do
    for $i = 1,\ldots,n-1$ do                    // A single BKZ-$\beta$ tour
        $\mathbf{w} \leftarrow$ a shortest vector in $\mathcal{L}\left(\boldsymbol{B}_{[i:\min(i+\beta-1,n)]}\right)$;
        Lift $\mathbf{w}$ to a vector $\mathbf{v} \in \mathcal{L}(\boldsymbol{B})$ such that $\pi_i(\mathbf{v}) = \mathbf{w}$;
        Insert $\mathbf{v}$ in $\boldsymbol{B}$ at position $i$ ;
        Apply LLL on $\boldsymbol{B}$ to size-reduce and resolve linear dependencies;
    end
end
```

The BKZ-β algorithm outputs a basis that is 'close to' BKZ-β reduced after $O(n^2\log(n)/\beta^2)$ tours [HPS11; LN20b]. In practice, after a few dozen tours, the basis does not improve much in quality anymore. The cost of BKZ is thus mainly dominated by the exponential (in β) cost of finding a shortest vector in a β-dimensional lattice.

2.8.2 Progressive BKZ

The cost of a BKZ-β tour (i.e., a single loop over $i \in \{1, \dots, n-1\}$ in Algorithm 5) grows quickly with increasing β. It is a fact of experience that an already quite well-reduced basis requires less tours to reach a BKZ-β reduced basis. This suggests that first BKZ-reducing a basis with a smaller block size $\beta' < \beta$ will reduce the run time of expensive β-BKZ reduction afterwards. Apart from limiting the most expensive tours, using this 'mild BKZ reduction' beforehand also lowers the costs of the SVP call required for BKZ, since the basis has a better quality.

The idea of *progressive BKZ* is to apply this idea recursively, by running only a small number of tours (say 1 or 2) first for $\beta' = 2$, then $\beta' = 3$ up to $\beta' = \beta$.

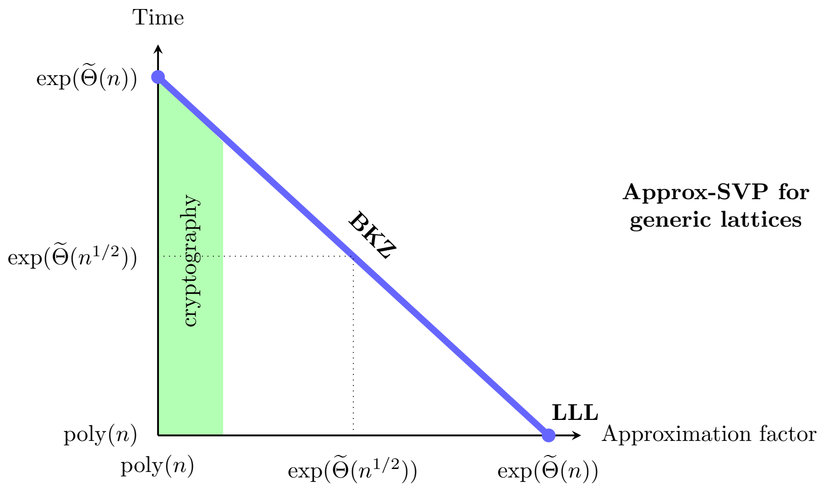

Time
$\exp(\widetilde{\Theta}(n))$
$\exp(\widetilde{\Theta}(n^{1/2}))$
poly(n)
cryptography
BKZ
Approx-SVP for generic lattices
LLL
Approximation factor
poly(n)
$\exp(\widetilde{\Theta}(n^{1/2}))$
$\exp(\widetilde{\Theta}(n))$

Figure 2.6: The BKZ-algorithm allows for a trade-off between basis quality (or equivalently, shorter vectors) and computation time. Paying more time allows for shorter vectors; specifically, one can achieve approximations to the shortest vector within a factor $\exp(\widetilde{\Theta}(n^c))$ within time $\exp(\widetilde{\Theta}(n^{1-c}))$, where n is the lattice dimension.

2.9 The basis quality of the reduction algorithms

In cryptographic context, one usually deals with high-dimensional lattices in which finding short vectors or even applying a good basis reduction algorithm is unfeasible. Of course, these lattices are *constructed* in such a way that finding short vectors, and thus breaking the cryptosystem, is hard.

Due to this general computational unfeasability of these algorithms in high-dimensional lattices it is nearly impossible to conduct real-life timing experiments on lattice reduction algorithms for these lattices. Hence, there is a lack of experimental evidence on the practical behavior of lattice reduction algorithms in high dimensions, that are so relevant for cryptography. Instead, one turns to more heuristic models to predict the behavior of the reduction algorithms LLL, BKZ and HKZ. These models are mostly founded on the *Gaussian heuristic* and extensive experiments in lower, feasible dimensions.

2.9.1 Gaussian Heuristic

The most commonly used heuristic for lattice algorithms is the so-called Gaussian Heuristic, which is heavily verified by experiments.

Heuristic 1 (Gaussian Heuristic)**.** *For a rank n lattice $\mathcal{L}$ and a measurable set $S \subset \operatorname{span} \mathcal{L}$, the Gaussian Heuristic states that*

$$|(\mathcal{L} \setminus \{0\}) \cap S| \approx \frac{\operatorname{Vol}_n S}{\operatorname{Vol} \mathcal{L}}.$$

Furthermore, the non-zero lattice vectors in $\mathcal{L} \cap S$ are uniformly distributed over S.

The Gaussian heuristic forgets about all structure and properties of the lattice, except the dimension and the covolume (and the 0-vector). As a consequence, the lattice is then viewed as a uniformly distributed set of points in space with density $1/\operatorname{Vol}(\mathcal{L})$. So, we would expect about $\operatorname{Vol}(S)/\operatorname{Vol}(\mathcal{L})$ non-zero lattice points in a reasonable volume S. In applications, the set S takes the shape of a (linear transformed) hypercube or ball. Applying the Gaussian heuristic to a ball, we obtain an often-used estimate for the first successive minimum $\lambda_1(\mathcal{L})$ of a lattice.

Heuristic Claim 1. *Let $\mathcal{L}$ be a rank n lattice with volume $\operatorname{Vol}(\mathcal{L})$. The expectation of the first minimum $\lambda_1(\mathcal{L})$ under the Gaussian Heuristic is given by*

$$\operatorname{gh}(\mathcal{L}) := \frac{\operatorname{Vol}(\mathcal{L})^{1/n}}{\operatorname{Vol}(\mathcal{B}_n)^{1/n}} \approx \sqrt{n/(2\pi e)} \cdot \operatorname{Vol}(\mathcal{L})^{1/n},$$

where $\mathcal{B}_n$ is the n-dimensional unit ball. We also denote $\operatorname{gh}(n) = \operatorname{Vol}(\mathcal{B}_n)^{-1/n} \approx \sqrt{n/(2\pi e)}$ for the expected first minimum of a rank n lattice with volume 1.

2.9.2 HKZ shape

The typical log-profile $(\log\|\mathbf{b}_j^*\|)_j$ of an HKZ reduced basis of a random lattice is somewhat concave, see Figure 2.7. This shape is correctly predicted by applying the Gaussian heuristic to the HKZ definition.

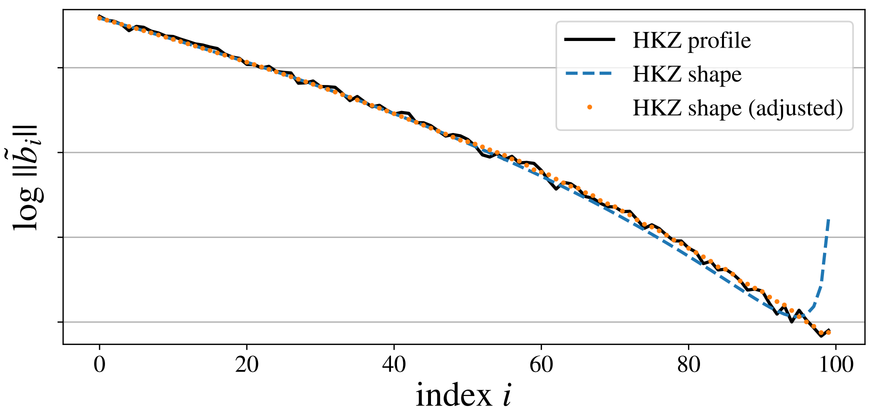


Figure 2.7: A typical HKZ reduced basis of a rank 100 lattice compared to the heuristic HKZ shape of Definition 13, with and without adjustments for the tail part.

Definition 13. *We define the HKZ shape* $(\ell_1, \ldots, \ell_n)$ *of rank* n *recursively as follows*

$$\ell_i = \mathrm{gh}(n-i+1) \cdot \left(\prod_{j=1}^{i-1} \ell_j \right)^{1/(n-i+1)}.$$

The above profile shape should be thought of as a Gram-Schmidt basis profile of a HKZ reduced basis of a random lattice with volume 1. For a lattice of covolume $\det(\mathcal{L})$, the HKZ reduced basis of rank n has basis profile $\log\|\mathbf{b}_i^*\| \approx \log \ell_i + \frac{1}{n}\log(\det \mathcal{L})$, according to the Gaussian heuristic. As can be seen in Figure 2.7, this estimate is quite accurate, especially for $i \ll n$ (say, $i < n - 50$).

Observe that the rightmost part of the HKZ shape does not correspond with the experiment, which can be explained by the fact that the Gaussian heuristic generally only applies well in higher dimensions. This issue is solved by experimentally computing an actual HKZ profile $(\|\mathbf{b}_i^*\|)_i$ for an average lattice of rank 50 and replace the inaccurate rightmost part of the Gaussian heuristic predicted HKZ shape by this experimentally computed profile. This adjusted estimate, predicts an actual HKZ profile quite well, as can be seen in Figure 2.7.

2.9.3 Geometric Series Assumption

A typical log-profile $(\log\|\mathbf{b}_i^*\|)_i$ of an LLL or BKZ-β reduced basis for a block size $\beta \ll n$ resembles a straight line. This can be modeled by the statement $\|\mathbf{b}_{i+1}^*\|/\|\mathbf{b}_i^*\| = \alpha_\beta$, where $\alpha_\beta > 1$ is some constant that only depends on β (where $\beta = 2$ for LLL). This model is also called the Geometric Series Assumption (GSA). Using the Gaussian heuristic, one can predict this constant for various instantiations of BKZ (and LLL).

Heuristic 2 (Profile under the Geometric Series Assumption (GSA))**.** *Let $\boldsymbol{B}$ be a BKZ-β reduced basis, then under the Geometric Series Assumption and the Gaussian heuristic the profile satisfies*

$$\log(\|\mathbf{b}_i^*\|) = \frac{n+1-2i}{2} \cdot \log(\alpha_\beta) + \frac{\log(\det(\boldsymbol{B}))}{n},$$

where $\alpha_\beta = \mathrm{gh}(\beta)^{2/(\beta-1)}$.

Justification. The above profile follows if we assume that $\|\mathbf{b}_{i+1}^*\|/\|\mathbf{b}_i^*\| = \alpha_\beta$ for all $1 \leq i < n$, and from the invariant $\sum \log(\|\mathbf{b}_i^*\|) = \log(\det(B))$. We now give a justification why $\alpha_\beta = \mathrm{gh}(\beta)^{2/(\beta-1)}$ by using the Gaussian heuristic. Because α_β only depends on the blocksize β, we can focus on a single BKZ block to determine this constant. We thus assume without loss of generality that our lattice $\mathcal{L}$ has dimension β and covolume 1. By the properties of BKZ we furthermore know that $\|\mathbf{b}_1^*\| = \|\mathbf{b}_1\| = \lambda_1(\mathcal{L})$. By the Gaussian heuristic, we thus have $\|\mathbf{b}_1^*\| = \lambda_1(\mathcal{L}) = \mathrm{gh}(\beta)$. Additionally, we have

$$\log\|\mathbf{b}_j^*\| = \log\|\mathbf{b}_1^*\| - (j-1)\log(\alpha_\beta) = \log(\mathrm{gh}(\beta)) - (j-1)\log(\alpha_\beta),$$

for all $1 \leq j \leq n$. By the fact that we assumed the covolume of $\mathcal{L}$ to be one, we have

$$0 = \frac{1}{\beta}\sum_{j=1}^{\beta} \log(\|\mathbf{b}_j^*\|) = \log(\mathrm{gh}(\beta)) - \frac{1}{2}(\beta-1)\log(\alpha_\beta),$$

where we used that $\frac{1}{\beta}\sum_{j=1}^{\beta}(j-1) = \frac{1}{2}(\beta-1)$. Solving for α_β gives $\alpha_\beta = \mathrm{gh}(\beta)^{2/(\beta-1)}$ as in the statement.

For block sizes exceeding 50 and reasonably small compared to the lattice rank (i.e., $\beta \ll n$), the GSA is quite precise. For accurate estimates for small $\beta < 50$ an experimentally computed value of α_β is used instead.

The Geometric Series Assumption should be used with care, especially for the start and the end of the basis profile. Namely, by the concavity of HKZ-reduced basis profiles in combination with the fact that the Gaussian heuristic is inherently an average-case heuristic, the profiles of BKZ tend to be slightly concave as well close to the start and end.

2.10 Algorithms for the Shortest Vector Problem

The BKZ algorithm requires an SVP oracle to reduce lattices. In this section we discuss the two most important heuristic algorithms for SVP: enumeration and sieving.

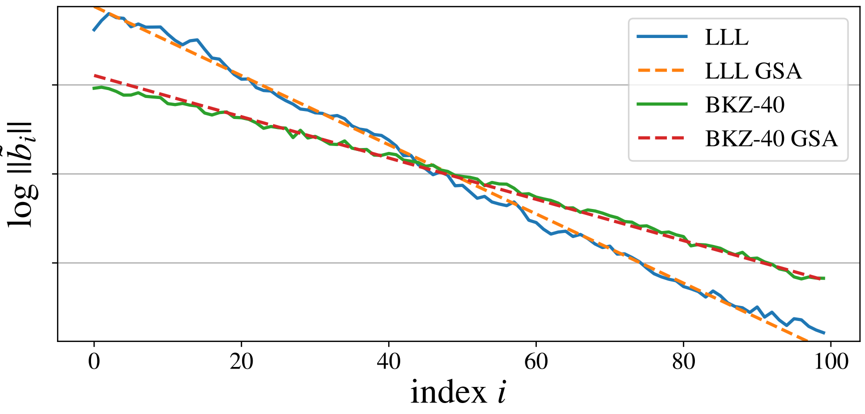


Figure 2.8: Typical basis profiles of LLL and BKZ compared with the predictions computed by the Geometric Series Assumption.

2.11 Enumeration

Enumeration algorithms can be seen as a generalization of Babai's nearest plane algorithm (Algorithm 1). Instead of making a greedy choice at each iteration to minimize $\langle \mathbf{e}, \mathbf{b}_i^* \rangle$, enumeration algorithms take multiple choices and recurse on those. A simple enumeration algorithm is presented in Algorithm 6. The complexity of enumeration heavily depends on how reduced the basis is. For example in the initial call there are already about $2R/\|\mathbf{b}_n^*\|$ choices, each of which starts a new enumeration call on a lattice of dimension $n-1$. For a badly reduced basis $\|\mathbf{b}_n^*\|^{-1}$ can be very large, while for a well-reduced basis it is smaller. Due to the recursion the other Gram-Schmidt norms $\|\mathbf{b}_{n-1}^*\|, \|\mathbf{b}_{n-2}^*\|, \ldots, \|\mathbf{b}_1^*\|$ are just as important, each leading to a multiplicative increase.

For an LLL reduced basis the cost of enumeration for $R = \lambda_1(\mathcal{L})$ is of order $2^{O(n^2)}$. For a HKZ-reduced basis the cost decreases to about $n^{O(n)}$. Enumeration algorithms are trivially parallelizable and use only a polynomial amount of memory. But their time-complexity is sub-optimal.

2.12 Sieving

Lattice sieving algorithms solve SVP in single-exponential time, making them asymptotically superior to enumeration techniques running in super-exponential time, at the cost of also using single-exponential space. The central idea of sieving algorithms is that given two lattice vectors $\mathbf{v}, \mathbf{w} \in \mathcal{L}$, their difference $\mathbf{v} - \mathbf{w} \in \mathcal{L}$ might be shorter. While in general this is not the case, given enough lattice vectors we can find enough of such pairs. Lattice sieving algorithms thus start with a large list of long lattice vectors, and try to find pairs of vectors

Algorithm 6: Standard enumeration algorithm.

```
Input   :
    • A basis $\boldsymbol{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice $\mathcal{L}$ with Gram-Schmidt
      orthogonalization $\boldsymbol{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$,
    • a target $\mathbf{t} \in \operatorname{span}(\mathcal{L})$ and a radius $R > 0$.
Output: A set $S$ of all lattice points $\mathbf{v} \in \mathcal{L}$ at distance $\|\mathbf{v} - \mathbf{t}\|$ at most
        $R$ from $\mathbf{t}$.
Enum($\boldsymbol{B}, \mathbf{t}, R$):
    $c \leftarrow \frac{\langle \mathbf{t}, \mathbf{b}_n^*}{\|\mathbf{b}_n^*\|}$
    $S \leftarrow \emptyset$
    for $x \in \mathbb{Z}$ such that $(c - x\|\mathbf{b}_n^*\|)^2 \leq R^2$ do
        $S' \leftarrow$ Enum$((\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}), \pi_{(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1})}(\mathbf{t} -$
          $z\mathbf{b}_n), \sqrt{R^2 - (c - x\|\mathbf{b}_n^*\|)^2})$
        $S \leftarrow S \cup \{x \cdot \mathbf{b}_n + s : s \in S'\}$
    end
    return $S$
```

that are close to each other. For each such pair the short difference vector is then inserted back into this list, possibly replacing longer vectors. This process is repeated until the list contains many short vectors, among which are the shortest ones.

Heuristically we need about $N = (4/3)^{n/2+o(n)}$ vectors to find enough close pairs. Naively this leads to an algorithm with running time of $N^2 = 2^{0.415n+o(n)}$. Using 'nearest-neighbor' techniques this can be reduced to an asymptotic runtime of $2^{0.292n+o(n)}$. Asymptotically the cost of running the BKZ algorithm with blocksize β is thus $2^{0.292\beta+o(\beta)}$.

The concrete performance of sieving algorithms is still an active field of research, and in the last years several polynomial and even sub-exponential speed-ups were discovered. As a result, sieving went from an asymptotic superior algorithm to a practical superior algorithm to enumeration. Recently, the crossover between the best enumeration and sieving algorithms was pushed as low as dimension 80. In Chapter 6 we will dive deeper into the concrete cost of the best sieving algorithm.

Chapter 3

Structured lattices

3.1 Introduction

3.1.1 Efficiency

In cryptography, one strives for security in the first place, but another property of a cryptographic protocol is also very important: *efficiency*. An inefficient cryptosystem that takes ages to encrypt will be competed away by their more efficient counterparts based, for example, on other hardness assumptions.

In lattice-based cryptography that uses general (unstructured) lattices, the largest inefficiency issue is mainly due to the size of the *key*, which consists of the basis of the lattice and requires about n^2 rather large numbers to store and transmit (this is not the full story, for a more in-depth explanation, see Section 4.3.1).

To circumvent this issue, so-called *structured lattices* were invented, whose bases have an specific shape so that the entire basis can be reconstructed from only partial information of this basis. For example, in so-called ideal lattices the entire basis can be reconstructed by giving just one single vector of the basis.

The main idea of structured lattices is that there is, apart from the additive lattice structure, another *multiplicative* ring-like structure. These lattices with this extra multiplicative structure arise naturally in the field of algebraic number theory, where they can be constructed out of ideals and modules.

3.1.2 Structure vs. Security

Intuitively, one might think that adding more structure to a certain cryptosystem allows for attacks that *exploit* this structure. Indeed, a vital part of the challenges in current cryptography is to add structure that *does* improve encryption and decryption efficiency, but *does not* decrease security.

Ideal lattices

The line of work of [CGS14; Cra+16; CDW17; Eis+14; PHS19] show that cryptosystems based on lattices with *too much structure*, namely *ideal lattices*, are potentially weaker than those based on generic lattices. This is because there exists a quantum algorithm that allows to find mildly short vectors in ideal lattices; those are lattice vectors within the bound $2^{\tilde{O}(\sqrt{t})} \det(I)^{1/t}$ of an ideal lattice I of a t-dimensional number field.

Though no candidates for standardization were based on ideal lattices, this line of work showed that, in cryptography, too much structure in lattices is to be avoided. A kind of structured lattice that is generally considered (much) better for cryptography purposes are *module lattices*.

Module lattices

Module lattices can be seen as *higher-rank generalizations* of ideal lattices, in the same way that matrices can be seen as higher-'rank' generalizations of numbers (the ordinary numbers are then the diagonal matrices).

Ideal lattices are a very special case of module lattices, they namely have *rank one*. The line of research described above seems *not to generalize* to module lattices of higher rank; it is believed among cryptographers that there is a computational barrier in between rank-one module lattices (ideal lattices) and rank-two (and higher) module lattices.

The belief in this computational barrier is partially because of a reduction for SVP in higher rank modules to SVP in rank 2 modules [Lee+19; MS19].

3.1.3 Module lattices in NIST candidates

For this reason, structured lattice-based cryptosystems are often based on module lattices of rank 2 or higher. All ModuleLWE-based cryptosystems have their ground in SVP in rank > 1 module lattices. RingLWE-based cryptosystems have a hardness that is somewhere 'in between' SVP in rank 1 and higher rank modules, as there is a reduction from IdealSVP to RingLWE [LS15], but RingLWE is also a rank one version of Module-LWE. For NTRU, recent work [PS21] shows that a similar 'sandwich' situation happens: a reduction from IdealSVP to the worst-case NTRU, and a reduction from NTRU to search-RingLWE. A recent work [FPS22] shows that NTRU is at least as hard as unique module-SVP in rank 2.

3.2 Number fields and number rings

In this section, we will quickly address algebraic number theory; for a more extensive treatment, we find Neukrich [NS13] a good reference. A number field K is a finite-dimensional field extension of the rational numbers $\mathbb{Q}$. In other words, $K \simeq \mathbb{Q}[X]/(f(X))$ for some irreducible polynomial $f(X) \in \mathbb{Q}[X]$. The dimension of K as a $\mathbb{Q}$-vector space is called the *degree* t of the number field.

Every number field element $\alpha \in K$ has a *minimal polynomial*, the unique monic, irreducible polynomial $m(X) \in \mathbb{Q}[X]$ that satisfies $m(\alpha) = 0$. If, additionally, this minimal polynomial of α lies in $\mathbb{Z}[X]$ (i.e., has integral coefficients) we call α an *integral element* of K. The integral elements in K together form a ring $\mathcal{O}_K$, which is called the *ring of integers* of K. Subrings of such a ring of integers of some number field K are called *number rings*.

The most commonly used number rings in cryptography are of the shape $R = \mathbb{Z}[x]/(\psi(x))$, where $\psi(x) \in \mathbb{Z}[x]$ is an irreducible polynomial. A typical example is the *power of two cyclotomic ring*, which is obtained by putting $\psi(x) = x^t + 1$ with $t = 2^k$ for some $k > 0$.

For example, by taking $t = 256$, one obtains the number ring of CRYSTALS-Kyber; this cryptosystem uses module LWE of rank $r = 2, 3$ or 4 over this ring [Bos+18; Alb+18] with $m = 2r$ samples (or equivalently $m = r$ samples and a short secret). We will see in Chapter 5 that attacks on these cryptosystems can be seen as BDD instances in module-lattices of rank $m = 4, 6, 8$.

3.2.1 The Minkowski embedding

Let $K = \mathbb{Q}[X]/(\psi(X))$ be a number field defined by the irreducible polynomial $\psi(X) \in \mathbb{Q}[X]$. This polynomial $\psi(X)$ has $\deg(\psi)$ distinct roots in the complex numbers $\mathbb{C}$. This gives $\deg(\psi)$ different *field embeddings* $K \hookrightarrow \mathbb{C}$, respectively, by sending $\bar{X} \in K = \mathbb{Q}[X]/(\psi(X))$ to any of the roots of ψ in $\mathbb{C}$. Those are all possible field embeddings of K into $\mathbb{C}$.

By concatenating these field embeddings next to each other, one obtains the *Minkowski embedding* $K \to \bigoplus_{\sigma:K\to\mathbb{C}} \mathbb{C},\ \alpha \longmapsto (\sigma(\alpha))_\sigma$. In most of the literature, the codomain of this Minkowski embedding is restricted to $K_\mathbb{R} = \{x_\sigma \in \bigoplus_{\sigma:K\to\mathbb{C}} \mathbb{C} \mid \overline{x_\sigma} = x_{\bar{\sigma}}\}$, where $\bar{\sigma}$ is the embedding $\bar{\sigma} : K \hookrightarrow \mathbb{C}$ obtained by applying first σ and then complex conjugation in $\mathbb{C}$. By component-wise addition and multiplication, $K_\mathbb{R}$ is an $\mathbb{R}$-algebra (and thus in particular an $\mathbb{R}$-vector space).

3.3 Ideal Lattices

Ideals

The image of a full-rank number ring R under the Minkowski embedding is a *discrete subgroup* in $K_\mathbb{R}$ (only considering the additive structure) [NS13, Ch. 1, § 4]. In other words, the number ring R forms a *lattice* under this embedding (see Figure 3.1). The same is true for any non-zero *ideal* of R. Recall that an ideal of R is a subgroup $I \subseteq R$ of the additive group of R that is stable under multiplication with elements in R, i.e., $R \cdot I \subseteq I$.

Ideal lattices

The image of an ideal $I \subseteq R$ under the Minkowski embedding is an example of an *ideal lattice*; it has the additive structure of a lattice and the ring-like

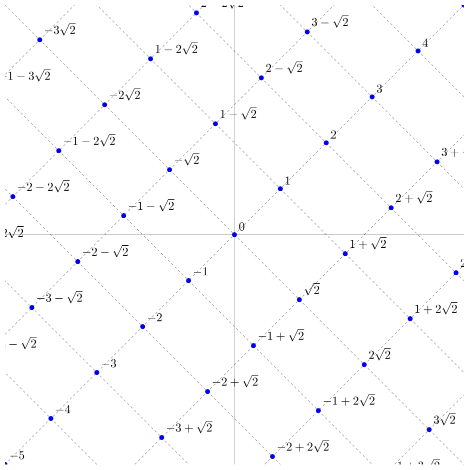


Figure 3.1: The number ring $\mathbb{Z}[\sqrt{2}]$ visualized on the real plane, using the Minkowski embedding, sending $\sqrt{2} \mapsto (\sqrt{2}, -\sqrt{2})$ and $1 \mapsto (1, 1)$.

structure of an ideal. More generally, an *ideal lattice* is defined as any non-zero lattice $L \subseteq K_{\mathbb{R}}$ that satisfies $R \cdot L \subseteq L$, where the action of R happens component-wise after the Minkowski embedding. Equivalently, considering $K_{\mathbb{R}}$ as an R-algebra, ideal lattices are discrete (rank 1) R-submodules of $K_{\mathbb{R}}$.

One can show that these lattices can be written as $x \cdot I$, where $I \subseteq R$ is a lattice, and where $x = (x_\sigma)_\sigma \in K_{\mathbb{R}}$. Here we mean by $x \cdot I \subseteq K_{\mathbb{R}}$ the lattice

$$x \cdot I = \{(x_\sigma \cdot \sigma(\alpha))_\sigma \mid \alpha \in I\}.$$

3.4 Module Lattices

Module lattices are generalizations of an ideal lattice, in the sense that they are discrete R-submodules of $K_{\mathbb{R}}^{r'}$ for some $r' \geq 1$. Concretely, a module lattice is any non-zero lattice $L \subseteq K_{\mathbb{R}}^{r'}$ that satisfies $R \cdot L \subseteq L$.

One can show that any such module lattice is of the shape

$$\mathcal{L} = \mathbf{b}_1 \cdot \mathfrak{a}_1 + \ldots + \mathbf{b}_k \cdot \mathfrak{a}_k \tag{3.1}$$

for suitable $\mathbb{R}$-linearly independent $\mathbf{b}_1, \ldots, \mathbf{b}_k \in K_{\mathbb{R}}^r$ and ideals $\mathfrak{a}_1, \ldots, \mathfrak{a}_k \subseteq R$. The set $(\mathbf{b}_j, \mathfrak{a}_j)_j$ is called a *pseudo-basis* of the module lattice L. A full treatment of pseudo-bases is beyond the scope of this text; we refer to Cohen's book [Coh12] for a more extensive treatment. An alternative view on modules-lattices is by using module filtrations, as in the paper of Mukherjee and Stephens-Davidowitz [MS19].

In this survey we will only treat *free module lattices*, which are modules of the shape as in Equation (3.1), where all ideals $\mathfrak{a}_i = \mathcal{O}_K$ are equal to the ring of integers. In that case, one can write the module lattice by an 'ordinary' basis, by putting $\mathbf{B} := (\mathbf{b}_1, \ldots, \mathbf{b}_k)$, by which we then mean the module lattice

$$\mathcal{L} = \mathbf{b}_1 \cdot \mathcal{O}_K + \ldots + \mathbf{b}_k \cdot \mathcal{O}_K \quad \text{(module lattice)}.$$

Note the analogy with 'ordinary' lattice bases, where the $\mathcal{O}_K$ is replaced by $\mathbb{Z}$:

$$\mathcal{L} = \mathbf{b}_1 \cdot \mathbb{Z} + \ldots + \mathbf{b}_k \cdot \mathbb{Z} \quad \text{(ordinary lattice)}.$$

As we will see in Chapter 8, many of the operations (row/column operations and Gram-Schmidt orthogonalization) on ordinary lattices have their generalizations to module lattices. With these generalizations it is then possible to phrase a module version of LLL (and BKZ), though these are not more powerful (in the pure lattice reduction sense) than the ordinary versions of these algorithms. This is the subject of Chapter 8.

Chapter 4

NTRU & LWE

4.1 Introduction

To do lattice-based cryptography, one needs hardness assumptions for concrete and computable lattices that can be easily constructed and amended by computers. Additionally, such a hardness assumption must allow for *trapdoors*, so that it is usable in a cryptographic context.

For example, the lattices considered must only involve integer arithmetic and preferably fast operations, whereas the hardness assumptions must have a tight relation with the 'natural' hard problems in lattices, like the Closest and the Shortest Vector Problem.

The two most famous of such cryptographic hardness assumptions for lattices are Learning With Errors, originating from Regev [Reg09], and NTRU, originating from Hoffstein, Pipher and Silverman [HPS98].

4.2 LWE

4.2.1 Introduction

In this subsection, we will explain at a high level about Learning With Errors (LWE), a lattice-based cryptography framework. To keep things simple, parameters are not yet instantiated and certain distributions are not yet made explicit.

Later on, whenever we discuss the specific variants of LWE, we will focus more on the precise instantiations and also focus on what instantiations are used in the NIST candidates.

4.2.2 The LWE setup

Learning with errors always works with modular arithmetic modulo some number $q \in \mathbb{N}_{>1}$, which is called the *modulus* and is often chosen to be prime or a prime power.

The secret key $\mathbf{s}$ in LWE is a *vector* $\mathbf{s} \in (\mathbb{Z}/q\mathbb{Z})^n$, i.e., $s_i \in \mathbb{Z}/q\mathbb{Z}$ for $i \in \{1, \ldots, n\}$. This secret vector $\mathbf{s}$ is sampled uniformly beforehand.

The public information consists of m slightly perturbed *random inner products with* $\mathbf{s}$. That is, the public information consists of m pairs of the shape $(\mathbf{a}, b)$ such that $b = \mathbf{a} \cdot \mathbf{s} + e \in \mathbb{Z}/q\mathbb{Z}$, where, in each pair $(\mathbf{a}, b)$, the vector $\mathbf{a}$ is drawn uniformly random from $(\mathbb{Z}/q\mathbb{Z})^n$ and e is drawn from a specific distribution over $\mathbb{Z}/q\mathbb{Z}$ close to zero. For intuition, it suffices to think of e to be uniformly drawn from $\{-2, -1, 0, 1, 2\} \pmod q$ for each pair. The dot in $\mathbf{a} \cdot \mathbf{s}$ denotes the *inner product* of the two vectors in $(\mathbb{Z}/q\mathbb{Z})^n$ and results in a number in $\mathbb{Z}/q\mathbb{Z}$.

For each of the m public pairs $(\mathbf{a}, b)$, the associated errors e are unknown to the public. Also, this error is sampled again for *each* pair. Both these facts are important to understand the hardness of the *LWE assumption*.

4.2.3 The LWE assumption

Recall that the public observer in the LWE setup gets to know m pairs of the shape $(\mathbf{a}, b)$, where $\mathbf{a} \in (\mathbb{Z}/q\mathbb{Z})^n$ and $b \in \mathbb{Z}/q\mathbb{Z}$. Here, the public observer knows that there *exists* some secret $\mathbf{s}$ such that $b = \mathbf{a} \cdot \mathbf{s} + e$, but does not know the *value* of $\mathbf{s}$.

The search-LWE assumption

The *search*-LWE assumption can be phrased as follows: given m such pairs of the shape $(\mathbf{a}, b)$ as in the LWE setup, it is *hard* to find the secret $\mathbf{s}$.

Different authors have different assumptions on this particular hardness, and this hardness also relies on the various parameters of the LWE setup: the modulus q, the dimension n, the number of samples m and the error distribution of the e.

The distinguishing-LWE assumption

The *distinguishing*-LWE assumption essentially states that the m samples of the shape $(\mathbf{a}, b)$ look very much like *uniform samples*. In other words, this assumption states that it is *hard* to algorithmically distinguish between m samples $(\mathbf{a}, b)$ from the LWE setup, and m uniform samples $(\tilde{\mathbf{a}}, \tilde{b})$, i.e., where $\tilde{\mathbf{a}}$ is uniformly random distributed over $(\mathbb{Z}/q\mathbb{Z})^n$ and $\tilde{b}$ is uniformly random distributed over $\mathbb{Z}/q\mathbb{Z}$ (and where each of the m uniform samples are independent of each other).

Difference between the two assumptions

In the search-LWE assumption the public observer is *promised* that the samples are from the LWE setup. That means that the public observer is given the (true) promise that the samples $(\mathbf{a}, b)$ are of the shape $b = \mathbf{a} \cdot \mathbf{s} + e$ for some secret $\mathbf{s}$ and some given distribution of the errors e.

In the decision-LWE assumption the public observer is given no such promise. In fact, deciding whether there *exists* such secret $\mathbf{s}$ (in the LWE setup) or not (in the uniform setup) is the computational task that is assumed to be hard.

Regev showed that these two assumptions are essentially equivalent, whenever q is bounded by a polynomial in n [Reg09; Pei09].

4.2.4 Encryption and decryption

Encryption in LWE relies on the fact that the public observer (which has m samples of the shape $(\mathbf{a}, b)$) is capable of making a *new* LWE sample with a slightly worse 'disturbance error'. Recall, an LWE sample is a pair $(\mathbf{a}, b)$ for which $b = \mathbf{a} \cdot \mathbf{s} + e$.

Indexing the m public samples as follows: $(\mathbf{a}^{(1)}, b^{(1)}), \ldots, (\mathbf{a}^{(m)}, b^{(m)})$ (satisfying $b^{(j)} = \mathbf{a}^{(j)} \cdot \mathbf{s} + e^{(j)}$ for all j), generating this new LWE sample by the public observer proceeds by taking a *small linear combination* of these samples:

$$(\mathbf{a}_{new}, b_{new}) = \sum_{j=1}^{m} c_j \cdot (\mathbf{a}^{(j)}, b^{(j)}) = (\sum_{j=1}^{m} c_j \mathbf{a}^{(j)}, \sum_{j=1}^{m} c_j b^{(j)}), \tag{4.1}$$

where the $c_j \in \mathbb{Z}$ are all *small* (say, $-1, 0$ or 1) and all arithmetic is modulo q.

Indeed, this looks like an LWE sample, as it satisfies

$$b_{new} = \sum_{j=1}^{m} c_j b^{(j)} = \sum_{j=1}^{m} c_j (\mathbf{a}^{(j)} \cdot \mathbf{s} + e^{(j)}) = \underbrace{\sum_{j=1}^{m} c_j \mathbf{a}^{(j)}}_{\mathbf{a}_{new}} \cdot \mathbf{s} + \underbrace{\sum_{j=1}^{m} c_j e^{(j)}}_{e_{new}} \tag{4.2}$$

$$= \mathbf{a}_{new} \cdot \mathbf{s} + e_{new}. \tag{4.3}$$

Here e_{new} is *defined* as the small linear combination of errors $\sum_{j=1}^{m} c_j e^{(j)}$, which must in general be of a larger size than the $e^{(j)}$ themselves. So the 'quality', say, of such a sample is worse than the 'real' LWE samples $(\mathbf{a}^{(j)}, b^{(j)})$.

Encryption

Encryption (by the public observer) of a single bit $B \in \{0, 1\}$ now happens by generating such a new (worse quality) random LWE sample $(\mathbf{a}_{new}, b_{new})$ by taking a random small linear combination of the public LWE samples $(\mathbf{a}^{(j)}, b^{(j)})$. By subsequently adding $B \cdot \lfloor q/2 \rceil$ to b_{new}, one obtains the encryption[1]. That is, generate random small $c_i \in \mathbb{Z}$ and put

$$\text{Enc}(B) = (\sum_{j=1}^{m} c_j \mathbf{a}^{(j)}, \sum_{j=1}^{m} c_j b^{(j)} + B \cdot \lfloor q/2 \rceil) = (\mathbf{a}_{new}, b_{new} + B \cdot \lfloor q/2 \rceil)$$

[1]Here, $\lfloor q/2 \rceil$ means rounding $q/2$ to the nearest integer

Decryption

As the receiver of the message is in possession of the secret $\mathbf{s}$, it is possible to compute $\mathbf{a}_{new} \cdot \mathbf{s}$ and subtract it from $b_{new} + B \cdot \lfloor q/2 \rceil$, which yields, by Equation (4.3),

$$b_{new} + B \cdot \lfloor q/2 \rceil - \mathbf{a}_{new} \cdot \mathbf{s} = \underbrace{b_{new} - \mathbf{a}_{new} \cdot \mathbf{s}}_{e_{new}} + B \cdot \lfloor q/2 \rceil = e_{new} + B \cdot \lfloor q/2 \rceil$$
$$\approx B \cdot \lfloor q/2 \rceil.$$

Where the last approximate equation follows from the fact that e_{new} is reasonably *small* with respect to q. Therefore, the receiver is capable to retrieve B; the receiver simply deduces that B must be equal to zero if $e_{new} + B \cdot \lfloor q/2 \rceil$ is small, and equal to one otherwise.

Idea of security

The rough, intuitive idea for the security of this scheme stems from the distinguishing-LWE assumption. As the pair $(\mathbf{a}_{new}, b_{new})$ is an LWE sample, the distinguishing-LWE assumption implies that this is algorithmically indistinguishable from a uniform sample in $(\mathbb{Z}/q\mathbb{Z})^m \times (\mathbb{Z}/qZ)$.

The reasoning why an attacker is then never able to retrieve the message bit B (without the secret key), generally happens by a proof by contradiction as follows. Suppose (as to obtain a contradiction) that an attacker has an algorithm $\mathcal{A}$ that *does allow* to retrieve the message bit B from the encryption $\text{Enc}(B)$. Then, this algorithm $\mathcal{A}$ essentially can distinguish between LWE-samples (a $B = 0$ encryption) and non-LWE-samples (a $B = 1$ encryption). This distinguishing power can then also be used to (at least partially) distinguish between *random samples* and LWE samples (in some quantified sense), contradicting the hardness assumption.

4.2.5 Formal definition of 'plain' LWE

In the preceding text, we informally explained 'plain' LWE. For the purposes of phrasing this problem as a lattice problem, it will be useful to stack the m samples $(\mathbf{a}^{(j)}, b^{(j)} = \mathbf{a}^{(j)} \cdot \mathbf{s} + e^{(j)})$ together in matrix notation. By putting the vectors $\mathbf{a}^{(j)} \in (\mathbb{Z}/q\mathbb{Z})^n$ as the *rows* of a matrix that we will name $\boldsymbol{A}$, putting $b^{(j)}$ into a column vector $\mathbf{b} \in (\mathbb{Z}/q\mathbb{Z})^m$ and $e^{(j)}$ into a column vector $\mathbf{e} \in (\mathbb{Z}/q\mathbb{Z})^m$, we obtain

$$\mathbf{b} = \boldsymbol{A} \cdot \mathbf{s} + \mathbf{e}.$$

This leads to the following general definition of the LWE distribution.

Definition 14 (LWE, [Reg09])**.** *Let n, m and q be positive integers, let χ be a probability distribution on $\mathbb{Z}^m$ and let $\mathbf{s} \in (\mathbb{Z}/q\mathbb{Z})^n$ be a 'secret vector'.*

We denote $L_{n,m,q,\chi}$ for the distribution on $(\mathbb{Z}/q\mathbb{Z})^{m\times n} \times (\mathbb{Z}/q\mathbb{Z})^m$ given by choosing $\boldsymbol{A} \in (\mathbb{Z}/q\mathbb{Z})^{m\times n}$ uniformly at random, $\mathbf{e} \leftarrow \chi$ (from $\mathbb{Z}^m$), and *outputting*

$$(\boldsymbol{A},\ \mathbf{b} = \boldsymbol{A} \cdot \mathbf{s} + \mathbf{e} \text{ mod } q)$$

We will call such a pair $(\boldsymbol{A},\ \mathbf{b} = \boldsymbol{A} \cdot \mathbf{s} + \mathbf{e} \text{ mod } q)$ *an LWE instance with parameters* (q, n, m, χ).

The number n is called the dimension, the number m the sample size, the number q is generally referred to as the modulus of the LWE problem, and the distribution χ over $\mathbb{Z}^m$ is called the *error distribution.*

Definition 15 (Search-LWE, Decision-LWE)**.** *Search-LWE is the problem of recovering the secret vector* $\mathbf{s}$ *from a sample* $(\boldsymbol{A}, \mathbf{b}) \leftarrow L_{n,m,q,\chi}$.
Decision-LWE is the problem of distinguishing between samples from $L_{n,m,q,\chi}$ *and from the uniform distribution* $\mathcal{U}_{n,m,q}$ *over* $(\mathbb{Z}/q\mathbb{Z})^{n \times m} \times (\mathbb{Z}/q\mathbb{Z})^m$.

Note that we left the distribution of the original secret vector open in these definitions. Originally, the secret $\mathbf{s}$ is sampled uniformly at random from $(\mathbb{Z}/q\mathbb{Z})^n$, and unless otherwise stated we assume that this is the case. For efficiency reasons concrete schemes often deviate from this however, for example, by sampling *small secrets.* Often this means that the secret coefficients are sampled from the same distribution as the error coefficients. This does not hurt the security as the uniform and small secret variant are actually equivalent up to the number of samples one gets. In particular, a small secret LWE instance with m samples can be turned into a uniform LWE instance with $m' = m + n$ samples, and the reverse reduction also applies [App+09]. The small secret variant can thus be seen as compressing $m+n$ samples in only m samples, thereby reducing the amount of information that has to be send. This is precisely what makes small secrets useful for concrete schemes.

Summary 1 (LWE assumption). The LWE assumption states that, for adequate parameters (q, n, m, χ), 'noisy' linear combinations $\mathbf{b} = \boldsymbol{A} \cdot \mathbf{s} + \mathbf{e} \bmod q$ are computationally indistinguishable from uniform vectors mod q, for a random $\boldsymbol{A}$.

4.3 Variants of LWE

Though there are many variants of LWE [PP19], they can be roughly divided into three categories: Ring-LWE, Module-LWE and the original 'plain' LWE. This ordinary 'plain' LWE is essentially what is explained and formally defined in the previous section.

The first two categories, Ring-LWE and Module-LWE differ from this plain LWE in the sense that (part of) the inner product $(\mathbf{a} \cdot \mathbf{s})$ is replaced by multiplication in a ring. For Ring-LWE the inner product is *fully* replaced by a ring multiplication, whereas in Module-LWE this inner product is only (in some sense) *partially* replaced by ring multiplication. This is primarily done to decrease the key size and ciphertext size of the cryptosystem (see Section 4.3.1).

To summarize, the Ring-LWE variant is the 'most structured' one, the Module-LWE variant slightly less structured, and the original 'plain' LWE is what one could call ‘unstructured’, in the sense that its arithmetic is not tied to a ring other than $\mathbb{Z}$ or quotient groups thereof.

Remark 4. *Among the early NIST candidates, there are also other variants on LWE, most notably, Polynomial-LWE (PLWE), Order-LWE (OLWE), Middle-Product LWE (MPLWE) and Integer (Module) LWE (I(M)LWE). The first three are more or less equivalent to Ring-LWE [PP19], whereas the situation for Integer (Module) LWE’s is less clear. Gu [Gu19] shows that there is a relation between Ring-LWE and Integer Ring LWE, but the author of ThreeBears [HR17] suggest that this relation is not tight for their cryptosystem.*

Another variant of LWE is called Learning With Rounding (LWR), where the main difference is the way a ‘disturbance’ e is applied to the linear relations. In LWE that disturbance happens in a random fashion, whereas LWR intends to remove this randomness by making such disturbance deterministic by means of ‘rounding’. This in order to avoid attacks that exploit weaknesses due to the random sampling, like timing attacks (for which in LWE additional measures are taken).

4.3.1 Plain LWE and its key and ciphertext size

For most NIST candidates that have an LWE-alike cryptosystem, the number of samples m is of the order of n; in many cases $m = 2n$ (or $m = n$ in the small secret variant). This means that the public key $(\boldsymbol{A}, \mathbf{b})$ consists of an $m \times n$ matrix and a m-dimensional vector, both with coefficients in $\mathbb{Z}/q\mathbb{Z}$. One large advantage of LWE is that the matrix $\boldsymbol{A}$ is random and thus one could derive it with a pseudorandom function from a small seed. So the (pseudorandom) public key $(\boldsymbol{A}, \mathbf{b})$ could be transferred using $\Theta(m \log_2(q)) = O(n \log_2(q))$ bits. Now let’s have a look at the encryption procedure, as shown in Section 4.2.4 the encryption of a single bit produces a ciphertext $(\mathbf{a}, b) \in (\mathbb{Z}/q\mathbb{Z})^n \times \mathbb{Z}/q\mathbb{Z}$ in the order of $\Theta(n \log_2(q))$ bits. Sending B bits of information therefore requires $\Theta(B \cdot n \log_2(q))$ bits.

Generally, in the setting of a KEM, the dimension n and the shared key size B are roughly proportional to the logarithm of the advance of calculation speed of computers, say λ (also known as the bit-security parameter); intuitively, to withstand an attack of a twice faster computer, the lattice dimension n and the shared key size B needs to be (additively) increased by a constant. The ciphertext of a plain LWE cryptosystem is of size roughly $nB \log_2(q) = \tilde{\Omega}(\lambda^2)$, and needs to be transmitted over a network, say, the internet (just to *share* the key). So, in order for such a cryptosystem to be also usable in the future, the network speed should keep up quadratically with the logarithm of the calculation speed. Some would argue that network speed depends on large-scale infrastructure, and this is therefore not something to be expected. Others say that such a ciphertext size of $\tilde{\Omega}(\lambda^2)$ is untenable with respect to other (pre- and post-quantum) cryptosystems and will be competed away by their more efficient counterparts.

Note that the public-key size is only of order $\tilde{\Omega}(\lambda)$, while the ciphertext size is of order $\tilde{\Omega}(\lambda^2)$. By applying additional trade-offs these sizes can be balanced to obtain a total size in the order of $\tilde{\Omega}(\lambda^{3/2})$. The idea is to have roughly $O(\sqrt{\lambda})$ secret vectors s hidden in the public key, and roughly $O(\sqrt{\lambda})$ ciphertexts, which when properly combined can transfer $O(\sqrt{\lambda}^2) = O(\lambda)$ bits of information. This was for example done in FrodoKEM [Nat17]. Nevertheless, a key and ciphertext size in the order of $\tilde{\Omega}(\lambda^{3/2})$ is still rather large for practical purposes.

So, for future implementation one could state that plain LWE has key sizes that are too large for everyday use. Even at the current time NIST considers the key sizes of cryptosystems based on plain LWE too large for (direct) standardization [Moo+20, §3.6], though they do state for example that the plain LWE-based FrodoKEM "suitable for use cases where the high confidence in the security of unstructured lattice-based schemes is much more important that performance" [Moo+20, §3.6].

Summary 2 (Reason for using 'structured' LWE). 'Plain' LWE with parameters $q, n, m = \Theta(\lambda)$ has key and ciphertext sizes $\tilde{\Omega}(\lambda^2)$ or with some improvements $\tilde{\Omega}(\lambda^{3/2})$, which is generally considered too large for everyday use. This is the reason why 'structured' variants of LWE are considered.

Public key and ciphertext sizes of Ring-LWE and Module-LWE

The structured variants of LWE (i.e., Ring-LWE and Module-LWE) are invented to diminish the public key size and cipher text size by using ring arithmetic and to improve calculation speed of multiplicative operations by means of Fourier transforms.

Due to the ring arithmetic, Ring and Module-LWE have public keys of size $O(n \log q)$ and cipher texts of size $O(n \log q)$ for an n-bit message, with maybe some additional logarithmic factors. We now examine how this efficiency improvement is obtained.

4.3.2 Ring-LWE

NIST finalists

As none of the NIST finalist are based on Ring-LWE (but many *candidates* are), we will only briefly treat this variant. Covering Ring-LWE first will ease the explanation of Module-LWE.

Remark 5. *The reason that the NIST finalist lack Ring-LWE based schemes is mostly due to the preference for low rank Module-LWE based cryptosystems instead, as the latter are of comparable efficiency but have less algebraic structure (e.g., [Moo+20, §3.12]).*

Ring-LWE

In Ring-LWE, one attempts to lower this public key size by using *ring arithmetic*. By using the multiplicative structure of the ring, one can 'compactify' the matrix

NIST candidates	Assumption	Security (bits)	Public key (bytes)
FrodoKEM (FrodoKEM–976)	LWE	154	15632
Round 5 (R5N1_3PKE_0d)	LWR	175	9660
CRYSTALS-Kyber (Kyber768)	Module-LWE	164	1184
SABER	Module-LWR	185	992
Three Bears (MamaBear)	I-MLWE	213	1194
NTRU (ntruhps4096821)	NTRU	178	1230
NTRUPrime (sntrup761)	NTRU	139	1158
LAC (LAC-192-v3a)	RLWE	267	1056
NewHope (NewHope1024)	RLWE	233	1824

Table 4.1: The NIST 3rd round lattice-based KEM finalists and alternatives with their assumptions and key sizes for Category III security [Rav+21], with the exception of NewHope, which has no category III proposal (instead category V is listed). Note the large public key sizes in the unstructured LWE-based cryptosystems; almost ten times larger than those based on structured lattices.

$\boldsymbol{A}$ from Definition 14.

In most Ring-LWE schemes, the arithmetic happens in the ring $R = \mathbb{Z}[x]/(x^t + 1)$, with $t = 512, 1024$ or 2048; so we will focus on this ring. Additionally we have a prime number q, which is of reasonably large size[2]. A Ring-LWE sample is of the form $(\mathbf{a}_R, \mathbf{b}_R) \in (R/qR) \times (R/qR)$ where

$$\mathbf{b}_R = \mathbf{a}_R \cdot \mathbf{s}_R + \mathbf{e}_R \bmod q.$$

Here, the subscript R of the element indicates that we are computing in the ring R, as to make the difference with plain LWE. The secret key $\mathbf{s}_R \in R$ is random (in most schemes chosen to have small coefficients), $\mathbf{a}_R$ is uniformly randomly drawn from R/qR, and $\mathbf{e}_R$ generally is drawn from a distribution over R/qR that is concentrated around 0 (think about a discrete Gaussian centered at 0 with a deviation much smaller than q). The multiplication operation is *ring multiplication*, not the inner product.

Ciphertext sizes of Ring-LWE

As argued before, the ciphertext sizes of Ring-LWE are significantly smaller ($O(t \log q)$ for a t-bit message) than that of ordinary LWE. This can be seen by the following observation. For ordinary LWE, to encrypt a single bit, the tuple $(\mathbf{a}_{new}, b_{new}) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$ must be sent, of size $O(n \log q)$, see Section 4.2.4.

For Ring-LWE, one can pick a random element $\mathbf{y}_R = \sum_{j=0}^{t-1} y_i x^i \in R/qR$ with entries in $\{-1, 0, 1\}$ and translate an t-bit string $\kappa_0 \kappa_2 \dots \kappa_{t-1}$ into an element in R by putting $\mathbf{k}_R = \lfloor q/2 \rceil \sum_{j=0}^{t-1} \kappa_j x^j \in R$.

Then, encryption of $\mathbf{k}$ is done by sending over

$$\mathbf{y}_R \cdot \mathbf{a}_R \text{ and } \mathbf{y}_R \cdot \mathbf{b}_R + \mathbf{k}_R \bmod q.$$

[2]It can range from $q = 251$ to $q = 16900097$, e.g. [Alb+18]

Note that this are two ring elements from R, and thus take space $O(t \log q)$.

Decryption then uses the secret key $\mathbf{s}_R$, by computing

$$\begin{aligned} & \mathbf{y}_R \cdot \mathbf{b}_R + \mathbf{k}_R - \mathbf{y}_R \cdot \mathbf{a}_R \cdot \mathbf{s}_R & (4.4) \\ =& \mathbf{y}_R \cdot (\mathbf{a}_R \cdot \mathbf{s}_R + \mathbf{e}_R) + \mathbf{k}_R - \mathbf{y}_R \cdot \mathbf{a}_R \cdot \mathbf{s}_R & (4.5) \\ =& \mathbf{y}_R \cdot \mathbf{e}_R + \mathbf{k}_R & (4.6) \end{aligned}$$

Now, by using that $\mathbf{y}_R$ and $\mathbf{e}_R$ have very small coefficients, and therefore their product also has very small coefficients, whereas $\mathbf{k}_R$ has coefficients that are either $\lfloor q/2 \rceil$ or zero, one retrieves the i-th bit of the bit message $\kappa_0 \ldots \kappa_{t-1}$ by observing whether the i-th coefficient of $\mathbf{y}_R \cdot \mathbf{e}_R + \mathbf{k}_R$ is closer to $q/2$ or to 0.

Remark 6. *Something very similar happens in Module-LWE (with a small loss), which has thus the same efficiency benefits as RingLWE in terms of public key size and ciphertext size.*

Summary 3 (Public key size and ciphertext size of LWE versus Ring/Module-LWE). For encryption of a t-bit message 'ordinary' LWE requires transmission of $O(t^2 \log q)$ bits, whereas Ring-LWE (and Module-LWE) requires $O(t \log q)$ bits. FrodoKEM, by using a special trick, requires transmission of $O(t^{3/2} \log q)$ bits for a t-bit message.

Seeing Ring-LWE as an LWE instance

In plain LWE, the expression

$$\mathbf{b} = \boldsymbol{A} \cdot \mathbf{s} + \mathbf{e} \bmod q \quad \text{(LWE)} \tag{4.7}$$

involves vectors $\mathbf{b}, \mathbf{e}, \mathbf{s} \in (\mathbb{Z}/q\mathbb{Z})^n$ and a matrix $\boldsymbol{A} \in (\mathbb{Z}/q\mathbb{Z})^{n \times m}$. In Ring-LWE, the expression

$$\mathbf{b}_R = \mathbf{a}_R \cdot \mathbf{s}_R + \mathbf{e}_R \bmod q \quad \text{(Ring-LWE)} \tag{4.8}$$

involves ring elements $\mathbf{b}_R, \mathbf{e}_R, \mathbf{s}_R, \mathbf{a}_R \in R/qR$ and ring multiplication and addition.

In the following explanation we will show that the ring-LWE is a variant of LWE where $\mathbf{b}_R$, $\mathbf{e}_R$ and $\mathbf{s}_R$ can be seen as *vectors* in $(\mathbb{Z}/q\mathbb{Z})^n$ and the ring multiplication $\mathbf{a}_R \cdot -$ corresponds to multiplication by a matrix $M_{\mathbf{a}_R}$ associated with $\mathbf{a}_R$, called the *multiplication matrix*. Such a matrix is of a special structured shape, sometimes called 'anti-circulate'.

An element $\mathbf{c}_R \in R/qR$ is a polynomial of which the coefficients can be chosen to be in $\mathbb{Z}/q\mathbb{Z}$.

$$\mathbf{c}_R = \sum_{j=0}^{t-1} c_j x^j \text{ with } c_j \in \mathbb{Z}/q\mathbb{Z}. \tag{4.9}$$

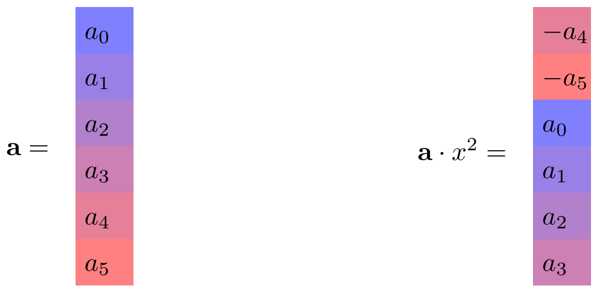


Figure 4.1: The anti-circular multiplication of $\mathbf{a}_R = a_0 + a_1x^1 + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$ by x^2 in the ring $R = \mathbb{Z}[x]/(x^6+1)$. This yields $\mathbf{a}_R \cdot x^2 = -a_4 + -a_5x^1 + a_0x^2 + a_1x^3 + a_2x^4 + a_3x^5$. Note that $x^6 = -1$ in such a ring, causing the negative sign.

This polynomial can just be represented by the coefficients, which gives a vector $(c_0, \ldots, c_{t-1}) \in (\mathbb{Z}/q\mathbb{Z})^t$, which we will denote $\mathbf{c}$ (without the subscript R). This is how we can see $\mathbf{b}_R, \mathbf{s}_R$ and $\mathbf{e}_R \in R$ as vectors $\mathbf{b}, \mathbf{s}, \mathbf{e} \in (\mathbb{Z}/q\mathbb{Z})^t$. So, we have an additive group isomorphism between R/qR and $(\mathbb{Z}/q\mathbb{Z})^t$. Note that the standard basis vectors $\boldsymbol{\epsilon}_j$ are just the *monomials* x^j for $j \in \{0, \ldots, t-1\}$.

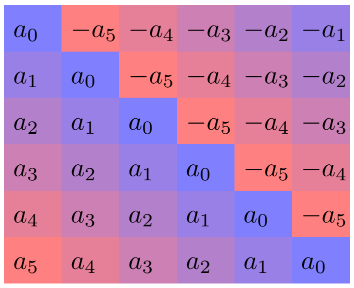
The element $\mathbf{a}_R$ gets a different treatment because that ring element is involved in the multiplication. The action $\mathbf{c}_R \mapsto \mathbf{a}_R \cdot \mathbf{c}_R$ maps R/qR to R/qR in a linear fashion, as $\mathbf{a}_R(\mathbf{c}_R + \mathbf{d}_R) = \mathbf{a}_R\mathbf{c}_R + \mathbf{a}_R\mathbf{d}_R$. By using the additive isomorphism $R/qR \simeq (\mathbb{Z}/q\mathbb{Z})^t$, one can thus see this map 'multiplying by $\mathbf{a}_R$' as a *linear map*. Thus, we can represent it by a matrix by looking how it acts on the standard basis vectors $\boldsymbol{\epsilon}_k \leftrightarrow x^k$. Writing $\mathbf{a}_R = \sum_{j=0}^{t-1} a_jx^j \in R/qR$, we have

$$\mathbf{a}_R \cdot x^k = \left(\sum_{j=0}^{t-1} a_jx^j\right) \cdot x^k = \sum_{j=0}^{t-1} a_jx^{j+k} = -\sum_{j=0}^{k-1} a_{t-k+j}x^j + \sum_{\ell=k}^{t-1} a_{\ell-k}x^\ell,$$

see also Figure 4.1. So the matrix of multiplication by $\mathbf{a}_R$ has a special shape, which is often called anti-circulant (due to the negative sign after the 'circular' movement). For the multiplication matrix $M_{\mathbf{a}_R}$ of $\mathbf{a}_R = \sum_{j=0}^{t-1} a_jx^j$ we can write

$$M_{\mathbf{a}_R} = \left(\mathrm{sign}(j-k) \cdot a_{j-k \bmod t}\right)_{\substack{j=0,\ldots,t-1 \\ k=0,\ldots,t-1}},$$

where $j-k \bmod t$ is chosen to be in $\{0, \ldots, t-1\}$ and $\mathrm{sign}(j-k) = 1$ if $j-k \geq 0$ and -1 otherwise. An example of how such a anti-circulant matrix looks like for the case $R = \mathbb{Z}[x]/(x^6+1)$ can be seen in Figure 4.2.



$$\begin{array}{cccccc} a_0 & -a_5 & -a_4 & -a_3 & -a_2 & -a_1 \\ a_1 & a_0 & -a_5 & -a_4 & -a_3 & -a_2 \\ a_2 & a_1 & a_0 & -a_5 & -a_4 & -a_3 \\ a_3 & a_2 & a_1 & a_0 & -a_5 & -a_4 \\ a_4 & a_3 & a_2 & a_1 & a_0 & -a_5 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \end{array}$$

Figure 4.2: An example of an anti-circulant multiplication matrix of $\mathbf{a}_R \in R$ in the ring $R = \mathbb{Z}[x]/(x^6 + 1)$.

The 'compactified' matrix $\boldsymbol{A}$

So, concluding, one can see that an Ring-LWE sample

$$\mathbf{b}_R = \mathbf{a}_R \cdot \mathbf{s}_R + \mathbf{e}_R \bmod q$$

can be seen as a plain LWE sample by seeing the polynomials $\mathbf{b}_R, \mathbf{s}_R$ and $\mathbf{e}_R$ as vectors in $(\mathbb{Z}/q\mathbb{Z})^t$ by considering their coefficients, and by putting $\boldsymbol{A} = M_{\mathbf{a}_R}$, the anti-circulant multiplication matrix of $\mathbf{a}_R$.

$$\mathbf{b} = \underbrace{\boldsymbol{A}}_{=M_{\mathbf{a}_R}} \cdot \mathbf{s} + \mathbf{e} \bmod q$$

Note that the matrix of Ring-LWE is fully determined by the first column; the rest of the column actually follows from the 'anti-circulant' rule. Therefore, to send over this multiplication matrix, we only need to send the first column, or equivalently, the coefficients of the polynomial $\mathbf{a}_R$.

Summary 4 (Ring-LWE as 'plain' LWE). In essence one can see Ring-LWE as a special version of plain LWE in which the matrix $\boldsymbol{A}$ is of such a shape that it is fully defined by its first column (and where the rest of the columns are defined by a linear transformation of the first column), see Figure 4.2.

Remark 7. *In Section 5.1, we will deduce that we can see (any variant of) LWE as a BDD instance a very much related lattice whose dimension equals the number of samples m. Then by embedding this BDD target we can reduce it to an (unique)SVP-instance in a lattice of dimension $m + 1$. For the system to have a unique solution the number of samples is generally at least $n + 1$ or often even $2n$ where n is the LWE dimension or rank. In particular, note that the dimension of the lattice problem is not directly determined by the LWE dimension, but more by the number of samples.*

In a very similar way, one can see Ring-LWE with $m \geq 2$ samples as a BDD instance in a rank m module lattice (hence keeping the structure), or a (unique)SVP-instance in a rank $m + 1$ module lattice.

For Module-LWE of rank r' with $m \geq r' + 1$ samples, essentially the same reduction applies; it can be considered as a (unique)SVP-instance in a rank $m+1$ module lattice, again by embedding the error in a new rank.

4.3.3 Module-LWE

NIST finalists

All LWE-based NIST finalists are based on Module-LWE of rank $2, 3, 4$ or 5. The module ranks of the earlier NIST *candidates* based on Module-LWE does not exceed 5 either. The underlying ring R is, in all MLWE (and MLWR) cases, a cyclotomic ring of the shape $\mathbb{Z}[x]/(x^t + 1)$ for t a power of two, or $\mathbb{Z}[x]/(x^t + x^{t-1} + \ldots + x + 1)$, for $t + 1$ a prime number [Alb+18].

Remark 8. *The low rank of module-LWE is for efficiency; the lower the rank, the more 'compactified' the public key can be stored and sent, and the more efficient arithmetic in the underlying ring is. Module-LWE of rank 1 is just equal to Ring-LWE.*

NIST has a slight preference for (low rank but ≥ 2) module-LWE over Ring-LWE [Moo+20, §3.12], as Module-LWE is generally believed (but yet unproven) to be a more difficult problem to tackle algorithmically than Ring-LWE (which is rank 1 Module-LWE).

Module-LWE

Module-LWE is some 'blend' between Ring-LWE and plain LWE. It has the ring multiplication of Ring-LWE, and a kind of (different) inner product from LWE.

We will denote r' for the *rank* of the Module-LWE problem. Given a *secret* $(\mathbf{s}_R^{(1)}, \ldots, \mathbf{s}_R^{(r')}) \in R^{r'}$ consisting of a r'-tuple of ring elements, a Module-LWE sample consists of $((\mathbf{a}_R^{(1)}, \ldots, \mathbf{a}_R^{(r')}), \mathbf{b}_R) \in R^{r'} \times R$ where

$$\mathbf{b}_R = \mathbf{a}_R^{(1)} \cdot \mathbf{s}_R^{(1)} + \ldots + \mathbf{a}_R^{(r')} \cdot \mathbf{s}_R^{(r')} + \mathbf{e}_R \bmod q \qquad \text{(rank } r' \text{ module-LWE)},$$

where the subscripts R indicate that an element is from R (multiplication is in R as well), where all $\mathbf{a}_R^{(j)}$ are uniformly randomly chosen from R/qR and where $\mathbf{e}_R \in R$ follows a specific distribution concentrated around 0.

A number m of such samples of the shape $((\mathbf{a}_R^{(1)}, \ldots, \mathbf{a}_R^{(r')}), \mathbf{b}_R) \in R^{r'} \times R$ can be put in a *matrix* form, *with coefficients in* R/qR, like with LWE. The m different $\mathbf{b}_R$ and $\mathbf{e}_R$ can be assembled into m-dimensional row vectors with entries in R/qR, and the secret $(\mathbf{s}_R^{(1)}, \ldots, \mathbf{s}_R^{(r')}) \in R^{r'}$ into a r'-dimensional column vector with entries in R/qR.

$$\begin{bmatrix} \mathbf{b}_R^{(1)} \\ \vdots \\ \mathbf{b}_R^{(m)} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{a}_R^{(1,1)} & \ldots & \mathbf{a}_R^{(1,r')} \\ \vdots & \ddots & \vdots \\ \mathbf{a}_R^{(m,1)} & \ldots & \mathbf{a}_R^{(m,r')} \end{bmatrix}}_{\boldsymbol{A}_R} \cdot \begin{bmatrix} \mathbf{s}_R^{(1)} \\ \vdots \\ \mathbf{s}_R^{(r')} \end{bmatrix} + \begin{bmatrix} \mathbf{e}_R^{(1)} \\ \vdots \\ \mathbf{e}_R^{(m)} \end{bmatrix} \bmod q \tag{4.10}$$

The subscript 'R' in the matrix $\boldsymbol{A}_R$ indicates that this matrix has entries in R/qR, as opposed to $\boldsymbol{A}$ in plain LWE, which has entries in $\mathbb{Z}/q\mathbb{Z}$.

Definition 16 (Module-LWE, e.g. [PP19])**.** *Let n, m and q be positive integers, let R be a ring whose additive group is isomorphic with $\mathbb{Z}^n$, let χ be a probability distribution on R^m and let $\mathbf{s} \in R^{r'}$ be a 'secret vector'.*

We denote $L_{m,q,\chi,R,r'}$ for the distribution on $(R/qR)^{m\times r'} \times (R/qR)^m$ given by choosing $\boldsymbol{A} \in (R/qR)^{m\times r'}$ uniformly at random, $(\mathbf{e}_R^{(1)}, \dots, \mathbf{e}_R^{(m)}) \leftarrow \chi$ (from R^m), and outputting

$$(\boldsymbol{A}_R, (\mathbf{b}_R^{(1)}, \dots, \mathbf{b}_R^{(m)}))$$

where $\boldsymbol{A}_R$ and $(\mathbf{b}_R^{(1)}, \dots, \mathbf{b}_R^{(m)})$ are as in Equation (4.10).

Similar to plain LWE, we then have the following two problems.

Definition 17 (Search-MLWE, Decision-MLWE)**.** *Search-MLWE is the problem of recovering the secret vector $\mathbf{s}$ from a sample $(\boldsymbol{A}_R, (\mathbf{b}_R^{(1)}, \dots, \mathbf{b}_R^{(m)})) \leftarrow L_{m,q,\chi,R,r'}$.*
Decision-MLWE is the problem of distinguishing between samples from $L_{m,q,\chi,R,r'}$ and from the uniform distribution $\mathcal{U}_{m,q,R,r'}$ over $(R/qR)^{m\times r'} \times (R/qR)^m$.

Seeing Module-LWE as a plain LWE instance

Likewise as with what we did with Ring-LWE, we would like to write the *multiplication matrix* of the action of the matrix $\boldsymbol{A}_R \in (R/qR)^{m\times r'}$ on the secret vector

$$\begin{bmatrix} \mathbf{s}_R^{(1)} \\ \vdots \\ \mathbf{s}_R^{(r')}. \end{bmatrix} \tag{4.11}$$

As in the Ring-LWE setting, we can essentially write each $\mathbf{s}_R^{(j)} \in R = \mathbb{Z}[x]/(x^n + 1)$ as a vector in $(\mathbb{Z}/q\mathbb{Z})^n$ by considering the coefficients of the polynomial (modulo q).

To see what the multiplication matrix of $\boldsymbol{A}_R$ looks like, it is insightful to consider the rank two case ($r' = 2$) with two samples[3] ($m = 2$). In that case, the secret is of the shape $(\mathbf{s}_R^{(1)}, \mathbf{s}_R^{(2)}) \in (R/qR)^2$, and we have two LWE samples

$$((\mathbf{a}_R^{(1,1)}, \mathbf{a}_R^{(1,2)}), \mathbf{b}_R^{(1)}) \in R^{r'} \times R$$
$$((\mathbf{a}_R^{(2,1)}, \mathbf{a}_R^{(2,2)}), \mathbf{b}_R^{(2)}) \in R^{r'} \times R$$

[3]This is not a real-life example, for multiple reasons: in actual cryptosystems the number of samples is generally twice the dimension to avoid the linear system to be underdetermined, and the underlying number ring has a larger degree (e.g., CRYSTALS-Kyber uses a degree 256 number field).

satisfying

$$\begin{aligned}\mathbf{b}_R^{(1)} &= \mathbf{a}_R^{(1,1)} \cdot \mathbf{s}_R^{(1)} + \mathbf{a}_R^{(2,1)} \cdot \mathbf{s}_R^{(2)} + \mathbf{e}_R^{(1)} \mod q\\ \mathbf{b}_R^{(2)} &= \mathbf{a}_R^{(1,2)} \cdot \mathbf{s}_R^{(1)} + \mathbf{a}_R^{(2,2)} \cdot \mathbf{s}_R^{(2)} + \mathbf{e}_R^{(2)} \mod q,\end{aligned}$$

or, equivalently,

$$\begin{bmatrix}\mathbf{b}_R^{(1)}\\ \mathbf{b}_R^{(2)}\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{a}_R^{(1,1)} & \mathbf{a}_R^{(1,2)}\\ \mathbf{a}_R^{(2,1)} & \mathbf{a}_R^{(2,2)}\end{bmatrix}}_{\boldsymbol{A}_R} \cdot \begin{bmatrix}\mathbf{s}_R^{(1)}\\ \mathbf{s}_R^{(2)}\end{bmatrix} + \begin{bmatrix}\mathbf{e}_R^{(1)}\\ \mathbf{e}_R^{(2)}\end{bmatrix} \mod q.$$

The action of the left part of the matrix $\mathbf{a}_R^{(1,1)}, \mathbf{a}_R^{(1,2)}$ only acts on the upper component of the secret $\mathbf{s}_R^{(1)}$, and, similarly, the right part of the matrix on the lower component of the secret vector). Therefore, one can see the multiplication matrix of $\boldsymbol{A}_R$, denoted $M_{\boldsymbol{A}_R}$, acting on the *coefficients of the secret* $(\mathbf{s}_R^{(1)}, \mathbf{s}_R^{(2)}) \in (R/qR)^2$ as a block matrix consisting of the respective multiplication matrices.

$$M_{\boldsymbol{A}_R} = \begin{bmatrix} M_{\mathbf{a}_R^{(1,1)}} & M_{\mathbf{a}_R^{(1,2)}}\\ M_{\mathbf{a}_R^{(2,1)}} & M_{\mathbf{a}_R^{(2,2)}}\end{bmatrix} \in (\mathbb{Z}/q\mathbb{Z})^{2n\times 2n} \qquad \text{(Module-LWE, } r' = 2, m = 2\text{)}$$

An example of such a multiplication matrix can be seen in Figure 4.3. We just treated the case of rank 2 ($r' = 2$) and two samples ($m = 2$). In the general case, the multiplication matrix has the following shape, by the very same argument.

$$M_{\boldsymbol{A}_R} = \begin{bmatrix} M_{\mathbf{a}_R^{(1,1)}} & \dots & M_{\mathbf{a}_R^{(1,r')}}\\ \vdots & \ddots & \vdots\\ M_{\mathbf{a}_R^{(m,1)}} & \dots & M_{\mathbf{a}_R^{(m,r')}}\end{bmatrix} \in (\mathbb{Z}/q\mathbb{Z})^{mn\times rn}$$

Summary 5 (Module-LWE as 'plain' LWE). Like with Ring-LWE, we can in essence see Module-LWE as a special version of plain LWE in which the matrix $\boldsymbol{A}$ has a shape that is fully determined by the columns $1, n+1, \dots, (r'-1)n+1$. The rest of the columns are a linear transformation of the defining column directly left to them.

Remark 9. *A rank r' module over a dimension t ring yields a lattice of dimension $d = r' \cdot t$. For fixed lattice dimension d, one can increase r' (and accordingly decrease t) in order to have less structure. In this way one can choose the amount of 'structuredness' in the scheme.*

For example, for a highly structured scheme one can put $r' = 1$ (Ring-LWE), for a slightly less structured LWE-based scheme one put r' slightly larger, like $2, 3, 4, 5$ (low-rank Module-LWE). For (almost) no structure, one can put r' to be close to d, that is, choose the ring dimension t to be small (high-rank Module-LWE). For $r' = d$ and $t = 1$ one recovers plain LWE. So in a sense, Module-LWE 'interpolates' between Ring-LWE and plain LWE.

$$\begin{array}{cccccc|cccccc}
a_0 & -a_5 & -a_4 & -a_3 & -a_2 & -a_1 & a'_0 & -a'_5 & -a'_4 & -a'_3 & -a'_2 & -a'_1 \\
a_1 & a_0 & -a_5 & -a_4 & -a_3 & -a_2 & a'_1 & a'_0 & -a'_5 & -a'_4 & -a'_3 & -a'_2 \\
a_2 & a_1 & a_0 & -a_5 & -a_4 & -a_3 & a'_2 & a'_1 & a'_0 & -a'_5 & -a'_4 & -a'_3 \\
a_3 & a_2 & a_1 & a_0 & -a_5 & -a_4 & a'_3 & a'_2 & a'_1 & a'_0 & -a'_5 & -a'_4 \\
a_4 & a_3 & a_2 & a_1 & a_0 & -a_5 & a'_4 & a'_3 & a'_2 & a'_1 & a'_0 & -a'_5 \\
a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & a'_5 & a'_4 & a'_3 & a'_2 & a'_1 & a'_0 \\
\hline
a''_0 & -a''_5 & -a''_4 & -a''_3 & -a''_2 & -a''_1 & a'''_0 & -a'''_5 & -a'''_4 & -a'''_3 & -a'''_2 & -a'''_1 \\
a''_1 & a''_0 & -a''_5 & -a''_4 & -a''_3 & -a''_2 & a'''_1 & a'''_0 & -a'''_5 & -a'''_4 & -a'''_3 & -a'''_2 \\
a''_2 & a''_1 & a''_0 & -a''_5 & -a''_4 & -a''_3 & a'''_2 & a'''_1 & a'''_0 & -a'''_5 & -a'''_4 & -a_3 \\
a''_3 & a''_2 & a''_1 & a''_0 & -a''_5 & -a''_4 & a'''_3 & a'''_2 & a'''_1 & a'''_0 & -a'''_5 & -a'''_4 \\
a''_4 & a''_3 & a''_2 & a''_1 & a''_0 & -a''_5 & a'''_4 & a'''_3 & a'''_2 & a'''_1 & a'''_0 & -a'''_5 \\
a''_5 & a''_4 & a''_3 & a''_2 & a''_1 & a''_0 & a'''_5 & a'''_4 & a'''_3 & a'''_2 & a'''_1 & a'''_0
\end{array}$$

Figure 4.3: An example of the multiplication matrix in $(\mathbb{Z}/q\mathbb{Z})^{12\times 12}$ of $\begin{bmatrix}\mathbf{a}_R & \mathbf{a}'_R \\ \mathbf{a}''_R & \mathbf{a}'''_R\end{bmatrix} \in R^{2\times 2}$ where $R = \mathbb{Z}[x]/(x^6+1)$. This type of matrices occur in rank 2 module-LWE cryptosystems with a number of Module-LWE samples of $m = 2$. Note that the entire matrix is dictated by the first and the seventh column. The columns 2 to 6 are defined as a linear transformation of the first column, whereas the columns 8 to 12 as a transformation of the seventh column.

Summary 6 (Structured vs. unstructured LWE). Among the three variants of LWE treated in this survey, RingLWE is the most structured one, and 'plain' LWE is the least structured one. ModuleLWE lies in between, and the rank parameter r' of ModuleLWE exactly indicates the similarity with plain LWE (and the dissimilarity with RingLWE): the larger r', the more ModuleLWE 'looks like' plain LWE.

4.4 NTRU

4.4.1 Introduction

The NTRU cryptosystem originates from Hoffstein, Pipher and Silverman [HPS98] and uses arithmetic in the ring[4] $R = \mathbb{Z}[x]/(x^t + 1)$ modulo two different primes p and q; where $t = 2^k$ is a power of two.

The underlying hard problem could be informally named the 'short modular fraction problem'. Given a $\mathbf{h} \in R/qR$, write it as a 'short' fraction

$$\mathbf{h} = \frac{\mathbf{g}}{\mathbf{f}} \bmod q = \mathbf{f}^{-1}\mathbf{g} \bmod q,$$

provided that it exists; where 'short' means that we want $\mathbf{g}, \mathbf{f} \in R$ to have (very) small polynomial coefficients.

4.4.2 The NTRU setup

The NTRU setup already happens in a ring R and is therefore already 'structured'. Usually, this ring is taken to be $R = \mathbb{Z}[x]/(x^t + 1)$ where $t = 2^k$ is a power of two. Such a ring can be considered as all polynomials with degree strictly less than t with coefficients in $\mathbb{Z}$, and where multiplication of two polynomials is given by ordinary polynomial multiplication followed by substituting x^t by -1 where it occurs.

Additionally, two prime numbers play a role, a *large prime number* q and a *small prime number* p. Most of the time in the NTRU setup, one calculates with the finite ring $R/qR = (\mathbb{Z}/q\mathbb{Z})[x]/(x^t+1)$, i.e., where the coefficients of the polynomials are reduced modulo q. In a small part one also needs to calculate in R/pR, so, the same type of ring but then the coefficients are reduced modulo p.

The secret key

The secret key consists of two polynomials $\mathbf{f}, \mathbf{g} \in R$ that have coefficients in $\{-1, 0, 1\}$; such polynomials are called *ternary*. Here, the polynomial $\mathbf{f}$ is required to be invertible in both R/qR and in R/pR, i.e., there exists 'inverses'

[4]In the original paper [HPS98], the ring $R = \mathbb{Z}[x]/(x^t - 1)$ is used instead; but is currently replaced by $R = \mathbb{Z}[x]/(x^t + 1)$ because $x^t - 1$ is reducible and therefore more vulnerable to 'subfield attacks'. The choice of ring is not limited to those two cases. For example, the variant NTRUPrime [Ber+18] uses a different ring $\mathbb{Z}[x]/(x^p - x - 1)$, for reasons of diminishing Galois symmetries.

$\mathbf{f}_q, \mathbf{f}_p \in R$ such that

$$\mathbf{f} \cdot \mathbf{f}_q \equiv 1 \bmod q \text{ and } \mathbf{f} \cdot \mathbf{f}_p \equiv 1 \bmod p$$

The public key

The public key is the p-multiplied 'fraction'

$$\mathbf{h} = p \cdot \mathbf{g} \cdot \mathbf{f}_q = p \cdot \frac{\mathbf{g} \bmod q}{\mathbf{f} \bmod q} \in R/qR$$

4.4.3 The NTRU assumption

For NTRU-like protocols, like with LWE, there are two main assumptions.

The search-NTRU assumption

The *search*-NTRU assumption states that it is *hard* to find elements $\mathbf{f}', \mathbf{g}' \in R$ with small coefficients such that

$$\mathbf{h} = p \cdot \frac{\mathbf{g}' \bmod q}{\mathbf{f}' \bmod q} \in R/qR.$$

Note that it is not strictly required to find the secrets $\mathbf{f}, \mathbf{g}$ themselves.

The distinguishing-NTRU assumption

Similarly to the case of LWE, the distinguishing-NTRU assumption essentially states that the fraction $\mathbf{h} = p \cdot \frac{\mathbf{g} \bmod q}{\mathbf{f} \bmod q}$ looks much like uniformly sampled elements in R/q, for an adequate distribution choice for the small-coefficient $\mathbf{f}, \mathbf{g}$. Slightly more precise: It is assumed that it is *algorithmically hard* to distinguish between a uniform sample from R/qR and an $\mathbf{h}$ constructed as the fraction of two (random) polynomials with coefficients in $\{-1, 0, 1\}$.

4.4.4 Encryption and Decryption

Encryption

A message $\mathbf{m} \in R$ is a polynomial with coefficients that are smaller than $p/2$ in absolute value. Note that the encrypter is in possession of the public key $\mathbf{h}$ (as everybody is). Encryption of $\mathbf{m} \in R$ happens by choosing a random small-coefficient polynomial $\mathbf{r} \in R$, and multiplying it with $\mathbf{h}$ and use that product to obscure the message. More precisely, the cipher text is

$$\mathbf{c} = \mathbf{r} \cdot \mathbf{h} + \mathbf{m} \pmod q,$$

where multiplication is in the polynomial ring R/qR.

Decryption

The decrypter is in possession of the secret key $\mathbf{f}$ (and its inverses modulo p and q), and can therefore compute

$$\begin{aligned}\mathbf{d} = \mathbf{f} \cdot \mathbf{c} = \mathbf{f} \cdot (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}) \pmod q \\ = \mathbf{f} \cdot \left(\mathbf{r} \cdot p \cdot \frac{\mathbf{g}}{\mathbf{f}} + \mathbf{m}\right) \pmod q \\ = p \cdot \mathbf{r} \cdot \mathbf{g} + \mathbf{f} \cdot \mathbf{m} \pmod q\end{aligned}$$

Above element $\mathbf{d}$ is only computed mod q, so, we choose a representative $\hat{\mathbf{d}}$ in $R = \mathbb{Z}[x]/(x^t + 1)$ where every coefficient lies in $\{-q/2, \ldots, 0, \ldots, q/2\}$. As $\mathbf{r}, \mathbf{g}, \mathbf{f}$ and $\mathbf{m}$ are small, we can deduce (for appropriate parameter choices) that for this representative $\hat{\mathbf{d}}$ holds

$$\hat{\mathbf{d}} = p \cdot \mathbf{r} \cdot \mathbf{g} + \mathbf{f} \cdot \mathbf{m}$$

Note the *absence of 'modulo q'*; this is an equality in the ring $R = \mathbb{Z}[x]/(x^t + 1)$. Indeed, since $\mathbf{r}, \mathbf{f}, \mathbf{g}$ and $\mathbf{m}$ are small, there is no 'overflow' over the (rather large) modulus q. Taking $\hat{\mathbf{d}}$ modulo the small prime p removes the $p \cdot \mathbf{r} \cdot \mathbf{g}$ part:

$$\hat{\mathbf{d}} = \mathbf{f} \cdot \mathbf{m} \bmod p.$$

It remains to multiply this by the p-inverse $\mathbf{f}_p$ of $\mathbf{f}$, to obtain $\mathbf{m}$ modulo p. As this message $\mathbf{m}$ was assumed to have coefficients smaller than $p/2$ in absolute value, the decrypter can successfully recover $\mathbf{m}$.

$$\mathbf{f}_p \cdot \hat{\mathbf{d}} = \mathbf{f}_p \cdot \mathbf{f} \cdot \mathbf{m} = \mathbf{m} \bmod p.$$

Idea of security

A rough idea for the proof sketch for the security of NTRU is very analogous to that of the security of the LWE-based cryptosystem. In fact, for the above encryption scheme the security is related to that of small secret RLWE. For this we assume that the message $\mathbf{m}$ is small and random (one can add an additional small error $p\mathbf{e}$ otherwise in the scheme).

Now recall the distinguishing-NTRU assumption, i.e., that it is algorithmically hard to distinguish 'small modular fractions' $\mathbf{h} = \mathbf{g}/\mathbf{f} \bmod q$ in R/qR from uniform samples $\mathbf{h}_u$ in R/qR; where 'small' means that $\mathbf{g}, \mathbf{f}$ have small coefficients. So under the distinguishing-NTRU assumption we can assume that $\mathbf{h}$ is in fact uniform in R/qR. But if we now look at the ciphertext

$$\mathbf{c} = \mathbf{r} \cdot \mathbf{h} + \mathbf{m} \mod q,$$

we see that it is in fact a small secret RLWE sample, where $\mathbf{r}$ is the small secret, $\mathbf{h}$ is uniform and public, and $\mathbf{m}$ is the secret small error. Under the (small secret) RLWE assumption it is thus hard recover the message.

4.4.5 Formal definition of NTRU

Though there are slight differences between candidates, almost all NTRU-based cryptosystems rely on the hardness of the following problem.

Definition 18 (search-NTRU [HPS98])**.** *Let t, q be positive integers and $\psi(x) \in \mathbb{Z}[x]$ be a monic irreducible polynomial of degree t and put $R = \mathbb{Z}[x]/(\psi(x))$.*

Let $\mathbf{f}, \mathbf{g} \in R$ have small coefficients (following some specific distributions $\chi_{\mathbf{f}}$ and $\chi_{\mathbf{g}}$ respectively), provided that $\mathbf{f}$ is invertible mod q. Given $\mathbf{h} = \mathbf{g}/\mathbf{f}$ mod q, the Search-NTRU is the computational problem of recovering $\mathbf{g}, \mathbf{f} \in R$.

We call $\mathbf{h}$ $(= \mathbf{g}/\mathbf{f})$ an NTRU instance with parameters $(q, n, \chi_{\mathbf{f}}, \chi_{\mathbf{g}}, \psi(x))$.

Remark 10. *The variants of NTRU mostly differ in the way the defining polynomial $\psi(x) \in \mathbb{Z}[x]$ is chosen, or the way the invertible key $\mathbf{f} \in R$ is chosen.*

For example, $\psi(x)$ is often chosen to be a cyclotomic polynomial with degree $p - 1$ or degree a power of 2 (e.g., $x^t + 1$ with t a power of 2). Alternatively the polynomials $x^t - 1$ and $x^t - x - 1$ are used.

The way $\mathbf{f} \in R$ is chosen falls roughly in a few large classes. Sometimes $\mathbf{f}$ is of the shape $p \cdot \mathbf{f}'$ or $p^{-1} \cdot \mathbf{f}'$ mod q for some $\mathbf{f}' \in R$ with small coefficients. In other cases, to make $\mathbf{f}$ trivial to invert modulo p, it is set to $\mathbf{f} = 1 + p\mathbf{f}'$ with small coefficient ring element $\mathbf{f}' \in R$.

In the concluding sections, we will not consider all these variants, for lack of space.

Summary 7 (NTRU assumption). The NTRU assumption states that, for adequate parameters $\psi(x) \in \mathbb{Z}[x]$, $q \in \mathbb{N}_{>0}$, fractions of polynomials $\mathbf{h} = \mathbf{g}/\mathbf{f}$ modulo q and $\psi(x)$ are computationally indistinguishable from uniform polynomials modulo q and $\psi(x)$.

Remark 11. *A recent work of Felderhoff, Pellet-Mary and Stehlé [PS21; FPS22] shows that NTRU is as least as hard as unique Module-SVP in rank 2 (by showing a reduction from the latter to the former). Additionally, it is shown that NTRU can be reduced to RingLWE.*

NTRU and RingLWE are both at least as hard as IdealSVP, as there is a reduction from IdealSVP to those two problems.

4.5 Variants of NTRU

Though none of the NIST candidates actually implements this, there exists also a unstructured variant of NTRU. This could be seen as an NTRU analogue of plain LWE, in which no ring is involved.

We will treat this variant because it will be useful for translating the original (ring) NTRU into a lattice problem.

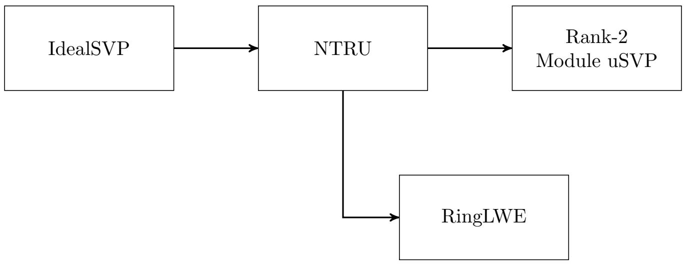


Figure 4.4: A simplified diagram of the reductions between IdealSVP, NTRU, RingLWE and unique module SVP.

4.5.1 'Unstructured' NTRU

In unstructured NTRU, the large prime q and a small prime p essentially play the same role as in the structured variant. The ring elements $\mathbf{h}, \mathbf{g}, \mathbf{f}$ are instead replaced by square matrices $\boldsymbol{H}, \boldsymbol{F}, \boldsymbol{G} \in \mathbb{Z}^{n\times n}$, that satisfy

$$\boldsymbol{H} = p \cdot \boldsymbol{F}^{-1} \cdot \boldsymbol{G} \bmod q,$$

where $\boldsymbol{F}, \boldsymbol{G}$ have *very small entries* and where $\boldsymbol{F}$ is assumed to be invertible mod q. Additionally, $\boldsymbol{F}$ is also assumed to be invertible mod p.

Encryption and decryption

Encryption and decryption happens in the same fashion; a message $\mathbf{m} \in \mathbb{Z}^n$ with entries bounded in absolute value by $p/2$ is encrypted by putting

$$\mathbf{c} = \boldsymbol{H}\mathbf{r} + \mathbf{m} \bmod q$$

for a small random $\mathbf{r} \in \mathbb{Z}^n$. Such a ciphertext is decrypted by multiplying $\mathbf{c}$ on the right with $\boldsymbol{F}$ to obtain

$$\mathbf{d} = \boldsymbol{F}\mathbf{c} = \boldsymbol{F}(\boldsymbol{H}\mathbf{r} + \mathbf{m}) = p\boldsymbol{G}\mathbf{r} + \boldsymbol{F} \cdot \mathbf{m} \bmod q$$

Taking a representative $\hat{\mathbf{d}} \in \mathbb{Z}^n$ and taking it modulo p, yields (due to the smallness of all elements)

$$\hat{\mathbf{d}} = \boldsymbol{F} \cdot \mathbf{m}.$$

By multiplying with $\boldsymbol{F}^{-1}$ modulo p, one obtains the original message again.

Original NTRU as a 'unstructured' NTRU problem

Most NTRU-based cryptosystems already happen in a ring setting, with $R = \mathbb{Z}[x]/(\psi(x))$. In order to make an NTRU-instance into an 'unstructured' NTRU instance, we need to transform the ring-based equality (with $\mathbf{f}, \mathbf{g}, \mathbf{h} \in R$)

$$\mathbf{h} = \mathbf{f}^{-1} \cdot \mathbf{g} \bmod q$$

into one of matrices (with $\boldsymbol{H}, \boldsymbol{F}, \boldsymbol{G} \in \mathbb{Z}^{n\times n}$)

$$\boldsymbol{H} = \boldsymbol{F}^{-1} \cdot \boldsymbol{G} \bmod q.$$

Exactly as with LWE, this transformation is done by seeing the additive part of the ring $R = \mathbb{Z}[x]/(\psi(x))$ as a $\mathbb{Z}$-module: $R \simeq \mathbb{Z}^t$ (taking $n = t$). Here, the basis elements can be chosen to be the monomials, $1, x, x^2, \ldots$, as in Section 4.3.2. Multiplication by $\mathbf{f}, \mathbf{g}$ or $\mathbf{h}$,

$$R \xrightarrow{\cdot \mathbf{f}} R, \quad \mathbf{x} \mapsto \mathbf{x} \cdot \mathbf{f}$$

can then be seen as an *invertible linear map*, described by a matrix, which are then respectively coined $\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}$.

As in the Ring-LWE case, these matrices will then have a specific shape, depending on the structure of the ring R. For example, for the ring $R = \mathbb{Z}[x]/(x^t + 1)$, it will be of the anti-circulant form, as in Figure 4.2.

Summary 8 (NTRU as a lattice problem). Like with structured LWE, original NTRU translates into a lattice problem by considering the multiplication matrix of the public key $\mathbf{h}$.

Part II

Unstructured attacks and refinements

Chapter 5

Attacks on Unstructured Lattices

In this chapter we discuss attacks on (variants of) LWE and NTRU assumptions by interpreting them as generic lattice problems. The security estimates for all lattice-based NIST candidates is mainly based on these attacks. For usual parameters, the best known attacks are based on BKZ and SVP calls in lower dimensions. In this chapter we will mostly consider the block-size β of BKZ or the dimension β of these SVP calls, as the main cost metric. In the next chapter we will go deeper into the concrete costs of running BKZ or solving SVP with such parameters.

5.1 LWE and NTRU as lattice problems

5.1.1 Introduction

In Sections 4.3 and 4.5 we saw that all variants of LWE and NTRU can be phrased as a 'plain' LWE or a 'plain' NTRU problem, involving matrices, vectors and plain lattices in $\mathbb{R}^n$. As this chapter is about unstructured attacks, we will only consider these plain variants of LWE and NTRU, leaving the structured attacks on the respective structured variants to the later Chapter 7.

5.1.2 Search-LWE as a lattice problem

Recall from Definition 15 the search-LWE problem: given $(\boldsymbol{A}, \mathbf{b}) \in (\mathbb{Z}/q\mathbb{Z})^{m\times n} \times (\mathbb{Z}/q\mathbb{Z})^m$ satisfying, for some secret $\mathbf{s} \in (\mathbb{Z}/q\mathbb{Z})^n$,

$$\mathbf{b} = \boldsymbol{A} \cdot \mathbf{s} + \mathbf{e} \bmod q \tag{5.1}$$

where $\mathbf{e} \leftarrow \chi$, some 0-centered distribution over $(\mathbb{Z}/q\mathbb{Z})^m$. The task is to find the secret $\mathbf{s}$ from this information.

LWE as a BDD instance

One could consider the lattice

$$\mathcal{L}_q(\boldsymbol{A}) = \{\mathbf{y} \in \mathbb{Z}^m \mid \mathbf{y} = \boldsymbol{A}\mathbf{x} \bmod q \text{ for some } \mathbf{x} \in \mathbb{Z}^n\} = \boldsymbol{A}\mathbb{Z}^n + q\mathbb{Z}^m.$$

If the small error vector $\mathbf{e}$ in Equation (5.1) did not exist, the vector $\mathbf{b}$ would be in this lattice $\mathcal{L}_q(\boldsymbol{A})$. But, we know, because of this small error $\mathbf{e}$, that the vector $\mathbf{b}$ does *not* lie in this lattice, although it must be very *close* to the lattice! Namely, due to the modular equation there exists some $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{b} = \boldsymbol{A} \cdot \mathbf{s} + q\mathbf{z} + \mathbf{e}$. Since $\boldsymbol{A} \cdot \mathbf{s} + q\mathbf{z} \in \mathcal{L}_q(\boldsymbol{A})$, the distance of $\mathbf{b}$ to the lattice $\mathcal{L}_q(\boldsymbol{A})$ is bounded by $\|\mathbf{e}\|$.

$$\operatorname{dist}(\mathbf{b}, \mathcal{L}_q(\boldsymbol{A})) := \min_{\ell \in \mathcal{L}_q(\boldsymbol{A})} \|\mathbf{b} - \ell\| \leq \|\mathbf{b} - \boldsymbol{A} \cdot \mathbf{s} - q\mathbf{z}\| = \|\mathbf{e}\|.$$

Though, the error vector $\mathbf{e}$ is known to be concentrated around $0 \in (\mathbb{Z}/q\mathbb{Z})^n$, thus one may assume that there is a known, small bound on $\|\mathbf{e}\|$. So, almost surely, $\boldsymbol{A} \cdot \mathbf{s} + q\mathbf{z}$ *is the closest vector to* $\mathbf{b}$. As retrieving $\boldsymbol{A} \cdot \mathbf{s} + q\mathbf{z}$ allows to recover $\mathbf{s}$ (by just modular inversion of the matrix $\boldsymbol{A}$), search-LWE is an instance of the *closest vector problem* (CVP). Even more so, because there is a given upper bound on the closeness (namely $\|\mathbf{e}\|$), this problem is even a *Bounded Distance Decoding* (BDD) instance.

So one can see search-LWE with parameters n, m, q, χ as a Bounded Distance Decoding instance with target $\mathbf{b} \in (\mathbb{Z}/q\mathbb{Z})^m$ and lattice $\mathcal{L}_q(\boldsymbol{A})$, with closeness promise $\operatorname{dist}(\mathbf{b}, \mathcal{L}_q(\boldsymbol{A})) \leq \|\mathbf{e}\| \leq u_\chi$ (with high probability). Here, $u_\chi \in \mathbb{R}_{>0}$ is a high-probability upper bound on $\|\mathbf{e}\|$ for $\mathbf{e} \leftarrow \chi$.

The lattice $\mathcal{L}_q(\boldsymbol{A})$ has dimension m and (with high probability) determinant q^{m-n}. The Gaussian heuristic indicates that the first minimum is about $\operatorname{gh}(\mathcal{L}_q(\boldsymbol{A})) \approx \sqrt{m/(2\pi e)} \cdot q^{\frac{m-n}{m}}$. For an error distribution χ with independent 0 centered coefficients with some constant standard deviation $\sigma > 0$, we expect an error of size about $\sqrt{m}\sigma$. The BDD distance ratio δ then becomes

$$\delta \approx \frac{\sqrt{m} \cdot \sigma}{\sqrt{m/(2\pi e)} \cdot q^{\frac{m-n}{m}}} = \sqrt{2\pi e}\sigma \cdot q^{\frac{n-m}{m}}.$$

Solving a BDD instance can be done by a so-called *dual attack*, which tries to find short *dual lattice vectors* in the *dual lattice* of $\mathcal{L}_q(\boldsymbol{A})$, in order to solve BDD in the original (primal) lattice. We discuss this attack in Section 5.2.2.

LWE as a uSVP instance

Alternatively, one can also see the search-LWE problem as an instance of an unusually-short SVP (uSVP) instance, as every BDD-instance can be translated into such a uSVP instance (where the dimension is increased by one).

This translation can be done by creating a new matrix, constructed by stacking the column vector $\mathbf{b}$ next to $\boldsymbol{A}$ and adding an extra dimension to keep the columns linearly independent.

$$\begin{bmatrix} \boldsymbol{A} & \mathbf{b} \\ \mathbf{0} & c \end{bmatrix}$$

Here, $\mathbf{0}$ is a n-dimensional row vector consisting of zeros, and $c \in \mathbb{R}_{>0}$ is chosen in an appropriate way (namely, close to $\|\mathbf{e}\|$ for a proven reduction, or just a small constant in practice). Then the following lattice,

$$\mathcal{L}_q(\boldsymbol{A}, \mathbf{b}) := \left\{ \mathbf{y} \in \mathbb{Z}^{m+1} \;\middle|\; \mathbf{y} = \begin{bmatrix} \boldsymbol{A} & \mathbf{b} \\ \mathbf{0} & c \end{bmatrix} \cdot \mathbf{x} \bmod q \text{ for some } \mathbf{x} \in \mathbb{Z}^{n+1} \right\}$$

of dimension $m+1$ has the unusually short vector

$$\begin{bmatrix} \boldsymbol{A} & \mathbf{b} \\ \mathbf{0} & c \end{bmatrix} \cdot \begin{bmatrix} \mathbf{s} \\ -1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}\mathbf{s} - \mathbf{b} \\ -c \end{bmatrix} = \begin{bmatrix} \mathbf{e} \\ -c \end{bmatrix}$$

Finding such a unusually short vector in a lattice is done by a so-called *primal attack*, in which short vectors in the original (called ‘primal’, as opposed to ‘dual’) lattice solve the problem at hand. This is the subject of Section 5.2.1.

5.1.3 Search-NTRU as a lattice problem

Recall from Section 4.5, the (unstructured) search-NTRU problem: given $\boldsymbol{H} \in (\mathbb{Z}/q\mathbb{Z})^{n \times n}$, find any $\boldsymbol{F}, \boldsymbol{G} \in (\mathbb{Z}/q\mathbb{Z})^{n \times n}$ with small coefficients such that

$$\boldsymbol{H} = \boldsymbol{G} \cdot \boldsymbol{F}^{-1} \bmod q, \tag{5.2}$$

given that such $\boldsymbol{F}, \boldsymbol{G}$ exist.

NTRU as a uSVP instance

One can phrase an NTRU instance as a lattice problem in the following way, by putting the public $n \times n$ matrix $\boldsymbol{H}$ into a $2n \times 2n$ matrix in the following block shape.

$$\boldsymbol{B_H} = \begin{bmatrix} q\boldsymbol{I} & \boldsymbol{H} \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix}, \tag{5.3}$$

where $\boldsymbol{I}$ is the $n \times n$ identity matrix. The lattice $\mathcal{L}(\boldsymbol{B_H})$ generated by the columns of $\boldsymbol{B_H}$ then contains any column of the $2n \times n$ matrix

$$\boldsymbol{B_{GF}} = \begin{bmatrix} \boldsymbol{G} \\ \boldsymbol{F} \end{bmatrix}. \tag{5.4}$$

This can be deduced from Equation (5.2), which implies that $\boldsymbol{H} \cdot \boldsymbol{F} = \boldsymbol{G} + q\boldsymbol{W}$ for some $\boldsymbol{W} \in \mathbb{Z}^{n \times n}$. Therefore, using block-wise multiplication, we obtain

$$\boldsymbol{B_H} \cdot \begin{bmatrix} -\boldsymbol{W} \\ \boldsymbol{F} \end{bmatrix} = \begin{bmatrix} q\boldsymbol{I} & \boldsymbol{H} \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix} \cdot \begin{bmatrix} -\boldsymbol{W} \\ \boldsymbol{F} \end{bmatrix} = \begin{bmatrix} -q\boldsymbol{W} + \boldsymbol{H}\boldsymbol{F} \\ \boldsymbol{F} \end{bmatrix} = \begin{bmatrix} \boldsymbol{G} \\ \boldsymbol{F} \end{bmatrix} \tag{5.5}$$

Concluding, we see that the lattice $\mathcal{L}(\boldsymbol{B_H})$ contains n unusually short vectors consisting of the columns of $\boldsymbol{B_{GF}}$. So, one can see a search-NTRU problem as

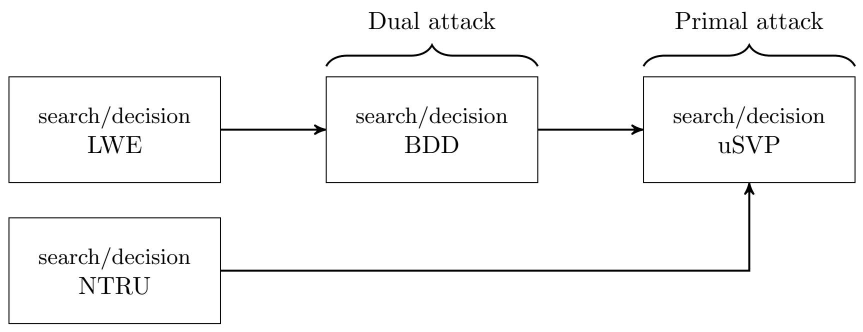


Figure 5.1: Overview of reductions from LWE and NTRU to the general lattice problems BDD or uSVP.

an instance of a uSVP problem in $\mathcal{L}(\boldsymbol{B_H})$ where the (unusual) shortness of the vectors is parametrized by the distribution of the matrices $\boldsymbol{G}$ and $\boldsymbol{F}$.

The dimension of the NTRU lattice $\mathcal{L}(\boldsymbol{B_H})$ is $2n$ and the determinant is q^n. The Gaussian heuristic indicates a first minimum of $\mathrm{gh}(\mathcal{L}(\boldsymbol{B_H})) = \sqrt{n/\pi e} \cdot \sqrt{q}$, but the lattice contains unusually short secret vectors (the columns of $\boldsymbol{B_{GF}}$). Suppose $\boldsymbol{G}$ and $\boldsymbol{F}$ have independent 0 centered coefficients with standard deviation σ, then the n secret columns have norm about $\sqrt{2n}\sigma$. The uSVP gap δ between the actual minimum and the expected minimum is

$$\delta = \frac{\sqrt{2n}\sigma}{\sqrt{n/\pi e} \cdot \sqrt{q}} = \frac{\sqrt{2\pi e}\sigma}{\sqrt{q}}.$$

One can roughly compare NTRU to LWE with $m = 2n$ samples. The presence of not 1 but n unusual shortest vectors makes the NTRU problem in general slightly easier than LWE for similar parameters. For large moduli however, the presence of these n vectors can make the NTRU problem significantly easier than a similar LWE instance, this regime is discussed in Section 5.2.4.

Summary 9 (LWE and NTRU as lattice problems). The LWE problem can be interpreted as a Bounded Distance Decoding problem in a q-ary lattice of dimension m. Alternatively, LWE (and any BDD instance) can be transformed into a uSVP instance of dimension $m + 1$. The NTRU problem can directly be interpreted as a uSVP problem in dimension $2n$, where the unusually short vectors are given by the columns of the secret key $\boldsymbol{B_{GF}}$.

5.1.4 Influence of parameters on hardness

The hardness of a BDD or uSVP instance generally increases with the dimension and the error norm or the (unusually small) first minimum. The latter is often summarized as the ratio δ between the error or vector length and the Gaussian heuristic of the lattice. In the previous parts we discussed the relationship between the LWE or NTRU parameters and this ratio. In Table 5.1

we summarized the influence of increasing LWE or NTRU parameters on the hardness of the underlying problem.

Parameter	Dimension d	BDD/uSVP ratio δ	Influence
$\uparrow q$	$= d$	$\downarrow \delta$	Easier
$\uparrow n$	$= d$	$\uparrow \delta$	Harder
$\uparrow \sigma$ (or $\uparrow \mathbb{E}(\|\chi\|)$)	$= d$	$\uparrow \delta$	Harder (if $\delta < 1$).
$\uparrow m$ (for LWE)	$\uparrow d$	$\downarrow \delta$	Same or easier[1]

Table 5.1: Influence of increase of LWE and NTRU parameters on hardness of the associated BDD or uSVP problem. Decreasing the same parameters has the adverse effect.

Summary 10 (Influence of parameters on hardness). Generally, increasing the parameter n or the error/secret size makes the LWE and NTRU problems harder to solve. Increasing the modulus q makes the problem easier. For LWE, increasing the number of samples can make the problem easier.

5.2 Kind of attacks

5.2.1 Primal Attack

As the name suggests, the *primal attack* reduces the basis of the (primal) lattice. The goal is to recover an unusually short vector of the lattice. This unusually short vector is either inherently part of the lattice, or the result of embedding a BDD instance. The premise is that the lattice $\mathcal{L}$ contains a shortest vector $\mathbf{v}$ of length $\lambda_1(\mathcal{L}) \ll \mathrm{gh}(\mathcal{L})$, i.e., much shorter than expected. Note that, in general, to find a shortest vector, one needs to solve SVP in the full dimension d. However, when this vector is unusually short the BKZ algorithm already recovers it for a blocksize $\beta \ll d$. The larger the gap between $\lambda_1(\mathcal{L})$ and $\mathrm{gh}(\mathcal{L})$, the easier it is to recover a shortest vector.

Estimates.

Initially the focus of the primal attack was on the unique-SVP problem, where there is a large gap between $\lambda_1(\mathcal{L})$ and $\lambda_2(\mathcal{L})$. That the BKZ algorithm recovers such a unique shortest vector much earlier than expected was already observed experimentally in [GN08]. By extrapolating the experimental results the authors came to the following rough estimate: BKZ-β recovers a unique shortest vector with high probability whenever $\lambda_2(\mathcal{L})/\lambda_1(\mathcal{L}) \geq \tau_\beta \cdot \alpha_\beta^{d/2}$, where $\tau_\beta < 1$ is a experimentally determined constant that depends on the algorithm, lattice

[1]One can always ignore some samples and therefore assume a lower $m' < m$. For certain parameters, the best attacks do not use all samples.

family, and blocksize used. This estimate [GN08] is often referred to as '*the* 2008 *estimate*'.

Later, in [AFG13], the 2008 estimate was applied to (embedded) LWE instances. Experiment with small blocksizes where showed to match the 2008 estimate with a constant τ_β between 0.3 and 0.4. While the 2008 estimate gives quite reasonable predictions, it does not explain *how* a unique shortest vector is recovered, and therefore also lacks a heuristic explanation.

The first heuristically motivated estimate came in 2016 and became known as the *2016 estimate* [Alk+16]. It turns out that the uniqueness does not really matter, it is more about the gap between $\lambda_1(\mathcal{L})$ and $\mathrm{gh}(\mathcal{L})$. The general idea is that if $\mathbf{v}$ is unusually short, then so is the projection $\pi_{d-\beta+1}(\mathbf{v})$ to the last BKZ block $\mathcal{L}_{[d-\beta+1:d]}$. In case the length $\|\pi_{d-\beta+1}(\mathbf{v})\|$ is less than the expected minimum $\mathrm{gh}(\mathcal{L}_{[d-\beta+1:d]})$, it is likely that the SVP call in this last BKZ block will recover the projection $\pi_{d-\beta+1}(\mathbf{v})$. For large enough β, it is generally easy to then recover the full vector $\mathbf{v}$ from its projection. This reasoning leads to the following estimate.

Estimate 1 (Primal Attack)**.** *Let $\mathcal{L}$ be a lattice of dimension d and let $\mathbf{v} \in \mathcal{L}$ be a unusually short vector $\|\mathbf{v}\| \ll \mathrm{gh}(\mathcal{L})$. Then under the Geometric Series Assumption BKZ recovers $\mathbf{v}$ if*

$$\sqrt{\beta/d} \cdot \|\mathbf{v}\| < \sqrt{\alpha_\beta}^{2\beta-d-1} \cdot \mathrm{Vol}(\mathcal{L})^{1/d},$$

where $\alpha_\beta = \mathrm{gh}(\beta)^{2/(\beta-1)}$.

Justification. The left hand side of the inequality is an estimate for $\|\pi_{d-\beta+1}(\mathbf{v})\|$, while the right hand size is the expected norm of $\mathbf{b}^*_{d-\beta+1}$ under the GSA. When the inequality is satisfied we expect that the shortest vector in $\mathcal{L}_{[d-\beta+1:d]}$ is in fact (a projection of) the unusually shortest vector, and thus it is inserted by BKZ at position $d-\beta+1$. See Figure 5.2 for an illustration.

Except for very small blocksizes β, the unusually short vector $\mathbf{v}$ is recovered from its projection $\pi_{d-\beta}(\mathbf{v})$ with high probability, either directly by Babai's nearest plane algorithm, or by later BKZ tours on the blocks $\mathcal{L}_{[d-2\beta+2:d-\beta+1]}$, $\mathcal{L}_{[d-3\beta+3:d-2\beta+2]}, \ldots$; lifting the vector block by block. The 2016 estimate has been verified extensively [Dac+20; PV21] with experiments in feasible dimensions. In hindsight this implies that the 2008 estimate, in case $\lambda_2(\mathcal{L}) \approx \mathrm{gh}(\mathcal{L})$, is true for a constant $\tau_\beta \approx \mathrm{gh}(\beta)^{-1}$.

Asymptotics.

Consider a lattice $\mathcal{L}$ of dimension d and with gap $\mathrm{gh}(\mathcal{L})/\lambda_1(\mathcal{L}) = d^{g+o(1)}$ for some constant $g \geq 0$. Let $\beta = B \cdot d$ for some constant B. Estimate 1 asymptotically boils down to the inequality

$$B > \frac{1}{2g+1} + o(1).$$

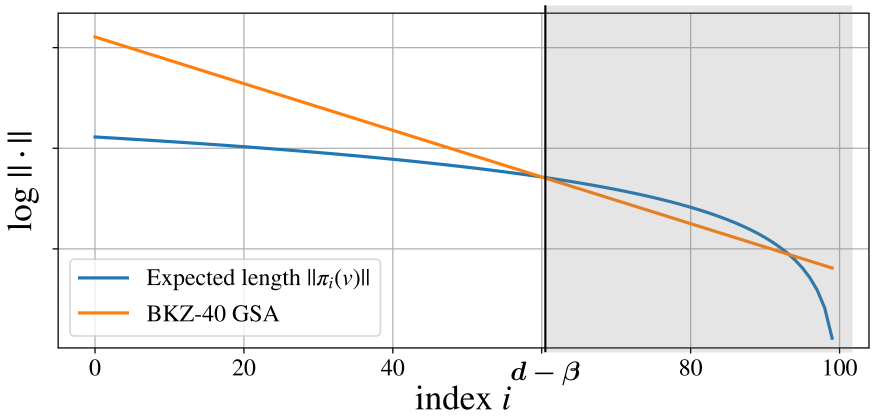


Figure 5.2: Illustration of the 2016 Estimate on a 100-dimensional lattice with an unusually short vector $\mathbf{v}$. Around $\beta = 40$ the projection $\pi_{100-\beta+1}(\mathbf{v})$ is expected to be the shortest vector in the projected sublattice $\mathcal{L}_{[d-\beta+1:d]}$, which is thus recovered by the SVP call on this block inside BKZ-β.

For example, with a gap of $\mathrm{gh}(\mathcal{L})/\lambda_1(\mathcal{L}) = \sqrt{d}$ we obtain $B = \frac{1}{2}$ and thus the primal attack only requires a blocksize of about $\frac{1}{2}d + o(d)$ to recover an unusually short vector. This is the typical regime for current cryptographic schemes, such as Falcon and Dilithium.

Refinements.

The 2016 Estimate gives a clear explanation on how and where the secret vector is recovered, namely its projection is found in the last BKZ block. This also allows to further refine the estimate and give concrete predictions. For example by using a BKZ-simulator instead of the GSA, and by accounting for the probability that after the projection $\|\pi_{d-\beta}(\mathbf{v})\|$ has been found, it is successfully lifted to the full vector $\mathbf{v}$. Also instead of working with the expected length of the projection, we can directly model the probability distribution under the assumption that $\mathbf{v}$ is distributed as a Gaussian vector. Such refinements were applied in [Dac+20; PV21], and the resulting concrete predictions match with experiments to recover an unusually short vector. Sometimes there is not just a single unusually short vector, but there are more of them, which makes it more likely that at least one of them is recovered. Because the refined concrete estimators already works with a probability distribution, we can easily take multiple vectors into account.

Summary 11 (Primal attack for uSVP). The primal attack solves the uSVP instance by running the BKZ lattice reduction algorithm on a (primal) lattice basis. Given an estimate for the length of the unusually short vector(s), there exist accurate (probabilistic) models and estimates for the BKZ blocksize that is required to recover such a vector.

5.2.2 Dual Attack

Dual attack for decision BDD.

The dual attack is a general way to distinguish BDD instances. So far it has mostly been considered and optimized for the LWE problem [MR09; Alk+16; Alb17; EJK20; GJ21; MAT22]. We will first consider a more natural and general version.

Every lattice $\mathcal{L}$ has a dual lattice $\mathcal{L}^* = \{\mathbf{w} \in \operatorname{span}(\mathcal{L}) : \langle \mathbf{v}, \mathbf{w} \rangle \in \mathbb{Z} \text{ for all } \mathbf{v} \in \mathcal{L}\}$. For a BDD instance $\mathbf{t} = \mathbf{v} + \mathbf{e}$ with $\mathbf{v} \in \mathcal{L}$, and a dual vector $\mathbf{w} \in \mathcal{L}^*$ we have

$$\langle \mathbf{t}, \mathbf{w} \rangle = \langle \mathbf{v}, \mathbf{w} \rangle + \langle \mathbf{e}, \mathbf{w} \rangle \equiv \langle \mathbf{e}, \mathbf{w} \rangle \bmod 1.$$

If $\mathbf{t}$ is uniform (over $\mathbb{R}^d/\mathcal{L}$), then $\langle \mathbf{t}, \mathbf{w} \rangle \bmod 1$ is uniform over $[-\frac{1}{2}, \frac{1}{2})$. However, if $\mathbf{e}$ and $\mathbf{w}$ are both small, then $\langle \mathbf{e}, \mathbf{w} \rangle$ is small and $\langle \mathbf{t}, \mathbf{w} \rangle \bmod 1$ becomes biased towards 0. This difference can be used to distinguish a uniform target from a BDD instance.

Summary 12 (Dual attack for BDD). For a BDD instance $(\mathcal{L}, \mathbf{t})$, the modular inner product $\langle \mathbf{t}, \mathbf{w} \rangle \bmod 1 \in [-\frac{1}{2}, \frac{1}{2})$ with a short dual vector $\mathbf{w} \in \mathcal{L}^*$ is biased towards 0. For a uniform target the outcome would also be uniform. The dual attack solves the decisional BDD problem computing the above inner product for many short dual vectors, and seeing if the result is uniform or not.

Lemma 2. *Let $\mathcal{L}$ be a lattice of dimension d, and let $\mathbf{d} \in \mathcal{L}^*$ be a dual vector of length $\|\mathbf{d}\| = \alpha \cdot \operatorname{gh}(\mathcal{L}^*)$. Let $\mathbf{e}$ be a uniform error of length $\delta \cdot \operatorname{gh}(\mathcal{L})$. The the advantage of distinguishing $\langle \mathbf{e}, \mathbf{w} \rangle \bmod 1$ from random is*

$$\epsilon = \exp\left(-\frac{\alpha^2 \cdot \delta^2 \cdot d}{2e^2}\right).$$

If $O(1/\epsilon^2)$ of such independent samples are available, the success probability for the distinguisher is close to 1. Generally, there is no reason for these samples to behave independently, but one can heuristically assume that they do to obtain a conservative estimate.

Estimate 2. *Let $\mathcal{L}$ be a lattice of dimension d, and let $D \subset \mathcal{L}^*$ be a finite list of distinct dual vectors of length at most $\alpha \cdot \mathrm{gh}(\mathcal{L}^*)$. If we assume the samples behave independently, then the dual attack correctly decides a δ-dBDD instance $(\mathcal{L}, \mathbf{t})$ with high probability if*

$$\#D \cdot \epsilon^2 \geq \Omega(1),$$

where $\epsilon = \exp\left(-\frac{\alpha^2 \cdot \delta^2 \cdot d}{2e^2}\right)$.

Remark 12 (Independence Heuristic)**.** *Since the work of [MAT22] the independence heuristic has been under sharp scrutiny and has been shown to lead to incorrect estimates [DP23b; WEB23]. In particular, these works partly invalidate the reduced security level claims of [MAT22]. These issues can partly be resolved but they do lead to worse estimates in certain regimes [DP23a].*

One could obtain such short dual vectors by first BKZ reducing the dual lattice with blocksize β, after which we obtain $\#D = (4/3)^{\beta/2+o(\beta)}$ dual vectors of length about $\sqrt{4/3} \cdot \sqrt{\alpha_\beta}^{d-1} \cdot \det(\mathcal{L}^*)^{1/d}$. This leads to the following estimate.

Estimate 3 (Dual Attack)**.** *Let $\mathcal{L}$ be a lattice of dimension d. Then under the Geometric Series Assumption and the independence assumption the dual attack with sieving correctly decides a δ-dBDD instance with high probability if*

$$\beta/2 \cdot \ln(4/3) \geq \frac{8\pi}{3e} \cdot \alpha_\beta^{d-1} \cdot \delta^2,$$

where $\alpha_\beta = \mathrm{gh}(\beta)^{2/(\beta-1)}$.

Justification. The left hand side represents the number of short dual vectors $\log(\#D) = \beta/2 \cdot \ln(4/3) + o(\beta)$ obtained from a sieving call on a β-dimensional sublattice. After BKZ reduction with blocksize β on the dual lattice, we obtain a basis $[\mathbf{d}_1, \ldots, \mathbf{d}_d]$, where $\|\mathbf{d}_1\| = \sqrt{\alpha_\beta}^{d-1} \cdot \det(\mathcal{L}^*)^{1/d}$. Given that $\|\mathbf{d}_1\| \approx \mathrm{gh}([\mathbf{d}_1, \ldots, \mathbf{d}_\beta])$ a full sieve on the sublattice generated by $\mathbf{d}_1, \ldots, \mathbf{d}_\beta$ gives vectors of length bounded by $\ell^* = \sqrt{4/3} \cdot \sqrt{\alpha_\beta}^{d-1} \cdot \det(\mathcal{L}^*)^{1/d}$, and thus

$$\alpha^2 := \frac{\ell^*}{\mathrm{gh}(\mathcal{L}^*)} = \frac{4}{3} \cdot \frac{\alpha_\beta^{d-1}}{d/2\pi e}.$$

We then obtain

$$\ln(1/\epsilon^2) = \frac{\alpha^2 \cdot \delta^2 \cdot d}{e^2} = \frac{8\pi}{3e} \cdot \alpha_\beta^{d-1} \cdot \delta^2,$$

which forms the right-hand side of the estimate.

While the distinguishing advantage of a dual vector can be some $\epsilon > 0$, and we need about $O(1/\epsilon^2)$ of such vectors, we still need a concrete distinguishing algorithm that manages to extract this advantage into a correct decision. We know that such an algorithm should be periodic with respect to the primal lattice $\mathcal{L}$. For a list $D = \{\mathbf{w}_1, \ldots, \mathbf{w}_K\} \subset \mathcal{L}^*$ of dual vectors, one naturally obtains the periodic function

$$F_D(\mathbf{t}) := \sum_{j=1}^{K} \exp(2\pi i \cdot \langle \mathbf{w}_j, \mathbf{t} \rangle).$$

If the inner products $\langle \mathbf{w}_j, \mathbf{t} \rangle$ mod 1 are uniformly distributed, then $F_D(\mathbf{t}) \approx 0$, whereas if $|\langle \mathbf{w}_j, \mathbf{t} \rangle|$ mod 1 is biased towards 0 then (the real part of) $F_D(\mathbf{t})$ is expected to be large. By choosing an appropriate threshold value T we obtain a decision algorithm. If all dual vectors are of about the same length, then the above strategy is close to optimal (see e.g. [GMW21]). If the dual vectors are not of the same length, one can introduce some weights in the summation that depends on their norm, giving more importance to shorter vectors.

Asymptotics.

Consider a lattice $\mathcal{L}$ of dimension d and a δ-dBDD instance with $\delta = d^{-g+o(1)} \cdot \mathrm{gh}(\mathcal{L})$ for some constant $g \geq 0$. Let $\beta = B \cdot d$ for some constant B. Estimate 3 asymptotically boils down to the inequality

$$B > \frac{1}{2g+1} + o(1).$$

For example, with $g = \sqrt{d}$ we obtain $B = \frac{1}{2}$ and thus the dual attack only requires a blocksize of about $\frac{1}{2}d + o(d)$. Note that asymptotically the dual attack coincides perfectly with the primal attack (after converting the BDD instance to an unusual SVP instance).

The cost of the basic dual attack is mainly dominated by the cost of obtaining the short dual vectors. The best sieving algorithms run in time $(3/2)^{d/2}$, while the distinguishing part only requires on the order of $\#D = (4/3)^{d/2} \ll (3/2)^{d/2}$ computations. We can improve this balance by creating multiple BDD instances on the same lattice, such that we can reuse the short dual vectors.

BDD search to decision

The same idea can also be used to solve the BDD instance (contrary to just distinguishing it). In particular if $\langle \mathbf{e}, \mathbf{w} \rangle \in [-\frac{1}{2}, \frac{1}{2})$, then we obtain a non-modular linear equation on the error $\mathbf{e}$. Computing d of such independent equations allow to fully recover the error. However, this would require a few very short dual vectors, while for the decision problem it is optimal to use many somewhat short vectors. To resolve this we refer to the next section on hybrid attacks, that naturally convert a (partial) search BDD instance into multiple decision BDD instance.

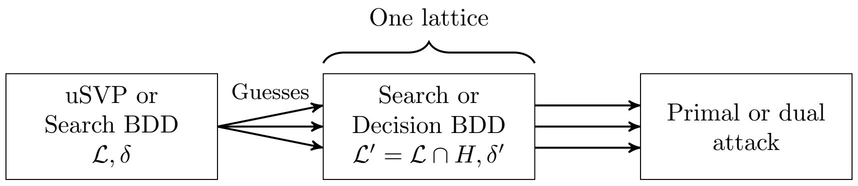
5.2.3 Hybrid Attack

Let $\mathcal{L}$ be a lattice and $\mathbf{w} \in \mathcal{L}^*$ be a primitive dual vector. Let $(\mathcal{L}, \mathbf{t} = \mathbf{v} + \mathbf{e})$ be a BDD instance, and suppose that we know that $\langle \mathbf{e}, \mathbf{w} \rangle \in S$ for some finite set

$S \subset \mathbb{R}$. Suppose $\langle \mathbf{e}, \mathbf{w} \rangle = c$, then we know that

$$\langle \mathbf{w}, \mathbf{v} \rangle = \langle \mathbf{w}, \mathbf{t} \rangle - \langle \mathbf{e}, \mathbf{w} \rangle = c',$$

and the right-hand side is known. This describes some hyperplane H in which the solution $\mathbf{v}$ lies, e.g. $\mathbf{v} \in \mathcal{L} \cap H$. Let $\mathbf{x} \in \mathcal{L} \cap H$, then $\mathcal{L}' := (\mathcal{L} \cap H) - \mathbf{x}$ is again a lattice, of dimension $d - 1$ and determinant $\det(\mathcal{L}') = \|\mathbf{w}\| \cdot \det(\mathcal{L})$. Furthermore the original BDD instance can be adapted to $(\mathcal{L}', \pi_H(\mathbf{t}) - \mathbf{x})$ with solution $\mathbf{v} - \mathbf{x}$. So by solving the new BDD instance in $\mathcal{L}'$ we obtain a solution to the original BDD instance.



One lattice
uSVP or Search BDD $\mathcal{L}, \delta$
Guesses
Search or Decision BDD $\mathcal{L}' = \mathcal{L} \cap H, \delta'$
Primal or dual attack

Figure 5.3: Diagram of hybrid attack. Here $\mathcal{L}'$ is some lower rank section of the original lattice.

In general we do not know such a linear equation $\langle \mathbf{e}, \mathbf{w} \rangle = c$ on our secret error, however it might be the case that there are only a few possible values for the inner product, or some that are highly likely. For example for a LWE lattice, which is integer, we always have the trivial unit dual vectors $\mathbf{u}_i = (0, \ldots, 1, \ldots, 0)$. In case of ternary errors the inner product $\langle \mathbf{e}, \mathbf{u}_i \rangle = e_i \in \{-1, 0, 1\}$ has only a few possibilities. Making these three guesses means we go from one BDD instance on $\mathcal{L}$, to three BDD instances of $\mathcal{L}'$. The corresponding new lattice $\mathcal{L}'$ has the same determinant as the original lattice, a lower dimension, and depending on the error distribution the absolute BDD radius can decrease. As a result the BDD instances on $\mathcal{L}'$ are relatively easier, and depending on the parameters solving the three BDD instances might be easier than solving the single original one. In fact one is only required to solve the decision BDD problem for the resulting instances, which indicates if the original guess was correct or not.

The process might be applied recursively, reducing the dimension to $d-k$ for some $k > 0$. At the same time the number of guesses often grows exponentially in k, so there is a careful trade-off to be made. One should also take note that when trying to identify a single BDD instance out of X guesses, that the needed distinguishing advantage must increase by a small factor $\sqrt{\ln(X)}$.

Summary 13 (Hybrid attack). A hybrid attack combines a primal or dual attack with some initial guessing phase. The hybrid attack is mostly effective if the secret (error) is sparse, or more generally, has small entropy. For example a hybrid attack might be effective in the case of binary or ternary secrets, or for very narrow discrete Gaussians.

FFT speed-up for hybrid dual attack

The hybrid attack reduces a single search BDD instance to many decisional BDD instances in the same lattice. This implies that if we instantiate the decision oracle by the dual attack, we only have to pre-compute the short dual vectors once. In contrast, for the primal attack, we would need, at the very least, to solve a large SVP instance, for *each* decisional BDD instance. The hybrid attack is thus mostly useful in combination with the dual attack.

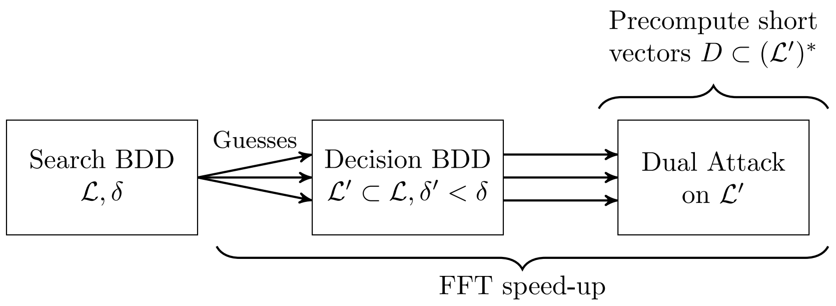

Precompute short vectors $D \subset (\mathcal{L}')^*$
Search BDD $\mathcal{L}, \delta$
Guesses
Decision BDD $\mathcal{L}' \subset \mathcal{L}, \delta' < \delta$
Dual Attack on $\mathcal{L}'$
FFT speed-up

Figure 5.4: Overview of full hybrid dual attack from search BDD.

Furthermore, it is possible to attain another speed-up when using the hybrid dual attack, by FFT techniques. Algorithmically, when instantiating the decision oracle by the dual attack, we have to compute (the real part) of

$$F_D(\mathbf{t} - \mathbf{g}) := \sum_{j=1}^{K} \exp(2\pi i \cdot \langle \mathbf{w}_j, \mathbf{t} - \mathbf{g} \rangle).$$

for each guess $\mathbf{g}$. If we have T guesses, and K dual vectors, then this would require about $O(T \cdot K)$ inner product computations. Given that usually both T and K are (exponentially) large, this can be a limiting factor for the hybrid dual attack.

If these guesses have some particular structure however, we can speed up this computation to about $\tilde{O}(\max T, K)$ inner products. Note that every guess $\mathbf{g}$ corresponds to some coset $\mathbf{g} + \mathcal{L}'$ w.r.t. the lower rank lattice $\mathcal{L}'$. Now suppose the set G of guesses is closed under addition modulo $\mathcal{L}'$, i.e. the guessed cosets form an additive group, then FFT techniques are applicable.

To see this let $\mathcal{L}'' = G + \mathcal{L}'$ be the superlattice of finite index. The idea is that in that case the value of the inner product $\langle \mathbf{w}_j, \mathbf{g} \rangle$ only depends on the coset of $\mathbf{w}_j \in (\mathcal{L}')^*/(\mathcal{L}'')^* = (\mathcal{L}''/\mathcal{L}')^* \cong \hat{G}$. We can rewrite

$$F_D(\mathbf{t} - \mathbf{g}) := \sum_{\hat{\mathbf{w}} \in \hat{G}}^{K} f(\hat{\mathbf{w}}) \exp(-2\pi i \cdot \langle \hat{\mathbf{w}}, \mathbf{g} \rangle),$$

where

$$f(\hat{\mathbf{w}}) := \sum_{\mathbf{w}_j \in D_{|\hat{\mathbf{w}}}} \exp(2\pi i \cdot \langle \mathbf{w}_j, \mathbf{t} \rangle).$$

Computing $(f(\hat{\mathbf{w}})_{\hat{\mathbf{w}} \in \hat{G}}$ has a total cost of $|D| = K$ inner products. From these values we can then compute $(F_D(\mathbf{t} - \mathbf{g}))_{\mathbf{g} \in G}$ by an FFT in time $O(|G| \log(|G|)$.

In theory, the FFT can reduce the running time by a square root. Unfortunately, if we guess only the small errors, the cosets do not form an additive group. When applying such an attack to e.g. LWE this could be resolved by guessing all possible errors modulo q, resulting in an FFT with cost in the order of $\tilde{O}(q^t)$ when guessing t coordinates. Given that q is generally quite large, the cost of this grows extremely fast, limiting t to small values. Recent works [GJ21; MAT22] have resolved this limitation. Either by only guessing the error coefficients modulo some smaller value γ (say $\gamma = 2$), or, similarly, by modulus switching techniques to translate the problem to some lower modulus γ. The FFT cost is then reduced to only $\tilde{O}(\gamma^t)$ when (partially) guessing t coordinates, allowing for much larger guessing dimensions t. Both techniques do introduce some additional round-off errors, which can be compensated by more short dual vectors. Overall, these techniques can improve the hybrid dual attack by half a dozen of bits.

Summary 14 (Hybrid dual attack). When we use the dual attack as a distinguisher, we can pre-compute the short dual vectors once, and an FFT can be used to speed-up the handling of all BDD instances at once.

5.2.4 Dense Sublattice Attack

The primal attack strongly relies on the existence of an unusually short vector in the lattice. However, on certain lattices, the primal attack can behave better than expected. It turns out that this is the case if the lattice contains a large rank dense sublattice, i.e., with covolume *and* if a somewhat good basis is already known.

In particular this applies to the NTRU lattice. The $n = d/2$ secret keys (f, g) generate a sublattice of rank n. Due to their shortness this sublattice is dense compared to the full lattice. Moreover, the q-vectors give a somewhat good basis. When the modulus q is relatively large, BKZ reduction recovers the dense sublattice and thereby the secret keys[2], with a much lower blocksize than what follows from the primal attack estimate. This regime is coined the *overstretched regime*.

Initially, attacks in the overstretched regime had a more algebraic nature [ABD16; CJL16]. This broke several NTRU-based FHE schemes. In a later work Kirchner and Fouque pointed out that the efficiency of the algebraic attack was in fact routed in the geometry of the NTRU lattice itself, and that the regular

[2]Once the secret dense sublattice is recovered, the dimension of the lattice problem is halved and the secret key recovery often follows immediately. Moreover, for regular NTRU the dense sublattice is an ideal lattice in which quantum algorithms can recover the secret short vectors in polynomial time (for certain parameters).

primal attack should perform better than expected. In fact, this dense sublattice attack is better than the original algebraic ones. Recently, the concrete complexity of the dense dual attack was thoroughly studied [DW21], leading to a precise concrete estimator for the blocksize that BKZ needs to recover the dense sublattice, extensively verified by experiments. This work showed that the dense sublattice attack is already applicable for reasonably low moduli. I.e., for uniform ternary secrets and NTRU parameter n, the cross-over is around $q = 0.004 \cdot n^{2.484}$. Asymptotically, this is above the usual NTRU parameters for encryption and signature schemes (for which $q = O(n)$), but due to the low constant one still needs to take care when choosing concrete parameters.

Estimate 4. *For an NTRU instance with parameters q, n and secrets of norm $\|\mathbf{f}\|, \|\mathbf{g}\| = O(\sqrt{n})$, the dense sublattice attack improves upon the primal attack when*

$$q > n^{2.484+o(1)}.$$

For ternary secret coefficients the precise cross-over point is around $0.004 \cdot n^{2.484}$.

Remark 13. *In theory one can also construct a lattice with a dense sublattice from structured LWE instances. I.e. by adding not a single target, but all (rotations of) the target, one can again obtain a dense rank n sublattice. However in these cases the dimension of the full lattice also grows significantly, and therefore the dense sublattice attack does not seem to improve upon the original primal attack for LWE.*

Summary 15 (Dense sublattice attack for NTRU). The NTRU lattice contains a dense sublattice of large rank. For large moduli q this leads to an attack that is asymptotically better than the primal and dual attacks. For typical small error distributions this happens when $q \geq n^{2.484+o(1)}$. Concretely, however, this might already occur for relatively small q, and thus the relevancy of this attack should be checked on a case by case basis.

5.2.5 Arora-Ge and BKW

We shortly discuss here some attacks that are generally not applicable. They generally require an error distribution with a very small support, or a lot of samples. Note that for the LWE-based NIST candidates, if executed properly, we obtain at most $m \leq 2n$ LWE samples.

The Arora-Ge attack [AG11] turns the LWE problem into a system of polynomial equations, which they solve in turn by standard linearization techniques. The general idea is that if an error coefficient e_i takes at most k values $\{c_1, \ldots, c_k\}$, then $\prod_{j=1}^{k}(e_i - c_j) = 0$. Now let $f_i(X) := f_i(X_1, \ldots, X_n) = b_i - \sum_{j=1}^{n} \mathbf{A}_{ij} \cdot X_j$ and note that $f_i(s_1, \ldots, s_n) = e_i$. So, each from each public LWE sample we can construct the multivariate polynomial $F_i(X) := \prod_{j=1}^{k}(f_i(X) - c_j)$ of degree k and with root $\mathbf{s}$. Doing this for all samples gives a polynomial system of degree k in n variables, m equations and solution $\mathbf{s}$. Such a system contains

$O(n^k)$ distinct monomials, and can thus be solved by linearization techniques if $m \geq O(n^k)$ equations (samples) are available.

For binary LWE, i.e. for which the errors take values in $\{0, 1\}$ the Arora-Ge attack runs in polynomial time if $m \geq \Theta(n^2)$ samples are available. However, given that there are at most $2n$ errors available such an attack does not apply, even in the extreme case of binary errors (both asymptotically and concretely).

Another option is to solve the polynomial system of equations using Gröbner basis techniques [Alb+14]. This allows for a trade-off between the number of available equations (samples) and the run-time. Again looking at the binary error case this leads to a sub-exponential attack whenever $m = \Theta(n \log \log n)$. Moreover, such an attack improves on e.g. the primal attack only when $m \geq 7.98n$ samples are available. For larger error sets the number of required samples quickly grows.

The BKW attack was initially developed for the LPN problem (similar to LWE, but with $q = 2$). Consider the LWE secret $s = (s_1|s_2)$ where $s_1 \in (\mathbb{Z}/q\mathbb{Z})^k$ and $s_2 \in (\mathbb{Z}/q\mathbb{Z})^{n-k}$. The idea is to find many pairs of LWE samples $(b_i, a_i), (b_j, a_j)$ such that say the first k coefficients of $a_i - a_j$ are 0. For such a pair we obtain a new LWE sample $(b_i - b_j, a_i - a_j)$. Because the first k coefficients of $a_i - a_j$ are 0, we can remove those to obtain an LWE sample for the secret s_2 of dimension $n - k$. We have thus reduced the dimension of the LWE problem. We can repeat this process until the LWE problem becomes feasible. Such an attack has two problems, firstly the error of the new LWE sample is larger than the original one, and secondly, we need about $m = \Theta(\sqrt{q^k})$ samples to find even a single pair that collides on the first k coefficients. To use this technique recursively we would constantly need about $m = \Theta(q^k)$ samples. The parameters k cannot be too small as otherwise the error would increase too much. All in all, for typical parameters, the BKW algorithm requires and exponential number of samples to function. With only $m = 2n$ available samples the BKW attack is not applicable.

Summary 16 (Arora-Ge, Gröbner bases and BKW). The Arora-Ge and Gröbner bases attacks are only applicable when many LWE samples are available and when the error support is very small. Only for binary errors the required number of samples is close to the $2n$ available ones (but still larger). The BKW attack requires significantly more samples.

5.3 BKZ Block-sizes β for NIST Candidates

We give an example for the computation of the BKZ blocksize needed to solve an LWE instance. We focus on the primal attack as it is the simplest, and does not involve other costs beyond BKZ reduction. For most schemes the primal attack also gives a good rough estimate of the security. For more refined estimates and for the other attacks we recommend the usage of estimation scripts which we will discuss more in Section 6.4.

Let us consider an LWE instance $(\boldsymbol{A}, \mathbf{b} = \boldsymbol{A} \cdot \mathbf{s} + \mathbf{e} \bmod q)$ with parameters (q, n, m, χ), where χ samples each coefficient as a discrete Gaussian with standard deviation σ. Following Section 5.1 we turn the LWE instance with m samples into a uSVP instance with the lattice spanned by $\mathcal{L}_q(\boldsymbol{A}) \oplus \mathbf{0}$ and the embedded BDD instance $(\mathbf{b}, 1)$. This lattice has dimension $d = m + 1$, volume q^{m-n} and contains (for common parameters) an unusually short vector $(\mathbf{e}, 1)$. The expected norm of the shortest vector is close to $\sqrt{m\sigma^2 + 1}$. Following Estimate 1 running BKZ on this lattice recovers the secret short vector $(\mathbf{e}, 1)$ when the blocksize β satisfies

$$\sqrt{\beta/(m+1)} \cdot \sqrt{m\sigma^2 + 1} < \sqrt{\alpha_\beta}^{2\beta - m - 2} \cdot q^{(m-n)/(m+1)},$$

where $\alpha_\beta = \mathrm{gh}(\beta)^{2/(\beta-1)}$.

For example for concrete parameters $n = 640, m = 2 \cdot 640, q = 2^{15}$ and $\sigma = 2.8$ we obtain the above whenever $\beta \geq 485.72$. This, however, might not be the best primal attack. Using less samples might be beneficial, so one has to find the lowest blocksize β along with some $m \leq 2 \cdot 640$, that satisfies the above inequality. For this example reducing the number of samples increases the blocksize β, and thus it is best to use all the samples.

Chapter 6

The fine-grained costs of BKZ and SVP

Cost estimates for cryptographic attacks generally have three traits that distantiate them from their real-life costs. The first trait is that they mostly consider asymptotic run-times, which generally omits multiplicative constants that are very relevant non-asymptotically. The second trait is that these cost estimates generally necessarily involve simplifications regarding the computations. For example, one can assume that memory usage is 'free' or that an attack can be totally parallelized. In reality, working on real machines, these simplifications are far from true. Thirdly, the behavior of the attack algorithms is usually simplified and analyzed using heuristics, and these heuristics can differ from the actual more complex behavior.

A relevant example of these differences between theoretically estimated costs and real-life costs is that of the comparison between sieving and enumeration in lattices in order to find short vectors. While asymptotically, sieving has a lower overall cost than enumeration, enumeration algorithms were for a long time more efficient in practice. Only quite recently, sieving has beaten enumeration in real-life costs, as can be seen by the recent solutions to the Darmstadt challenges by the lattice sieving framework G6K [SG10; Alb+19].

In this chapter we will discuss both the asymptotic, as well as the lower order improvements to the enumeration and sieving algorithms. Overall, these improvements, over the last decade, have lead to significant decreases to the concrete cost of the attacks discussed in Chapter 5. We will end with a discussion on how to compute concrete cost estimates for solving LWE and NTRU instances.

6.1 Enumeration

Recall that the enumeration algorithms find short (or close) vectors in a lattice $\mathcal{L}$ recursively by first enumerating short (or close) vectors in the projected lattices $\pi_i(\mathcal{L})$. The complexity of this enumeration process heavily depends on the reduction quality of the basis. Assuming the current basis is close to HKZ reduced, the time complexity of the standard enumeration algorithm is asymptotically given by $2^{\frac{\beta \log \beta}{2e} + O(\beta)}$ for a rank β lattice. This is roughly the case when using enumeration within progressive BKZ, and also in practice this leading term is achieved.

Super-exponential speed-ups

The time complexity of enumeration within BKZ was recently improved from $2^{\frac{\beta \log \beta}{2e} + O(\beta)}$ to $2^{\frac{\beta \log \beta}{8} + O(\beta)}$ [Alb+20b]. This potential improvement was already indicated in [HS10], and given as a heuristic lower bound in [Ngu09]. This improved constant was therefore already used in some pessimistic security estimates.

The idea is that enumeration is much faster on bases that follow the Geometric Series Assumption (see Heuristic 2), i.e. that have a straight log-profile. In theory, this would lead to a leading constant of $\frac{1}{8}$. However, for the reduction quality that is needed, the basis attains a more concave HKZ shape (see Definition 13). The concavity is what makes the constant increase to $\frac{1}{2e}$. For the general SVP problem there seems to be no way to circumvent this problem.

However, when using enumeration as an SVP oracle within BKZ, there is more freedom to pre-reduce the lattice basis. The trick in [Alb+20b] is to pre-reduce a slightly larger block around the BKZ block in which the enumeration is ran. Because the reduction is relatively weaker to the larger pre-processing dimension, the basis does actually attain the GSA shape in BKZ block. Some care has to be taken at the first and last few BKZ blocks, but this can be resolved. This strategy heuristically works as long as the BKZ blocksize β is sufficiently small compared to the total dimension d. In practice however it seems sufficient to have $d \geq 1.15\beta$, which makes it applicable for all NIST candidates for which usually $\beta \leq d/2 + o(d)$.

Exponential speed-ups

When looking for a vector $\mathbf{v} \in \mathcal{L}$ of length at most $R > 0$, the naive enumeration algorithm enumerates all vectors of at most length R in the projected lattices $\pi_i(\mathcal{L})$. However, assuming $\mathbf{v}$ is pointed in a random direction relative to the basis, one can expect that $\|\pi_i(\mathbf{v})\|^2 \approx \frac{n-i}{n} \cdot \|\mathbf{v}\|^2 \leq \frac{n-i}{n} \cdot R^2$. It would therefore be sufficient to enumerate all vectors of squared length at most $R_i^2 := \frac{n-i}{n} \cdot R^2 + \epsilon$ in the projected lattice $\pi_i(\mathcal{L})$, while still finding a short vector with high probability. For example, $R_{\lfloor n/2 \rceil} \approx R/\sqrt{2}$, which following the Gaussian Heuristic would result in a factor $\sqrt{2}^{n/2} = 2^{n/4}$ less vectors to enumerate in $\pi_{\lfloor n/2 \rceil}(\mathcal{L})$, giving an exponential speed-up.

We call this pruned enumeration and the radii $R = R_0 \geq R_1 \geq \ldots \geq R_n$ the pruning parameters [SH95]. By adjusting the pruning parameters there is a trade-off between the size of the enumeration tree, and the success probability of finding a short vector. It turns out that the best trade-off is achieved by decreasing the pruning parameters such that the success probability is exponentially small, but such that the search tree decreases even more [GNR10]. This regime is called extreme pruning. By re-randomizing and repeating the enumeration (an exponential number of times), the success probability can be boosted to find a short vector with high probability. This also leads to another large exponential speed-up over naive pruning. Concretely, the extreme enumeration cost when used inside BKZ with blocksize β was determined to be around $2^{0.184\beta\log_2(\beta)-0.995\beta+16.25}$.

Another line of works tried to make the enumeration inside BKZ more efficient by allowing it to find not exactly the shortest vector, but some approximate short vector. The loss of shortness is then compensated by running the enumeration in a slightly larger BKZ block. For reasonably small approximation factors $\alpha > 1$ the trade-off seems beneficial [Aon+16; Agg+20; LN20a]. In a later work [Alb+21] the approximate strategy was combined with the earlier discussed super-exponential speed-up by extended pre-processing. Concretely, this lead to the most efficient enumeration procedure within BKZ known so far, with a time complexity of $2^{0.125\beta\log(\beta)-0.654\beta+25.84}$ for BKZ with blocksize β.

Estimate 5. *The cost of BKZ reducing an d-dimensional lattice with blocksize β and enumeration as a subroutine can be done in time*

$$\mathit{poly}(d)\cdot 2^{0.125\beta\log(\beta)-0.654\beta+25.84}$$

The main benefit of enumeration techniques over sieving techniques is their almost trivial parallelization and the polynomial memory complexity. This makes it relatively easy to run large scale attacks on large clusters. The main drawback of course is the time complexity that is much worse than sieving algorithms. The advanced sieving techniques that we will discuss next are already faster than the enumeration techniques in dimensions as low as 70, and the gap quickly grows after that. This makes enumeration techniques currently not so relevant for security estimate against classical attacks.

The quantum side of things is a slightly different story. Enumeration techniques can in general benefit from a quadratic speed-up by applying Grover search (contrary to lattice sieving techniques). Lowering the concrete complexity to $2^{0.0625\beta\log(\beta)+O(\beta)}$. For low blocksizes this might improve a bit upon the fastest sieving algorithms. However, given the expected overhead of quantum computations over classical ones, and the inherent overhead of Grover search, it is unclear if quantum enumeration improves significantly over sieving in cryptographic dimensions (around security level I).

Summary 17 (SVP by enumeration). Enumeration algorithms solve the shortest vector problem in dimension β in super-exponential time $\exp O(\beta\log(\beta))$, and polynomial memory. They are easy to parallelize, but for dimensions relevant to cryptography their run-time is inferior to that of sieving algorithms.

6.2 Sieving

We will now discuss improvements to the lattice sieving algorithms. These are currently the most efficient algorithms to solve SVP, both asymptotically as in practice. Recall from Section 2.12 that heuristic sieving algorithms take a list of $N = (4/3)^{d/2+o(d)}$ vectors, and repeatedly tries to find pairs of close vectors $\mathbf{x}, \mathbf{y}$ in this list. The short(er) vector $\mathbf{x} - \mathbf{y}$ is then inserted back into the list, possibly replacing a longer one. This process is repeated until the list contains a short enough vector. The naive algorithm by Nguyen and Vidick [NV08], which checks all pairs in the list runs in time $N^{2+o(1)} = 2^{0.415d+o(d)}$.

Exponential speed-ups.

To reduce the time complexity below $2^{0.415d+o(d)}$ we need a more efficient way to find pairs of close vectors in the list. In a line of works [Laa15; BGJ15; BL16; Bec+16] the time complexity was improved to $2^{0.292d+o(d)}$ by nearest neighbor searching techniques. Instead of checking all pairs, the idea is to first apply some bucketing technique in which close vectors are more likely to fall into the same bucket. By only considering the somewhat-close pairs inside each bucket, the total number of checked pairs can be decreased.

We shortly highlight two of such bucketing variants, the practically efficient `bgj1` sieve with a time complexity of $2^{0.349d+o(d)}$ [BGJ15], and the asymptotically best `bdgl` sieve with a time complexity of $2^{0.292d+o(d)}$ [Bec+16]. The `bgj1` sieve creates buckets by first picking some random directions $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathcal{S}^{n-1} = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| = 1\}$. Then a vector $\mathbf{w}$ from the list is put in a bucket $\mathbf{v}_i$ if $\langle \mathbf{v}_i, \mathbf{w}/\|\mathbf{w}\| \rangle \geq \alpha$ for some fixed constant $\alpha > 0$. The geometric interpretation of this is that the vector $\mathbf{w}$ lies in the spherical cone that points in the direction of $\mathbf{v}_i$. Two vectors that lie inside the same spherical cone have a higher probability to be close that two uniformly random vectors. By tweaking the number of buckets and the constant α the trade-off between the bucketing and then finding the pairs inside each bucket can be optimized. In this case leading to buckets of size about $O(\sqrt{N})$, and a time complexity of $2^{0.349n+o(n)}$.

The `bdgl` sieve improves the bucketing part by using structured buckets. In particular the dimension d is split unto multiple parts $d = d_1 + \ldots + d_k$, and the total set of bucket is a product of random local buckets. By sorting and cleverly enumerating the inner product of a vector $\mathbf{w}$ with the (few) local buckets on can quickly enumerate all buckets that the vector $\mathbf{w}$ should be part of. In addition, for say $k = O(\log(d))$ the bucket directions are still close to uniformly distributed (more about this later). Due to this structure one can have many more buckets, while keeping the bucketing cost low. This leads to a different trade-off, with much smaller optimal bucket sizes of around $N^{1/(k+1)}$, and for slowly growing k a time complexity of $2^{0.292d+o(d)}$.

To lower the space cost below $N = 2^{0.2075+o(d)}$ one has to look beyond sums and differences of pairs, and consider triples $\mathbf{u} \pm \mathbf{v} \pm \mathbf{w}$ or more generally k-tuples [BLS16; HK17; HKL18]. The current best triple sieve [HKL18] has a space complexity of $2^{0.1887d+o(d)}$, but comes with a higher time complexity of

$2^{0.3588d+o(d)}$. Given that current estimates mostly ignore the cost of memory, we do not consider these memory saving variants further.

Quantumly, there is good news from the perspective of security. The speed-up obtained from Grover search techniques applied to sieving algorithms has so far only given minor improvements, far from the possible quadratic speed-up. A line of works [LMV15; Laa16a; CL21; Hei21] improved the classical time complexity of $2^{0.292d+o(d)}$ down to $2^{0.2570d+o(d)}$. Even though the improvement is far from quadratic, it might still be of importance for the security of schemes. However, even in unit cost quantum memory models, the concrete speed-up of quantum sieving algorithms is tenuous at best [Alb+20a]. With all the expected overhead of quantum computations, quantum sieving is so far not a threat. Note that all sieving algorithms based on pairs have a fundamental lower bound of $2^{0.2075d+o(d)}$ on the number of vectors that need to be stored. As a result, any quantum improvements to the search procedures will also be limited to that lower bound.

Summary 18 (SVP by sieving). The best sieving algorithm solves the shortest vector problem in dimension β in time $2^{0.292\beta+o(\beta)}$, and $2^{0.2075\beta+o(\beta)}$ memory. Quantum computers only improve the run-time to $2^{0.2570\beta+o(\beta)}$, while keeping the same exponential (quantum) memory usage.

Subexponential speed-ups.

Asymptotically the single exponential time complexity of sieving algorithms is much lower than the super-exponential time complexity of enumeration algorithms. However, naive sieving implementations can have large subexponential overheads. Enumerations algorithms are therefore faster in low dimensions, and for a long time sieving algorithms were not viewed as practical. This status-quo changed due to many subexponential and polynomial improvements, and currently sieving algorithms are already faster than enumeration in dimensions as low as 70.

One improvement, already discussed in Section 2.12, is progressive sieving. Instead of immediately sieving in the full dimension, we first sieve in a projected sublattice, and slowly undo the projection (while sieving) until we are sieving in the full dimension. Due to the exponential time complexity sieving in the lower dimensional projected sublattices is often much more efficient than in the full dimension. Once we reach the full dimension the vectors are already quite short, which implies that we only need to do a bit of sieving there.

Another large improvement, which works well in conjunction with progressive sieving, is the 'dimensions for free' technique [Duc18]. The key observation is that lattice sieving algorithms do not just find a single shortest vector, but a large list of short vectors. Heuristically, the list contains (a large fraction of) the N shortest vectors of the lattice. Now suppose we sieve in some projected sublattice $\pi_f(\mathcal{L})$ for some not too large $f > 1$. Suppose $\mathbf{v} \in \mathcal{L}$ is a shortest vector in the full lattice. If f is not too large, and if the basis is already quite a bit reduced, then we can expect that $\pi_f(\mathbf{v})$ is present in the sieving list. Thus by lifting all vectors obtained from the sieving process to the full lattice we expected to recover a shortest vector of the full lattice. A heuristic analysis shows that this can be expected if $f \leq \frac{\ln(4/3)d}{\ln(d/2\pi e)}$. We can thus solve exact SVP in dimension d by sieving only in dimension $d-f$, essentially getting f dimensions for free. Given that f is sublinear this leads to a subexponential speed-up. For BKZ for schemes of security level 1 this can reduce the sieving dimension by more than 40 dimensions, making it an important technique to take account of when making estimates.

Lastly, we discuss some small polynomial (or even constant) speed-up. Lattice sieving involves a lot of floating point inner product computations. The latest record computations (and their analysis) showed that a low floating point precision, such as 16-bit, is more than enough for the most computing-intensive task. Given that the gate footprint of such multiplications often grows like p^2 for a precision of p bits, this leads to some important constant speed-ups. This also enables usage on extremely efficient hardware such as Tensor cores on NVIDIA GPUs [DSW21], or even more efficient dedicated hardware.

An even more extreme technique is the so-called popcount hash trick. Here the idea is to create a small hash of each vector that captures some information about its direction. One approach is for example to take some direction vectors $\mathbf{h}_1, \ldots, \mathbf{h}_h$, and to record if the inner product $\langle \mathbf{h}_i, \mathbf{v} \rangle$ is positive or negative. This information can be pre-computed and stored as a bit string of length h (say $h = 2n$). Now if two vectors $\mathbf{v}, \mathbf{w}$ are close, then one can expected that their corresponding bit strings share many directions. By a simple binary XOR operations combined with a population count (counting the number of ones), we can thus quickly compute if two vectors are close or not. Commonly this is used as an initial filter, and only for those pairs passing the filter a full inner product is computed. In practice this can leads to a factor $10\times$ speed-up.

Estimates

The heuristic nature, and the many involved vectors, makes estimating the cost of sieving algorithms very complex. The so-called *core SVP* estimate ignores all subexponential factors and just computes the main complexity term. E.g. for the `bdgl` in dimension β this would result in a bit estimate of 0.292β. However, such estimates are overly optimistic, given that the overheads in terms of gates can be extremely large. The more precise estimates have tried to estimate the precise cost of the most computationally intensive tasks (bucketing and finding pairs), sometimes by explicitly setting up circuits [Alb+18; Alb+20a]. Still some overheads are ignored, but these estimates are getting closer to the truth. The most precise concrete estimate from [Alb+20a] gives a gate complexity summarized as $2^{0.356d+o(d)+23.08}$ for `bgj1` and $2^{0.299d+26.01}$ for `bdgl`, for dimensions up to about $\beta \leq 1024$. The estimate for `bdgl` has been updated after [MAT22] found a mistake in the estimation scripts of [Alb+20a]. These estimates are for a single sieving iteration in dimension β. This is somewhat realistic in the case of progressive sieving given that a single iteration (or less) might already be enough in each dimension. In that case we should also add the (constant factor) progressive overhead of sieving in the lower dimensions.

Ignored overheads

These estimates, certainly the one for `bdgl`, should be viewed as optimistic (or pessimistic from a cryptanalysis point of view). For example, the negative overhead of using structured buckets that are not uniformly random can be as large as subexponential. This overhead is completely ignored in the above estimate. In a recent work a more precise estimate was made that does include these overheads [Duc22]. Around security level 1 this leads to a run-time increase by a factor of about 2^6. This factor can in theory be decreased at the cost of a significant memory increase.

Another reason why these estimates might be optimistic is because the usage of memory is ignored (or counted at unit cost). The best attacks around security level 1 require at least 2^{95} memory. Achieving such a memory capacity might be much harder than achieving the say 2^{140} needed bit operations. Secondly, random access would require at least $\log_2(2^{95}) = 95$ traversal gates (and in practice much more), adding another overhead that is currently ignored. Physically, the memory would also be spread out, leading to latency and bandwidth constraints. Memory bandwidth was already a problem in current record computations on a single machine when using GPUs, and is expected to only increase when moving to a multi node setup.

Possible improvements

We shortly discuss a few areas where we might expect improvements to the current best lattice sieving algorithms.

At the moment there are no signs for more exponential speed-ups. The nearest neighbor line of improvements has culminated in the $2^{0.292d+o(d)}$ algorithm, and from a nearest neighbor perspective this is optimal [KL21]. Still there might be improvements in the lower order terms. For example, the 2^6 overhead from the structure in the `bdgl` sieve is not inherent, but originates from the choice of using a product of locally random buckets. Currently, this is the best approach known, but progress on the area of spherical decodable codes might remove the overhead. Note that this overhead was already ignored in the latest estimates.

Moving to sub-exponential speed-ups the dimensions for free technique might be stretched a bit further. Already in practice we attain more dimensions for free than expected, by lifting many vectors, that are encountered during sieving, on the fly. It is unclear how these practical observations scale to cryptographic dimensions. Another interesting trade-off might be to combine dimensions for free with the approximate BKZ techniques recently used in enumeration. Allowing a small SVP approximation factor instead of aiming for exact SVP can increase the dimensions for free significantly. Although this does not give exponential speed-ups as in the enumeration case, it might still lead to a significant sub-exponential speed-ups. Lastly, the current dimensions for free techniques are strongly related to the primal attack. In case it turns out that the dual attack is actually better, it might also be possible to increase the number of dimensions for free by using techniques from there. Possibly combined with hybrid methods.

There are still some simplifications made in the current estimates that are slightly pessimistic. For example, all vectors in the sieving list are assumed to be of about the same norm, i.e., they are assumed to be uniform over some sphere. Secondly, all vectors in some spherical bucket are also assumed to be uniform over the edge of this bucket. While asymptotically these assumptions are fine, they do not hold up when looking at lower order terms. It is easier to find a reduction between a short and a long vector than it is to find one between two long vectors. This is not as simple as replacing uniformity of spheres to that of balls, as sieving algorithms do not output vectors that are uniform over a ball, in fact they are slightly biased to vectors of small norm.

The current estimates compute the cost of finding (almost) all close pairs in a list of vectors. Which is the main operation in lattice sieving. If we actually require to find *all* pairs depends on multiple factors such as the saturation we want to achieve, the number of vectors in our list, and the length (and exact distribution) of them at the start of sieving. When sieving directly in the full dimension we might have to repeat this operation many times. Conversely, for progressive sieving, and not too harsh saturation constraints, it might be enough to find a fraction of all pairs, thus obtaining a speed-up over the estimates (at the cost of the progressive overhead).

Summary 19 (Concrete estimates for sieving). Precisely estimating the lower order factors in the time complexity of lattice sieving algorithms is still in its infancy. Further improvements could push current estimates a bit lower, while incorporating realistic overheads, such as for memory, could increase the estimates significantly. Generally, the obtained security estimates can be seen as conservative.

Estimates versus reality

We conclude this section on lattice sieving with a comparison of the current estimates and real world computations. In particular, we compare the gate-count estimates for sieving with the concrete records in Table 6.1.

For the estimates we used the latest (corrected) estimates from [Alb+20a; MAT22], and we added the progressive sieving overhead factor of $C = 1/(1 - 2^{-0.292}) \approx 5.46$, which accounts for sieving in the dimensions $2, \ldots, d$. For the records we used the reported floating point operations (FLOP) from [DSW21]. These estimates only counted the main operation of computing inner products between vectors in 16-bit precision. To obtain a gate-count we multiplied by 16^2, which is the estimate for a naive 16-bit multiplication circuit. Note that these records did not use the popcount trick, as for GPUs it was overall better to have a stronger filter. This should be seen as a rough comparison, to see if the estimates are anywhere close to the results obtained in practice.

We take a look at the comparison in Table 6.1. Firstly, we see that the

	Record runs		Gate estimates	
Sieving dimension	Cost (2^k FLOP)	Cost (2^k gates)	BGJ1	BDGL
143	67.5	75.5	76.4	65.2
146	68.6	76.6	77.5	66.1
150	69.9	77.9	78.9	67.3

Table 6.1: Comparing TU Darmstadt record sieving runs from [DSW21], with the sieving cost estimates. A FLOP represents a 16-bit floating point operation (mostly addition or multiplication). The estimate only counts the operations from the reduction phase and is thus an underestimate. The gate cost is an optimistic extrapolation by adding $\log_2(16^2)$ (naive cost of 16-bit multiplication), ignoring any gates needed for memory or other operations. The estimates correspond to the most precise estimates for the BGJ1 sieve (`random_buckets-classical`), and BDGL sieve (`list_decoding-classical`) from [MAT22; Alb+20a]. The last estimate is currently used for security estimates.

estimates for the `bgj1` sieve match very closely with the concrete record computations. Indeed, the sieve used in these computations was asymptotically equivalent to the `bgj1` sieve. The same work also implemented a CPU and GPU version of the `bdgl` sieve. For CPUs the `bdgl` sieve already improved upon the `bgj1` sieve in dimensions as low as 90, as expected by the estimates. However, when moving to GPUs, the trade-off seems to be in dimensions above 150. This is due to memory bandwidth constraints, for which the `bdgl` sieve is much more susceptible than the `bgj1` sieve. At the same time the estimates for the `bdgl` sieve are significantly lower, giving an improvement of more than 10 bits around a sieving dimension of 150. In practice, the `bgj1` sieve is faster. This gives an indication that the lower order overheads of the `bdgl` sieve are much higher in practice than what the current estimates give. This can partially be attributed to the structural overhead indicated by [Duc22], and partially by memory bandwidth constraints. This indicates that scaling the `bdgl` sieve to a multi-node system is expected to result in significant overheads.

Summary 20 (Lattice sieving in cryptographic dimensions). Scaling lattice sieving algorithms to cryptographic dimensions is non-trivial. In particular, the current asymptotic best sieving algorithm, is prone to memory bottlenecks in practice. Current cost estimates, that ignore the cost of memory, are therefore significantly lower than what we can (currently) achieve in practice.

6.3 BKZ

Cost estimate

In theory running the BKZ algorithm with a blocksize of β on a lattice of dimension d requires a polynomial number (in d) calls to an SVP oracle. More precisely, a single tour costs around d SVP oracle calls (most in dimension β), and up to $O(d^2 \log(d)/\beta^2)$ tours might be needed.

In practice, when using sieving, each block can often be HKZ reduced after a single SVP call (using the many short vectors found). So one only needs to consider about $d - \beta + 1$ blocks in a single tour. Secondly, instead of running many tours with blocksize β, it is better to progressively increase the blocksize and only run maybe a single tour per blocksize. When using the asymptotically fastest sieving algorithm as the SVP oracle the cost of running all tours up to dimension β is in theory only a factor about $C = 1/(1 - 2^{-0.292}) \approx 5.46$ more costly than running a single tour with blocksize β.

Estimate 6. *The cost of running a progressive BKZ algorithm with blocksize β is*

$$5.46 \cdot (d - \beta + 1) \cdot T_{svp}(\beta),$$

where T_{svp} is the cost of the SVP oracle.

Running progressive BKZ with only a single tour per blocksize introduces some loss of quality. The precise loss can be modeled using simulators. As a result one might have to increase the blocksize a bit to obtain the same profile as the Geometric Series Assumption indicates. Often this trade-off improves the overall run-time.

Simulator

While the Geometric Series Assumption gives a good first order estimate of the basis profile after BKZ-reduction, it is known to be inaccurate in small dimensions or when the dimension is only a small multiple of the blocksize due to the head and tail behavior. Additionally it does not account for the slower convergence when running progressive BKZ with only a few tours. To resolve these problems [CN11] introduced a BKZ simulator based on the Gaussian Heuristic.

This simulator keeps track of the basis profile $\ell = (\|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_d^*\|)$, and runs BKZ tours where instead of computing a shortest vector it just updates

$$\ell_\kappa = \min \left\{ \ell_\kappa, \mathrm{gh}(b) \cdot \left(\prod_{i=\kappa}^{\kappa+b-1} \ell_i \right)^{1/b} \right\},$$

for $b = \min\{\beta, n - \kappa\}$ (or experimental values of $\mathrm{gh}(b)$ for $b < 50$), and adjusts the remaining norms $\ell_\kappa, \ldots, \ell_{\kappa+b-1}$ accordingly. Such a simulator predicts correctly both the center (body) and tail part of the profile. This simulator was later refined by a probabilistic variant of the Gaussian Heuristic that can return slightly short vectors with a small probability [YD17; BSW18]. Due to this addition the head behavior is also captured. In short, these simulators allow for accurate and efficient predictions of the profile shape for random lattices, even for progressive BKZ with a limited number of tours.

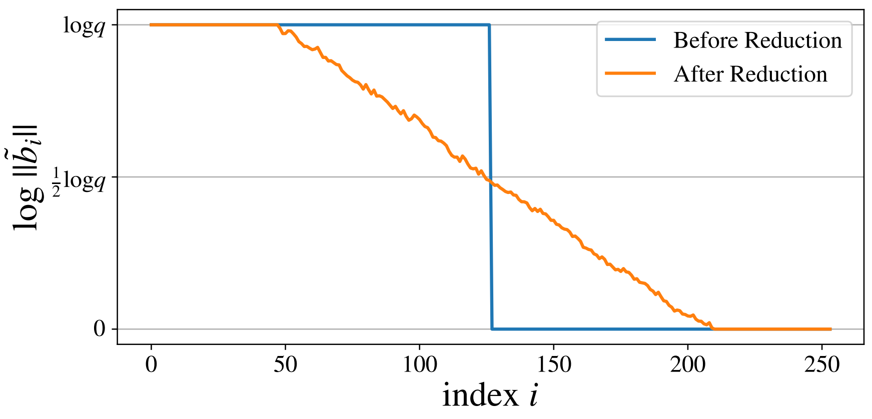


Figure 6.1: Typical Z-shape of the basis profile of q-ary lattices such as those arising from LWE or NTRU.

Remark 14. *The current lowest estimates for NIST candidates, such as [GJ21; MAT22], often use the GSA to estimate the basis profile, while using a progressive BKZ model for the cost. Accounting for this would increase the estimates by a few bits.*

Influence of Z-shape

Current BKZ simulators predict very well the profile of a random basis after BKZ reduction. However, sometimes the input basis is not so random. For example, in the common case of LWE and NTRU, the lattice is q-ary, and as a result the input bases often start with many orthogonal vectors of length q. These vectors can be much shorter than what one could get after e.g. LLL reduction, and they influence the shape of the basis profile, for example to a typical Z-shape as in Fig. 6.1. The GSA heuristic can be adapted to a Z-shape GSA heuristic, that is quite accurate as a first order approximation [Alb+21]. This special shape already enabled the dense sublattice attack for NTRU in the case the modulus q is large [ABD16; KF17; DW21]. For other attacks, such as the primal attack, it is generally assumed that the Z-shape structure has (mostly) disappeared around the successful blocksize. Because of this the Z-shape is ignored in current estimates.

When looking at the more refined simulators and cost models, there are still many open questions regarding the effect and cost of running BKZ on q-ary lattices. For example, experimentally (progressive) BKZ seems to improve the basis quicker, because essentially it only has to do much work on the smaller middle part. Secondly, current estimators fail to catch the precise shape of the reduced basis profile, with relatively large deviations in the last (non-flat) part of the profile. We expect the influence of these refinements to be minor, but they should still be modeled correctly to obtain precise estimates.

Summary 21 (Concrete behaviour of the BKZ algorithm). For general lattices the reduction behavior and cost of BKZ algorithms (in terms of SVP calls) can be predicted accurately by simulators. Most schemes resort to simplifications such as the GSA to obtain somewhat rougher estimates. There are still some open questions about the reduction behavior of BKZ on the q-ary lattices obtained from LWE and NTRU. However, for typical parameters the influence of the latter is expected to be small.

6.4 How do the NIST candidates get their security estimate?

The *bit-security* of a cryptosystem is calculated according to the most effective attack against that system. Since all these attacks have a different (sometimes complex) effect, it is not possible to abbreviate them all into a single simple formula. Even estimating and optimizing a single attack, with all the cost refinements, lower order improvements, and careful trade-offs, can be a complex computation.

Because of this complexity we depend on programs that try to estimate the cost of the individual attacks as best as possible, and based on the current state-of-the-art. There exist many such programs, each from different publications, each often focused on a few attacks. The currently most complete estimator is a product of multiple works [APS15; Alb+18; Alb+20a][1]. We included an Appendix to this work where it is explained how to use the mentioned estimator, both for current NIST proposals, as well as for custom parameters.

Summary 22 (Concrete bit-security estimates). The bit-security of a cryptosystem is based on the most effective attack. Computing the concrete cost of each attack can be rather complex, and thus is often done by programs that are written for this task. The Appendix includes an explanation on the usage of such programs, to go from LWE or NTRU parameters, to a concrete bit-security estimate.

[1] Available at `https://github.com/malb/lattice-estimator/`.

Part III

Structured attacks

Chapter 7

Attacks on ideal lattices

7.1 Introduction

Ideal lattices are specific structured lattices that arise from ideals in number fields. Though ideal lattices have only been sparsely used as the basis of a cryptographic system (e.g. [Gen09]), they are deemed interesting due to the apparent computational gap they show for classical computers versus quantum computers. Indeed, there seems to be a discrepancy between classical and quantum computers in how well they can find a short vector in these ideal lattices, especially those stemming from cyclotomic fields. Where classical computers can efficiently achieve an approximation factor of only 2^n using LLL-reduction, quantum computers efficiently achieve a much smaller $2^{O(\sqrt{n})}$ for ideal lattices in these cyclotomic fields [Cra+16; CDW17; DPW19; CDW21].

As said, this quantum attack applies also to general number fields [PHS19], though it has several drawbacks compared to cyclotomic fields. It is heuristic (i.e., not fully proven); it requires a pre-computation of worst-case $2^{O(n)}$ for a degree n number field (though this only needs to be computed *once* per field); and the approximation factor is expressed in slightly more complicated quantities, mainly involving the discriminant. Depending on the value this complicated quantity, the gain in shortness quality might be less impressive compared to the approximation factor of 2^n achieved by (classical) LLL-reduction.

This attack on ideal lattices heavily relies on the computation of the *(S-)unit group* and the *class group* of the number field considered. Without the knowledge and means of efficient computation within these groups, the attack would not proceed (as far as we know now). In the computation of these two groups is exactly where quantum computers play its key role: classically, no efficient algorithms are known computing these groups, whereas quantum computers can compute unit groups and class groups efficiently [Eis+14; BS15; BDF20]. Using a so-called *continuous hidden subgroup* framework, a quantum computer allows for efficiently computing class groups and (S-)unit groups, in the same fashion as quantum computers can factor large numbers into prime factors.

For rank > 1 module lattices, no such or similar attack is known (recall that ideal lattices can be considered as rank 1 module lattices). Though there are *reductions* known: one reduction[1] of the shortest vector problem in rank $r > 1$ module lattices to the same problem in a lower rank $r' > 1$ module lattice (where $r' < r$); and another reduction that reduces the shortest vector problem in rank 2 module lattices to a closest vector problem in a specific lattice that is related to the underlying number field [Lee+19; MS19]. These reductions are the subject of Chapter 8.

Road map

In this chapter, we postpone the quantum algorithm to the last section, for sake of clarity. In the first few sections we just *assume* that we can efficiently compute the (S-)unit group and class group of a number field and compute generators of principal ideals. In the last section, Section 7.5, we then show how this is done using a quantum computer. That way, the attack techniques for the shortest vector problem in ideal lattices are not clouded by the rather technical details of the quantum algorithm.

Section 7.2 introduces notions which are required for the ideal lattice attack, such as the definitions of the unit group and the class group. The next section, Section 7.3, explains how to find a mildly short vector in ideal lattices in cyclotomic fields. In Section 7.4 is shown how this attack is then generalized to general number fields. The last section, Section 7.5, treats the quantum computation of the class group and the (S-)unit group.

7.2 Preliminaries

In this chapter we will consider a number field K with a ring of integers $\mathcal{O}_K$. Recall from Section 3.2 that the Minkowski map $K \to K_\mathbb{R}, \alpha \mapsto (\sigma(\alpha))_\sigma$ makes ideals $I \subseteq \mathcal{O}_K$ into lattices of the $\mathbb{R}$-vector space $K_\mathbb{R}$. The length of an element $\alpha \in K$ is then defined as the length of the vector in the Minkowski space, i.e., $\|\alpha\| := \sum_\sigma |\sigma(\alpha)|^2$. We define the following logarithmic map $\mathrm{Log} : K_\mathbb{R} \to \mathrm{Log}(K_\mathbb{R})$, which is defined component-wise. For $(x_\sigma)_\sigma \in K_\mathbb{R}$, we define

$$\mathrm{Log}((x_\sigma)_\sigma) := (\log |x_\sigma|)_\sigma \in \mathrm{Log}(K_\mathbb{R}).$$

Note that via the Minkowski embedding this map is also defined on K, by $\mathrm{Log}(\alpha) := (\log |\sigma(\alpha)|)_\sigma \in K_\mathbb{R}$. This map has the nice property that it translates multiplication into addition: $\mathrm{Log}(\alpha\beta) = \mathrm{Log}(\alpha) + \mathrm{Log}(\beta)$. Hence, it maps multiplicative structures into additive structures, which is very useful in the multiplicative structure of the unit group.

[1]Here with 'reduction' is meant a computational reduction; a translation of one problem into another problem. Famous reductions in complexity theory are the one from SAT to 3SAT and from 3SAT to the clique problem in graphs.

Class groups and (S-)unit groups

Three groups play an important role in the attacks of this chapter: the unit group, the class group and the S-unit group (which can be considered a generalization of the first unit group). The unit group $\mathcal{O}_K^\times$ is defined by the following rule

$$\mathcal{O}_K^\times := \{u \in \mathcal{O}_K \mid \text{ there exists } v \in \mathcal{O}_K \text{ such that } uv = 1\}.$$

The unit group is a *multiplicative* group: if $u, v \in \mathcal{O}_K^\times$, then $uv \in \mathcal{O}_K^\times$. Hence, under the Log map, this multiplicative group changes into a discrete additive group inside the real vector space $K_\mathbb{R}$. Hence, $\operatorname{Log}(\mathcal{O}_K^\times)$, the image of $\mathcal{O}_K^\times$ under the map Log, is a *lattice*, which we call the logarithmic unit lattice.

To define the *class group* we first need the notion of *fractional* ideal. A fractional ideal is an ideal of the shape $\alpha \cdot I$, where $I \subseteq \mathcal{O}_K$ is an ideal and where $\alpha \in K\backslash\{0\}$. This notion is needed to make ideals *invertible*, such that one can speak of the group of all fractional ideals, which we call $\mathcal{I}_K$. So, the fractional ideal group is defined as follows.

$$\mathcal{I}_K := \{\alpha \cdot I \mid I \subseteq \mathcal{O}_K \text{ is an ideal and } \alpha \in K\backslash\{0\}\}.$$

Inside this commutative group lives the subgroup of *principal fractional ideals*, which are the fractional ideals generated by a single element, i.e., are of the shape $\alpha \cdot \mathcal{O}_K$. Formally,

$$\operatorname{Princ}_K := \{\alpha \cdot \mathcal{O}_K \mid \alpha \in K\backslash\{0\}\}.$$

The class group can then be defined as the quotient group of these two: $\operatorname{Cl}_K := \mathcal{I}_K/\operatorname{Princ}_K$. The class group is a *finite group* [Neu13] and somehow measures how many ideals can be written as a principal ideal. Most number fields have a large class group; as a general rule people expect it to be of the order of the $\sqrt{|\Delta_K|}$, where Δ_K is the discriminant of K, due to the Brauer-Siegel theorem.

Before we define the S-unit group, the notion of *prime ideals* of $\mathcal{O}_K$ is required. An ideal $\mathfrak{p} \subseteq \mathcal{O}_K$ is called a prime ideal if for all $\alpha, \beta \in \mathcal{O}_K$ holds that if $\alpha\beta \in \mathfrak{p}$ then $\alpha \in \mathfrak{p}$ or $\beta \in \mathfrak{p}$. Prime ideals are a generalization of prime numbers in $\mathbb{Q}$. The fundamental theorem of arithmetic generalizes to number fields, with the difference that it involves fractional ideals rather than numbers: any fractional ideal can be uniquely decomposed (up to order) into a product of prime ideals. I.e., any fractional ideal αI can be written as

$$\alpha I = \prod_{j=1}^{k} \mathfrak{p}_j^{n_\mathfrak{p}},$$

where $n_\mathfrak{p} \in \mathbb{Z}$ are allowed to be negative. Now let $S = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_\ell\}$ be a set of such prime ideals. Then the S-units are those elements $\alpha \in K\backslash\{0\}$, for which the fractional ideal $\alpha\mathcal{O}_K$ decomposes into primes in S. Formally,

$$\mathcal{O}_{K,S}^\times := \{\alpha \in K\backslash\{0\} \mid \alpha\mathcal{O}_K = \prod_{\mathfrak{p}\in S} \mathfrak{p}^{n_\mathfrak{p}} \text{ for some } n_\mathfrak{p} \in \mathbb{Z}\}.$$

Example 2. *The $\{2\}$-units in $\mathbb{Q}$ are precisely the elements $\pm 2^k$ for $k \in \mathbb{Z}$.*

Example 3. *Consider the number field $K = \mathbb{Q}(\sqrt{2})$. The units in $\mathcal{O}_K^\times$ are of the shape $\pm(3+2\sqrt{2})^k$ for some $k \in \mathbb{Z}$. That $3+2\sqrt{2}$ is a unit can be seen by evaluating $(3+2\sqrt{2})(-3+2\sqrt{2}) = 1$. That all units are of that specific shape follows by Dirichlet's unit theorem.*

Consider the prime ideal $\mathfrak{p}_7 = (7, 3+\sqrt{2}) = (3+\sqrt{2})$ of norm 7. Taking $S = \{\mathfrak{p}_7\}$, the S-units of $K = \mathbb{Q}(\sqrt{2})$ are $\mathcal{O}_{K,S}^\times = \{\pm(3+2\sqrt{2})^k \cdot (3+\sqrt{2})^j \mid k, j \in \mathbb{Z}\}$. In other words, these S-units consists of products of ordinary units and powers of $(3+\sqrt{2})$, the prime ideal $\mathfrak{p}_7$.

Ideal-SVP

We finish the preliminaries with the definition of the Ideal-SVP computational problem.

Problem 5 (Ideal-SVP)**.** *Given an* ideal *$I \subseteq \mathcal{O}_K$ of a degree t number field K, find a non-zero element $\alpha \in I$ such that*

$$\|\alpha\| \leq \gamma(t) \cdot N(I)^{1/t} \cdot |\Delta_K|^{1/(2t)} = \gamma(t) \cdot \mathrm{Vol}(I)^{1/t},$$

where $\gamma(t)$ is some function of the degree t, measuring the shortness of the vector.

One generally considers $\gamma(t) = \sqrt{t}$ to be *short* and $\gamma(t) = \mathrm{poly}(t)$ to be interesting for (ideal-lattice based[2]) cryptography, whereas $\gamma(t) = 2^t$ is considered trivial as solutions can then be found by means of LLL-reduction.

7.3 Solving Ideal-SVP in cyclotomic fields

7.3.1 Plan for Solving Ideal-SVP in Cyclotomic Fields

Finding a short vector in an ideal $I \subseteq \mathcal{O}_K$ in a cyclotomic field K consists of three parts, on which is elaborated in later sections.

- Finding a *close principal multiple*[3]. That is, finding another *small norm* integral ideal $J \subseteq \mathcal{O}_K$ such that the ideal product $I \cdot J$ is *principal*, i.e., generated by a *single element*: $I \cdot J = (\alpha_0)$ for some $\alpha_0 \in K^*$. In cyclotomic fields this can be done efficiently, by using a quantum algorithm for the class group discrete logarithm and by means of so-called *Stickelberger relations*. This step is treated in Section 7.3.2.
- Finding a (not necessarily small) *generator* α_0 of the principal ideal $I \cdot J$. This is done by a quantum algorithm, using the hidden subgroup framework, see Section 7.3.3.

[2]As said, almost no cryptography is based on ideals. For example, none of the NIST candidates is based on ideal lattices.

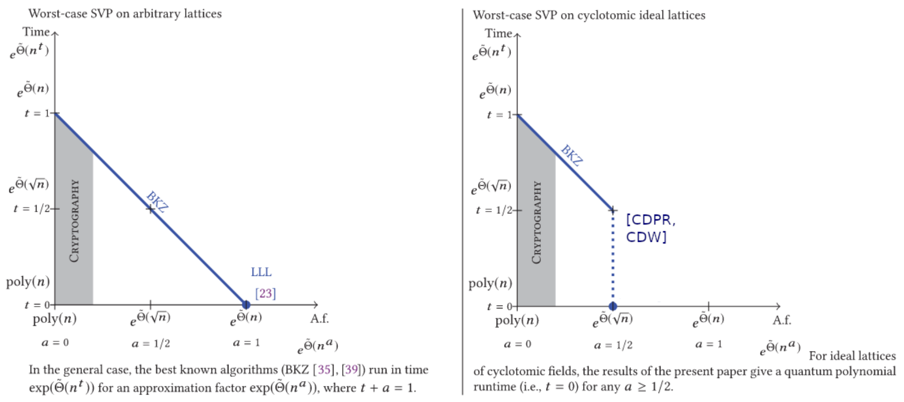
[3]Note that this can be skipped if ideal is principal (as is often the case in crypto)

- Reducing the generator, i.e., make the generator α_0 of $I \cdot J$ *smaller* by replacing it by $\alpha = \eta \cdot \alpha_0$ for a suitable unit $\eta \in \mathcal{O}_K^\times$. Also here, due to the special *cyclotomic units*, this can be done efficiently. This is explained in Section 7.3.4

This element $\alpha \in I \cdot J \subseteq I$ is then a possible solution for the Ideal-SVP problem on input I.

Result

We will see that this approach yields an approximation factor $\gamma(t) = \exp(O(\sqrt{t}))$ (in the context of Problem 5) within polynomial time using quantum computation power. I.e., this $\alpha \in I$ satisfies $\|\alpha\| \leq \exp(O(\sqrt{t})) \cdot \mathrm{Vol}(I)^{1/t}$.



In the general case, the best known algorithms (BKZ [35], [39]) run in time $\exp(\tilde{\Theta}(n^t))$ for an approximation factor $\exp(\tilde{\Theta}(n^a))$, where $t + a = 1$.

For ideal lattices of cyclotomic fields, the results of the present paper give a quantum polynomial runtime (i.e., $t = 0$) for any $a \geq 1/2$.

Figure 7.1: In arbitrary lattices, using BKZ, it takes time $\exp(\Theta(\sqrt{t}))$ to get a 'mildly' short vector of length $\exp(\Theta(\sqrt{t}))$ (left image). In the case of cyclotomic ideal lattices, such 'mildly' short vectors can be found in quantum polynomial time (right image). Picture from [CDW21].

7.3.2 Finding a close principal multiple

Given an ideal I, we would like to find another small ideal $J \subseteq \mathcal{O}_K$, such that $I \cdot J$ is principal (generated by a single element).

To tackle this problem, one looks at the *ideal class group* of K, which is the quotient group obtained by taking the group of fractional ideals and quotient it out by the group of principal ideals.

$$\mathrm{Cl}_K = \mathcal{I}_K / \mathrm{Princ}_K.$$

This is a *finite* group, and we denote the ideal class of an ideal I by $[I] \in \mathrm{Cl}_K$. The close principal multiple problem then translates into: find an ideal $J \subseteq \mathcal{O}_K$

that is small (i.e., has a small norm) and satisfies $[J] = [I]^{-1}$. Indeed, then $[IJ] = [I][J] = [I][I]^{-1} = 1$, which means that $IJ \in \text{Princ}_K$ is principal.

A reasonable guess would be that this small J consists of small multiples of relatively small prime ideals. So, to tackle this problem, one then often considers a set of (small) prime ideals $S = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_k\}$, and defines the following surjective[4] group homomorphism on the free commutative group $\mathbb{Z}^S$ to Cl_K.

$$\phi : \mathbb{Z}^S \to \text{Cl}_K, \quad (n_1, \ldots, n_k) \mapsto \left[\prod_{j=1}^{k} \mathfrak{p}_j^{n_j}\right].$$

Then, as $[I]^{-1} \in \text{Cl}_K$, one uses a *quantum algorithm*, namely the *class group discrete logarithm* algorithm (see Section 7.5), to find a (not necessarily small) inverse image of $[I]^{-1}$ under ϕ, i.e., $(n_1, \ldots, n_k) \in \mathbb{Z}^S$ such that

$$\phi(n_1, \ldots, n_k) = \left[\prod_{j=1}^{k} \mathfrak{p}_j^{n_j}\right] = [I]^{-1}.$$

So, then we could put $J = \prod_{j=1}^{k} \mathfrak{p}_j^{n_j}$, but there are two problems: (1) The exponents n_j could be large, so that J itself is large as well and (2) the exponents n_j could be negative, which makes J not an integral ideal.

The trick is to circumvent this by shifting $(n_1, \ldots, n_k)$ by elements of the *kernel lattice* of ϕ, i.e.,

$$\Lambda_0 = \left\{(m_1, \ldots, m_k) \in \mathbb{Z}^S \;\middle|\; [\prod_{j=1}^{k} \mathfrak{p}_j^{m_j}] = 1\right\} = \ker \phi.$$

In the case of cyclotomic fields, this kernel lattice has small so-called *Stickelberger relations* [CDW17; CDW21], to efficiently diminish the sizes of $(n_1, \ldots, n_k)$ and make them positive, in order to get a small $J = \prod_{j=1}^{k} \mathfrak{p}_j^{n_j}$ that satisfies $I \cdot J$ being principal.

7.3.3 Finding a generator of the principal ideal

Finding a (not necessarily small) generator of a principal ideal is done with a quantum algorithm, just like the class group discrete logarithm. As these two algorithms both fall into the same 'hidden subgroup' framework, the treatment is postponed to the later Section 7.5.

To continue the current discussion, we assume that we found a $\alpha_0 \in I \cdot J$ such that $(\alpha_0) = I \cdot J$.

[4] Assuming the Generalized Riemann Hypothesis, one can deduce that this homomorphism is surjective whenever S consists of the prime ideals with norm up to $12 \log(|\Delta_K|)^2$.

7.3.4 Reducing the generator

The generator α_0 of the principal ideal $I \cdot J$ found by the quantum algorithm is generally not short. In order to find a *short* generator of $I \cdot J$, we attempt to multiply α_0 by a suitable unit $\eta \in \mathcal{O}_K^\times$. It is a fact that $\alpha = \eta \cdot \alpha_0$ generates $I \cdot J$ if and only if α_0 does.

So, the remaining task is: Find $\eta \in \mathcal{O}_K^\star$ such that $\|\eta \cdot \alpha_0\|$ is minimal. In the literature (e.g., [Cra+16]), one now resorts to taking logarithms to translate this multiplicative problem into an additive problem. This compound logarithm, denoted Log, happens component-wise after the Minkowski embedding.

$$\mathrm{Log} : K \to \mathrm{Log}(K_{\mathbb{R}}),\ \alpha \longmapsto (\log|\sigma(\alpha)|)_\sigma.$$

Under this embedding, the units $\mathcal{O}_K^\times$ map to the *log unit lattice* $\mathrm{Log}(\mathcal{O}_K^\times)$ which is of full rank in the hyperplane $H = \{(x_\sigma)_\sigma \in \mathrm{Log}(K_{\mathbb{R}}) \mid \sum_\sigma x_\sigma = 0\}$.

CVP in the Log-unit-lattice

Reducing the generator α_0 is now done by reducing in the *logarithmic unit lattice*; the task is (by taking the logarithm) changed into: find a $\ell \in \mathrm{Log}(\mathcal{O}_K^\times)$ such that $\|\mathrm{Log}(\alpha) + \ell\|_\infty$ is minimal[5].

By projecting $\mathrm{Log}(\alpha_0)$ orthogonally to the hyperplane H, we obtain $\mathrm{Log}(\alpha_0)|_H \in H$, so that the minimizing problem translates in a *CVP instance* of the logarithmic unit lattice: find $\ell \in \mathrm{Log}(\mathcal{O}_K^\times)$ closest to $\mathrm{Log}(\alpha_0)|_H$.

In cyclotomic fields $\mathbb{Q}(\zeta_{p^k})$ of prime power modulus [Cra+16, Thm. 3.1] the *dual lattice* logarithmic unit lattice can be shown to have *very short vectors*. These vectors are the duals of the logarithms of the so-called 'cyclotomic units'. Such short dual vectors allow for *efficiently reducing* $\mathrm{Log}(\alpha_0)|_H$ to a much shorter $\mathrm{Log}(\alpha_0)|_H + \mathrm{Log}(\eta)$, thus finding the shortest generator $\alpha = \alpha_0 \cdot \eta$ of $I \cdot J$. Typically, even if one minimizes the length of $\mathrm{Log}(\alpha_0)|_H + \mathrm{Log}(\eta)$ by picking η optimally, one still has $\|\mathrm{Log}(\alpha_0)|_H + \mathrm{Log}(\eta)\|_\infty \geq \Theta(\sqrt{n})$, which is simply because the closest lattice point in the log-unit lattice can still be rather far away. This leads to an optimal but necessary factor $\exp(O(\sqrt{n}))$. In later work [CDW21] this argument is generalized for arbitrary cyclotomic fields, leading to the following summarized result.

Summary 23 (Attack on cyclotomic ideal lattices). There exists an attack on ideal lattices in prime-power cyclotomic fields of degree t that finds vectors $\mathbf{v} \in I$ with 'mildly short length' $\|\mathbf{v}\| \leq \exp(\tilde{O}(\sqrt{n})) \cdot \det(I)^{1/n}$.

[5]Note that the norm is changed from the 2-norm in the Minkowski space to the ∞-norm in the Log Minkowski space. Since we have (for a degree n number field) $\|(e^{x_\sigma})_\sigma\|_2 = (\sum_\sigma e^{2x_\sigma})^{1/2} \in [1, \sqrt{n}] \cdot e^{\|(x_\sigma)_\sigma\|_\infty}$, we could say that the infinity norm in the Log-space gives us the best relation with the 2-norm in the original space.

7.4 A Quantum Attack on Ideal Lattices in General Number Fields

7.4.1 Introduction

A natural question about the quantum attack on ideal lattices in cyclotomic fields would be: *Generalizes this attack to other number fields?*. The answer turns out to be *yes* heuristically [PHS19], but at the expense of a heavy pre-computation, that requires time and space exponential in the number field degree. Though, this pre-computation only needs to be done *once* for a fixed number field. In other words, with this pre-computed data associated with the number field K you can attack *all ideal lattices associated with the field* K.

As the underlying number field in a standardized cryptographic protocol is mostly assumed to be fixed, this 'single pre-computation per field' scenario is relevant indeed. Fortunately, the third round lattice-based NIST candidates are based on (rank > 1) module lattices rather than ideal lattices, so this attack seems not to apply to these candidates (as far as we know now). A possible generalization of this attack to (rank > 1) module lattices or even any attack on module lattices that is strictly better than a 'generic attack' (as in Chapter 5) is not found yet (see Chapter 8).

7.4.2 Differences between generalized and cyclotomic attack

The attack of Pellet-Mary, Hanrot and Stehlé [PHS19] generalizes the line of work [Cra+16; CDW17; Eis+14] and follows essentially the same plan as in Section 7.3.1. That is: Given an ideal I, find a close principal multiple $I \cdot J$, find a generator $\alpha_0 \in I \cdot J$ such that $(\alpha_0) = I \cdot J$ and reduce the generator α_0 using the units $\mathcal{O}_K^\times$ to obtain a small generator α, which is hopefully a short element in I.

However, the generalized attack allows more freedom in the step that reduces the (possibly large) generator α_0. Namely, the generator α_0 is reduced by the S-units $\mathcal{O}_{K,S}^\times \supset \mathcal{O}_K^\times$, which happens by reducing $\text{Log}(\alpha_0)$ by the Logarithmic S-unit lattice $\text{Log}(\mathcal{O}_{K,S}^\times)$. For this reduction to be successful and efficient, a pre-computation is needed, that essentially consists of building an exponential large data set that allows to solve BDD (Bounded Distance Decoding) in this Log-S-unit lattice efficiently [Laa16b].

Apart from the generalized attack being heuristic and requiring an expensive pre-computation step, it generally also has a worse quality output vector. The approximation factor achieved in this generalized attack is more intricately related to the discriminant and algorithmic properties of the logarithmic S-unit lattice.

This S-unit part of the generalized attack and its associated heavy pre-computation can in some way be seen as the compensation for the fact that general number fields (generally) do not have the nice Stickelberger properties and cyclotomic unit properties that the cyclotomic fields have. These structures in the class group and unit group of cyclotomic fields allow to make the close principal multiple step and the generator reduction step efficient (and successful up to the approximation factor $\exp(\sqrt{n})$).

7.4.3 Sketch of the attack

The goal of the attack is to find a (mildly) short non-zero element in $\alpha \in I$, where I is a given ideal of $\mathcal{O}_K$ of a number ring K.

- Start with defining $S = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_k\}$ to be the set of all prime ideals with norm up to the bound[6]$B = 12\log(|\Delta_K|)^2$.
- Use a quantum computer to find an $\alpha_0 \in I$ such that $(\alpha_0) = I \cdot \prod_{\mathfrak{p}\in S} \mathfrak{p}^{n_\mathfrak{p}}$, that is, solve the discrete logarithm in the class group.
- Attempt to minimize $\|\alpha_0 \cdot \eta\|$ over $\eta \in \mathcal{O}_{K,S}^\times$ (the S-units), given the constraint that all $m_\mathfrak{p} \geq 0$ in the decomposition $(\alpha_0 \cdot \eta) = I \cdot \prod_{\mathfrak{p}\in S} \mathfrak{p}^{m_\mathfrak{p}}$. This is done with the pre-computed data on the Log-S-unit lattice.
- Output $\alpha = \alpha_0 \cdot \eta$.

Example 4. *Suppose $\eta \in \mathcal{O}_{K,S}^\times$ has decomposition $(\eta) = \prod_{\mathfrak{p}\in S} \mathfrak{p}^{k_\mathfrak{p}}$, and $(\alpha_0) = I \cdot \prod_{\mathfrak{p}\in S} \mathfrak{p}^{n_\mathfrak{p}}$, then*

$$(\alpha_0 \cdot \eta) = I \cdot \prod_{\mathfrak{p}\in S} \mathfrak{p}^{n_\mathfrak{p}+k_\mathfrak{p}}.$$

Remark 15. *The constraint $m_\mathfrak{p} \geq 0$ for the decomposition $(\alpha_0 \cdot \eta) = I \cdot \prod_{\mathfrak{p}\in S} \mathfrak{p}^{m_\mathfrak{p}}$ is required in order to have $\alpha_0 \cdot \eta \in I$.*

Summary 24 (Quantum attack on ideal lattices in general number fields). The attack on ideal lattices in cyclotomic fields by [CDW21; Cra+16] described in Section 7.3 generalizes to general number fields; for the expense of an *exponentially heavy pre-computation* in the degree of the number field, that only needs to be computed once per field.

7.4.4 The Log-S-unit lattice and the problem of positive exponents

As with the unit attack, minimizing $\|\alpha_0 \cdot \eta\|$ is equivalent to minimizing the norm of

$$\mathrm{Log}(\alpha_0 \cdot \eta) = \mathrm{Log}(\alpha_0) + \mathrm{Log}(\eta) \text{ over } \eta \in \mathcal{O}_{K,S}^\times.$$

[6]In the actual attack, a quantum computer is used to diminish this bound to $\log(h_K)$, where h_K is the class number of K, since this gives better bounds. For our current intuitive explanation, we will skip this step.

So, essentially, the task is to find $\mathrm{Log}(\eta) \in \mathrm{Log}(\mathcal{O}_{K,S}^\times)$ that is the closest to $\mathrm{Log}(\alpha_0)$. But, in the S-units we have the additional constraint that $\alpha_0 \cdot \eta$ is required to be an *element of the input ideal* I, i.e., the decomposition

$$(\alpha_0 \cdot \eta) = I \cdot \prod_{\mathfrak{p} \in S} \mathfrak{p}^{m_\mathfrak{p}}$$

needs to have exponents $m_\mathfrak{p} \geq 0$.

This problem is solved by using the Log-S-unit lattice and adding some 'drift' β to the prime ideal coordinates.

The Log-S-unit lattice

One can define a log-S-unit lattice by giving each prime in S an own 'dimension', though this dimension is *integral* (i.e., equal to $\mathbb{Z}$). So, an element $\eta \in \mathcal{O}_{K,S}^\times$ maps to the vector

$$\mathrm{Log}_{(\mathrm{S})} : \mathcal{O}_{K,S}^\times \to H \times \mathbb{Z}^{|S|}, \eta \longmapsto (-c \cdot \mathrm{Log}(\eta)|_H, (v_\mathfrak{p}(\eta))_\mathfrak{p}).$$

Here $\cdot|_H : \log(K_\mathbb{R}) \to H$ projects the log-space to the hyperplane $H = \{(x_\sigma)_\sigma \mid \sum_\sigma x_\sigma = 0\}$; the factor $c = \frac{t^{3/2}}{|S|}$ is for the optimal balance between the 'H' part and the $\mathbb{Z}^{|S|}$ part. In the full analysis, this shows to be the optimal choice for obtaining the best vectors of the input ideal I. One can show that $\mathrm{Log}_{(\mathrm{S})}(\mathcal{O}_{K,S}^\times)$ is a lattice in $H \times \mathbb{Z}^{|S|}$.

Minimizing in the Log-S-unit lattice

Minimizing $\mathrm{Log}(\alpha_0 \cdot \eta)$ for $\eta \in \mathcal{O}_{K,S}^\times$ is equivalent to finding the closest vector in $\mathrm{Log}_{(\mathrm{S})}(\mathcal{O}_{K,S}^\times)$ to $\mathrm{Log}_{(\mathrm{S})}(\alpha_0) := (-c\,\mathrm{Log}(\alpha_0)|_H, (n_\mathfrak{p})_\mathfrak{p})$. Here, $\alpha_0 \in I$ is the (not necessarily small) element found by the quantum computer that satisfies $(\alpha_0) = I \cdot \prod_{\mathfrak{p} \in S} \mathfrak{p}^{n_\mathfrak{p}}$.

It might well be that the closest vector in $\mathrm{Log}_{(\mathrm{S})}(\mathcal{O}_{K,S}^\times)$ to $\mathrm{Log}_{(\mathrm{S})}(\alpha_0) := (-c\,\mathrm{Log}(\alpha_0)|_H, (n_\mathfrak{p})_\mathfrak{p})$ has *negative* prime coordinates, which gives us invalid solutions. To solve this, we add a 'drift' $\beta \in \mathbb{N}$ to each prime coordinate. This translates into: Find a vector $\mathbf{v} \in \mathrm{Log}_{(\mathrm{S})}(\mathcal{O}_{K,S}^\times)$ that is β-close to the vector

$$\mathrm{Log}_{(\mathrm{S})}(\alpha_0) - (\mathbf{0}, (\beta)_\mathfrak{p}) := (-c\,\mathrm{Log}(\alpha_0)|_H, (n_\mathfrak{p} - \beta)_\mathfrak{p}).$$

in the *maximum norm*, denoted by $\| \cdot \|_\infty$.

Properties of a β-close vector Suppose we found a $\eta \in \mathcal{O}_{K,S}^\times$ that solves this β-Closest Vector problem; i.e., $\eta \in \mathcal{O}_{K,S}^\times$ satisfies

$$\| \mathrm{Log}_{(\mathrm{S})}(\alpha_0) - (\mathbf{0}, (\beta)_\mathfrak{p}) + \mathrm{Log}_{(\mathrm{S})}(\eta)\|_\infty = \|(-c\,\mathrm{Log}(\alpha_0\eta)|_H, (n_\mathfrak{p} - \beta + v_\mathfrak{p}(\eta))_\mathfrak{p})\|_\infty \leq \beta.$$

Then surely, for all $\mathfrak{p} \in S$, $|n_\mathfrak{p} - \beta + v_\mathfrak{p}(\eta)| \leq \beta$, i.e., $0 \leq n_\mathfrak{p} + v_\mathfrak{p}(\eta) \leq 2\beta$, i.e., the decomposition of

$$(\alpha_0 \cdot \eta) = I \cdot \prod_{\mathfrak{p} \in S} \mathfrak{p}^{n_\mathfrak{p} + v_\mathfrak{p}(\eta)} \tag{7.1}$$

only has positive exponents $n_\mathfrak{p} + v_\mathfrak{p}(\eta)$ in the primes. So, phrasing the problem as a 'drifted' CVP problem indeed yields sound solutions.

7.4.5 Quality of the output

We can write

$$\operatorname{Log}(\alpha_0\eta) = \operatorname{Log}(\alpha_0\eta)|_H + \frac{1}{t} \cdot \log(N(\alpha_0\eta)) \cdot \mathbf{1},$$

where $t = [K : \mathbb{Q}]$ is the field degree and $\mathbf{1}$ is the all-one vector. This yields

$$\begin{aligned} \|\alpha_0\eta\|_2 &\leq \exp(\|\operatorname{Log}(\alpha_0\eta)|_H\|_\infty) \cdot \exp(\log(N(\alpha_0\eta)^{1/t})) \qquad (7.2) \\ &\leq \exp(\|\operatorname{Log}(\alpha_0\eta)|_H \cdot N(\alpha_0\eta)^{1/t}. \qquad (7.3) \end{aligned}$$

Starting with the right expression, we have (see Equation (7.1)),

$$N(\alpha_0\eta)^{1/t} = N(I)^{1/t} \cdot \prod_{\mathfrak{p}\in S} N(\mathfrak{p})^{\frac{n_\mathfrak{p}+v_\mathfrak{p}(\eta)}{t}} \leq N(I)^{1/t} \cdot (\operatorname{poly}(\log(\Delta_K)))^{\frac{2\cdot\beta\cdot|S|}{t}},$$

since $0 \leq n_\mathfrak{p} + v_\mathfrak{p}(\eta) \leq 2\beta$, and the norm of the largest prime in S is around $\operatorname{poly}(\log(|\Delta_K|))$.

Continuing with the left expression, we have $\|c \operatorname{Log}(\alpha_0\eta)|_H\|_\infty \leq \beta$, where $c = \frac{t^{3/2}}{|S|}$ is the 'correction factor'. Therefore,

$$\|\operatorname{Log}(\alpha_0\eta)|_H\|_2 \leq \sqrt{t}\|\operatorname{Log}(\alpha_0\eta)|_H\|_\infty \leq \sqrt{t} \cdot \beta/c = \beta \cdot |S|/t.$$

Putting these inequalities into Equation (7.3), one obtains

$$\|\alpha_0\eta\|_2 \leq \underbrace{\exp\left(O\left(\frac{\beta \cdot |S| \cdot \log\log \Delta_K}{t}\right)\right)}_{\text{approximation factor}} \cdot N(I)^{1/t}.$$

Using quantum computation power, one can diminish the size of $|S|$ to $\log|\Delta_K|$ [PHS19, Lemma 2.7]; writing $\beta = (\log|\Delta_K|)^\alpha$, we get the following result. Here $\tilde{O}$ is similar to the O-notation, except that in the $\tilde{O}$, polylogarithmic factors are suppressed.

Summary 25 (idealSVP in general number fields reduces to CVP in the log-unit lattice). Assuming quantum computation power, solving $(\log|\Delta_K|)^\alpha$-CVP (in the maximum norm $\|\cdot\|_\infty$, $\alpha > 0$) in the Log-S-unit lattice of a degree t number field K allows for solving Ideal-SVP in K with approximation factor

$$\exp\left(\tilde{O}\left(\frac{(\log|\Delta_K|)^{1+\alpha}}{t}\right)\right),$$

where Δ_K is the discriminant of K.

7.4.6 Complexity and quality of the output

CVP with pre-processing

Laarhoven [Laa16b, Corollaries 2 and 3] proved the following statement heuristically. Given $\alpha \in [0, 1/2]$ and given $\mathbf{t} \in \operatorname{span}(\mathcal{L})$ where $\mathcal{L}$ is a d-dimensional lattice, there exists an algorithm that computes a vector $\mathbf{v} \in \mathcal{L}$ with

$$\|\mathbf{t} - \mathbf{v}\| \leq O(n^{\alpha}) \cdot \operatorname{dist}_2(\mathbf{t}, \mathcal{L}),$$

in time $\exp(\tilde{O}(n^{1-2\alpha}))$, given a pre-processing stage depending only on $\mathcal{L}$, that requires time $\exp(O(n))$. Here, $\operatorname{dist}_2(\mathbf{t}, \mathcal{L})$ is the minimum distance of the target $\mathbf{t}$ to the lattice $\mathcal{L}$.

Cyclotomic fields and other 'small discriminant' fields

For degree t cyclotomic fields, we have that we have $\log |\Delta_K| \leq O(t)$, turning the approximation factor in Summary 25 into

$$\exp\left(\tilde{O}\left(\frac{(\log |\Delta_K|)^{1+\alpha}}{t}\right)\right) = \exp(\tilde{O}(t^{\alpha})),$$

if we are able to solve $\log |\Delta_K|^{\alpha} = t^{\alpha}$-$\mathrm{CVP}_{\infty}$ in the log-S-unit lattice[7]. The same holds of course for number field families that have $\log |\Delta_K| \leq O(t)$.

Combining this with Laarhoven's result, yields

Summary 26 (Quantum computers solve $\exp(\sqrt{n})$-idealSVP in general small-discriminatn fields, assuming precomputation). Assuming quantum computation power, with a pre-processing stage of time and memory $\exp(O(t))$, one can solve $\exp(\tilde{O}(t^{\alpha}))$-approximate IdealSVP in time $\exp(\tilde{O}(n^{1-2\alpha}))$ in degree t number fields K satisfying $\log |\Delta_K| = O(t)$, see Figure 7.2.

7.5 Quantum algorithm for computing (S-)unit groups and class groups

7.5.1 Overview

In this part we sketch how quantum algorithms are able to find a discrete logarithm in the class group, a generator of a principal ideal and the $(S-)$ unit group. Those are exactly the missing parts of the previous sections that rely on quantum computation. All these tasks fall into one general framework, namely that of the 'hidden subgroup problem'.

[7]CVP_{∞} means CVP with respect to the maximum norm

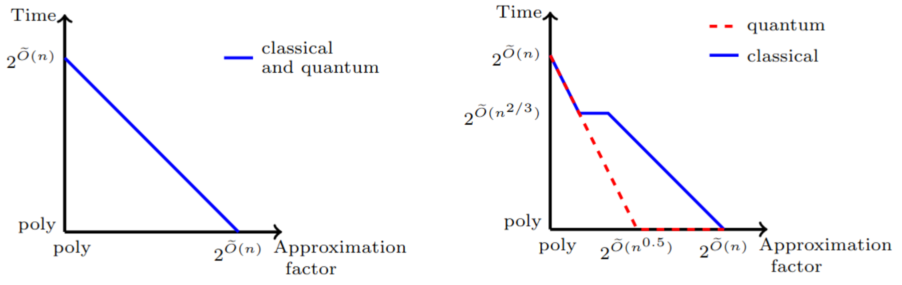


Figure 7.2: Left: The usual, unstructured time/approximation-factor trade-offs for Approximated SVP in number fields of degree n. On the right the trade-offs of PHS for number fields with $\log|\Delta_K| = \tilde{O}(n)$, taking into account a pre-processing of $\exp(O(n))$. Image from [PHS19].

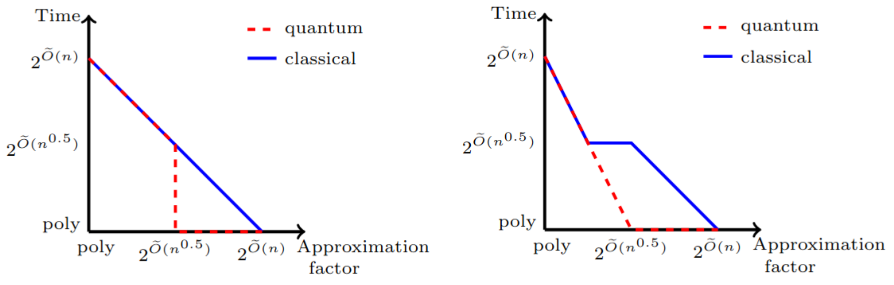


Figure 7.3: On the left, the time/approximation-factor trade-offs for the first structured attack on prime power cyclotomic fields of degree n, described in Section 7.3 (see also Figure 7.1). On the right, the trade-offs of the PHS attack with the same fields. Image from [PHS19].

7.5.2 The Hidden Subgroup Problem

The core of the quantum algorithms needed for the attacks in this chapter is a *hidden subgroup algorithm* which solves the so-called *hidden subgroup problem*, that can be superficially described as follows.

Definition 19 (Hidden subgroup problem)**.** *Given a group of the shape $\mathbb{Z}^k \times \mathbb{R}^n$ and a function*[8] $f : \mathbb{Z}^k \times \mathbb{R}^n \to \mathbb{C}^N$ *that satisfies* $f(x + t) = f(x)$ *for all* $t \in T$ *and* $x \in \mathbb{Z}^k \times \mathbb{R}^n$*, where* $T \subseteq \mathbb{Z}^k \times \mathbb{R}^n$ *is a 'hidden' subgroup of* G*. Additionally, the function* f *is required to be* smooth *and* not too constant[9].

The task is to find (approximated) generators and relations of the group T.

In this survey we assume that a quantum computer, given such adequate function f, is able to find the subgroup $T \subseteq \mathbb{Z}^k \times \mathbb{R}^n$. This function f is referred to as an *oracle function*, which essentially captures information about the subgroup T.

7.5.3 Preliminaries for the Oracle Function

In order to find a generator of a principal ideal, or to find the discrete logarithm in the class group, we need to define an adequate oracle function f. One of the ingredients of such oracle function is the *lattice quantum superposition*.

Ideal lattices

For this oracle function we need the space of (fractional) ideals to be more 'continuous', which we call the group of *ideal lattices of* K, denoted IdLat_K. Such a (continuous) ideal lattice is of the shape $x \cdot I$, where $I \subseteq \mathcal{O}_K$ is an ideal and where $x = (x_\sigma)_\sigma \in K_\mathbb{R}$. With $x \cdot I$ we mean the following lattice:

$$\{(x_\sigma \sigma(\alpha))_\sigma \in K_\mathbb{R} \mid \alpha \in I\}.$$

We can see that IdLat_K forms a group and the unit ideal lattice is $\mathcal{O}_K$. The representation $x \cdot I$ of the ideal lattice is not unique, there are many other $y \cdot J$ such that $x \cdot I = y \cdot J$. This will play a role for the later oracle function.

Another important part of the oracle function is the *exponential function* that maps H to $K_\mathbb{R}$. We already defined the Logarithmic map $\mathrm{Log} : K_\mathbb{R} \to \mathrm{Log}(K_\mathbb{R}) \supseteq H$ in Section 7.2, where

$$H = \{(h_\sigma)_\sigma \in \mathrm{Log}(K_\mathbb{R}) \mid \sum_\sigma h_\sigma = 0\}.$$

The exponential function $\mathrm{Exp} : H \to K_\mathbb{R}$ is the inverse of this map Logarithmic map and sends $(h_\sigma)_\sigma \in H$ to $(e^{h_\sigma})_\sigma \in K_\mathbb{R}$. We use throughout this section that H is isomorphic to $\mathbb{R}^\ell$ for some $\ell \geq 0$; indeed, a quantum computer can then apply the hidden subgroup problem (as the groups that will occur in the following text are then all of the shape $\mathbb{Z}^k \times \mathbb{R}^\ell$).

[8] This function is allowed to have *quantum states* as an output.

[9] The precise requirements for the function f are slightly more technical, see for example [Eis+14; BDF20]

The lattice quantum superposition

The most important part of the oracle function f will be the encoding of an ideal lattice $x \cdot I \subseteq K_{\mathbb{R}}$ (with $x \in K_{\mathbb{R}}$, and where the inclusion is via the Minkowski embedding) into a quantum state. This is done by a *Gaussian superposition*, which sends the ideal lattice $x \cdot I \subseteq K_{\mathbb{R}}$ to the quantum state[10]

$$|xI\rangle := \sum_{v \in x \cdot I} e^{-s\|v\|^2} \cdot |v\rangle, \tag{7.4}$$

for a certain large enough $s \in \mathbb{R}_{>0}$.

This superposition $|xI\rangle$ quite captures the 'essence' of the ideal lattice xI. Indeed, $|xI\rangle = |yJ\rangle$ if and only if $xI = yJ$.

Remark 16. *In order to build this superposition on a real-life quantum computer, one would need to take care of machine precision numbers, as it is not possible to represent elements of $K_{\mathbb{R}}$ in full precision. In this survey we ignore these issues.*

7.5.4 The oracle function for finding the generator of a principal ideal

On of the tasks that were left to be done by a quantum machine is the task of finding a generator of a principal ideal (see Section 7.3.3). That is, given a principal ideal $\mathfrak{a}$, one would like to find $\alpha \in \mathfrak{a}$ such that $\mathfrak{a} = (\alpha)$.

For any $h \in H = \{(x_\sigma)_\sigma \in \text{Log}(K_{\mathbb{R}}) \mid \sum_\sigma x_\sigma = 0\}$ in the hyperplane and any integer $j \in \mathbb{Z}$ we construct an ideal lattice $N(\mathfrak{a})^{-j/t} \cdot \text{Exp}(h) \cdot \mathfrak{a}^j$.

This is indeed an ideal lattice of the form $x \cdot I$, by putting $x = N(\mathfrak{a})^{-j/t} \cdot \text{Exp}(h) \in K_{\mathbb{R}}$ and $I = \mathfrak{a}^j$ (the factor $N(\mathfrak{a})^{-j/t}$ is put in to make the ideal have the same determinant as $\mathcal{O}_K$).

So, our oracle function f is then defined as follows:

$$\mathbb{Z} \times H \to \{\text{quantum states}\}, \qquad (j,h) \longmapsto \left| N(\mathfrak{a})^{-j/t} \cdot \text{Exp}(h) \cdot \mathfrak{a}^j \right\rangle, \tag{7.5}$$

where $|x \cdot I\rangle$ is as in Equation (7.4).

The hidden subgroup of this oracle function

What is the hidden subgroup T of the oracle function f as in Equation (7.5)? For this case it is enough to know which elements in $\mathbb{Z} \times H$ send to the same state as $|\mathcal{O}_K\rangle$, the 'unit' state (that encodes the ring of integers $\mathcal{O}_K$). Suppose

$$|N(\mathfrak{a})^{-j/t} \cdot \text{Exp}(h) \cdot \mathfrak{a}^j\rangle = |\mathcal{O}_K\rangle. \tag{7.6}$$

[10]Actually, quantum states are required to have unit norm, so this state must have a constant in the front as to make the total state having norm 1. In this survey, for sake of clarity and conciseness, we omit these constants.

Let us write $\mathfrak{a} = (\alpha)$ for some element $\alpha \in \mathfrak{a}$. Such an α necessarily exists, we just do not know this element yet. We have, by Equation (7.6) and the fact that the lattice quantum superposition is the same, that the following ideal lattices are the same.

$$|N(\alpha)|^{-j/t} \cdot \mathrm{Exp}(h) \cdot (\alpha^j) = N(\mathfrak{a})^{-j/t} \cdot \mathrm{Exp}(h) \cdot \mathfrak{a}^j = \mathcal{O}_K$$

In other words, by taking Logarithms and noting that $\mathcal{O}_K$ is can be generated by any unit $\eta \in \mathcal{O}_K^\times$, the element $h = (h_\sigma)_\sigma \in H$ satisfies, for some unit $\eta \in \mathcal{O}_K^\times$,

$$(h_\sigma)_\sigma = -\frac{j}{t} \cdot \log |N(\alpha)| - \mathrm{Log}(\alpha^j \cdot \eta)$$

So, the hidden subgroup $T \subseteq \mathbb{Z} \times H$ of f must consists of (j, h) with $h \in H$ of above shape, i.e.,

$$T = \left\{ \left(j, -\frac{j}{t} \cdot \log |N(\alpha)| - \mathrm{Log}(\alpha^j \cdot \eta)\right) \;\middle|\; j \in \mathbb{Z}, \eta \in \mathcal{O}_K^\times \right\}$$

If we just know two elements $(j, h), (j', h') \in T$ for which j and j' are coprime, one can find (by the extended euclidean algorithm), $a, a' \in \mathbb{Z}$ such that $aj + a'j' = 1$. In that case one necessarily obtains an element in T of the shape

$$(1, -\frac{1}{t} \cdot \log |N(\alpha)| - \mathrm{Log}(\alpha \cdot \eta),$$

for some $\eta \in \mathcal{O}_K^\times$. Since we can compute $\log |N(\alpha)| = \log(N(\mathfrak{a}))$, one can then recover (an approximation of) $\mathrm{Log}(\alpha \cdot \eta)$. From this logarithmic image, one is (with a good enough approximation) able to recover $\alpha \cdot \eta$, which is a generator of $\mathfrak{a}$.

Remark 17. *Note that no guarantee is given on the length of this generator; the output of the quantum algorithm is even given in logarithmic coordinates, which might yield generator $\alpha \cdot \eta$ which is even too large to write down.*

So, the next step of the total algorithm (namely, to reduce this element modulo the logarithmic unit lattice) is of importance to actually obtain a mildly short generator of $\mathfrak{a}$.

Remark 18. *In this example, we used an extended euclidean algorithm to obtain a generator of $\mathfrak{a}$. In reality, one might resort to many samples $(j, h) \in T$ and use lattice reduction techniques to get an overall small element of the shape $(1, h')$.*

7.5.5 The oracle function for finding the discrete logarithm in the class group

The task that we are trying to solve is: Given an ideal class $[\mathfrak{a}] \in \mathrm{Cl}_K = \mathcal{I}_K / K^*$ in the ideal class group represented by the ideal $\mathfrak{a}$, and given a set of k prime ideals $S = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_k\}$. Find exponents $n_j \in \mathbb{Z}_{\geq 0}$ such that $[\prod_{j=1}^k \mathfrak{p}_j^{n_j}] = [\mathfrak{a}]^{-1}$.

Again, we construct an oracle function, now defined on $\mathbb{Z} \times \mathbb{Z}^k \times H$, where H is again the hyperplane where the Logarithmic units live in.

The oracle function is defined as follows

$$f : \mathbb{Z} \times \mathbb{Z}^k \times H \to \{\text{quantum states}\}, \tag{7.7}$$

$$(r, (n_j)_{j \in [k]}, h) \longmapsto |N^{-1/t} \cdot \operatorname{Exp}(h) \cdot \mathfrak{a}^r \cdot \prod_j \mathfrak{p}_j^{n_j}\rangle \tag{7.8}$$

where $N = N(\mathfrak{a})^r \cdot \prod_j N(\mathfrak{p}_j)^{n_j}$; this is to scale the lattice down to have the same determinant as $\mathcal{O}_K$. Here, again, $|xI\rangle$ is the lattice quantum superposition.

The hidden subgroup of this oracle function

By the same reasoning as in the previous section, the hidden subgroup T are those elements $(r, (n_j)_j, (h_\sigma)_\sigma)$ in $\mathbb{Z} \times \mathbb{Z}^k \times H$ for which holds

$$N^{-1/t} \cdot \operatorname{Exp}(h) \cdot \mathfrak{a}^r \cdot \prod_j \mathfrak{p}_j^{n_j} = \mathcal{O}_K$$

By taking a linear integral combination of multiple such elements $(r, (n_j)_j, h) \in T$, one can compute one of the shape $(1, (n_j)_j, h) \in T$ for which $n_j \geq 0$, for all $j \in [k]$. This means that

$$\mathfrak{a} \cdot \prod_{j=1}^{k} \mathfrak{p}_j^{n_j} = N^{1/t} \cdot \operatorname{Exp}(h) \cdot \mathcal{O}_K,$$

from which you can conclude that $\mathfrak{a} \cdot \prod_{j=1}^{k} \mathfrak{p}_j^{n_j}$ is principal and that $[\mathfrak{a}]^{-1} = [\prod_{j=1}^{k} \mathfrak{p}_j^{n_j}]$. Moreover, by computing $N^{1/t} \cdot \operatorname{Exp}(h) \in K_\mathbb{R}$ sufficiently precise, one can then obtain an element $\alpha \in K$ such that $\alpha = N^{1/t} \cdot \operatorname{Exp}(h)$. By these means we then found

$$\mathfrak{a} \cdot \prod_{j=1}^{k} \mathfrak{p}_j^{n_j} = (\alpha),$$

i.e.m α is a generator of the ideal $\mathfrak{a} \cdot \prod_{j=1}^{k} \mathfrak{p}_j^{n_j}$.

Summary 27 (Quantum computers solve the 'continuous' hidden subgroup problem). Quantum computers are able to efficiently find the periodicity of a periodic function on a space of the shape $\mathbb{Z}^k \times \mathbb{R}^n$. By choosing this function adequately, this allows for computing class groups and unit groups of number fields efficiently.

Chapter 8

Attacks on Module Lattices

8.1 Introduction

8.1.1 Attacks

Module lattices arise in ModuleLWE, RingLWE and NTRU, and are mostly of rank 2 or higher. There are no attacks known on module lattices that are better than the attacks on generic lattices. In other words, module lattices, from a current-time algorithmic perspective, seem to be as hard as generic lattices for the time being. This is the main reason why lattice-based cryptography is mostly based on these lattices: using them is more efficient, and (as far as we know) there is no increase in security risks.

As far as we know, the quantum attack on ideal lattices (rank 1 module lattices) described in the previous sections *does not generalize* to higher rank module lattices.

Summary 28 (Attacks on Module lattices). There are no attacks known for (rank > 1) module lattices that are better than the 'generic attacks' that apply to general lattices. In other words, as far as we know now, there is no algorithm (better than generic ones) that exploits the structure of these module lattices.

8.1.2 Reductions

There are no known module-lattice specific attack, but there is a certain *reduction*, independently found by Lee–Pellet-Mary–Stehlé–Wallet and Mukherjee–Stephens-Davidowitz [Lee+19; MS19]. In this reduction, SVP in module lattices of higher rank r' are reduced to SVP in module lattices of rank 2, which is highly inspired on the LLL-algorithm and does in essence generalize it for number fields. For this reduction there is only a 'small' blow up (dominantly depending exponentially on the module rank r') in the approximation factor. As the rank for module-lattice based cryptography are generally quite small (most of them not larger than 5), this suggests that the hardness of these cryptographic schemes relies on the hardness of finding short vectors in rank 2 module lattices.

As far as we know now, no reduction is known from rank 2 module lattices to rank 1 module lattices (also known as ideal lattices), so it is believed that there is a significant hardness gap between rank 1 module lattices (ideal lattices) and rank > 1 module lattices.

Though there is no reduction for SVP on rank 2 module lattices to rank 1 module lattices, in the work [Lee+19] there is a (quantum) reduction from SVP on rank 2 module lattices (of a fixed number field) to (exact) CVP in a specific log-S-unit alike lattice; this reduction has similarities to the reduction in the quantum attack on ideal lattices in general number fields.

8.2 LLL (and BKZ) in module lattices

8.2.1 Introduction

In this section, we treat a generalization of the LLL algorithm to module-lattices, in the framework of Lee et al. [Lee+19]. Essentially same generalization can be applied to the BKZ algorithm [MS19], but will be omitted in this text.

8.2.2 Basis operations in module bases

The reduction for modules [Lee+19; MS19] are in essence generalizations of the LLL algorithm for number fields. Recall that basis reduction algorithms (like LLL) manipulate bases $\mathbf{B} \in \mathbb{Z}^{n \times n}$ by means of multiplying them on the right with $\mathbf{U} \in \mathcal{GL}_n(\mathbb{Z})$, yielding $\mathbf{BU}$. This multiplying on the right boils just down to (invertible) column operations (switching columns, adding a multiple of one column to another, etc.).

To generalize LLL to number fields, on now attempts to manipulate *module bases* $\mathbf{B} \in R^{n \times n}$ (where $R = \mathbb{Z}[x]/(\psi(x))$ for some polynomial[1]) by multiplying them on the right with $\mathbf{U} \in \mathcal{GL}_n(R) = \{\mathbf{U} \in R^{n \times n} \mid \det(\mathbf{U}) \in R^\times\}$. In other words, $\mathcal{GL}_n(R)$ are matrices with coefficients in R whose determinant is a *unit* in R. Again, this boils down to invertible column operations on $\mathbf{B}$, but now at 'ring coefficient' level.

Example 5. *Let $R = \mathbb{Z}[\sqrt{2}] = \mathbb{Z}[x]/(x^2 - 2)$ and let*

$$\mathbf{B} = \begin{bmatrix} 21 + 10\sqrt{2} & 2 + 3\sqrt{2} \\ 8 + 12\sqrt{2} & 1 + \sqrt{2} \end{bmatrix}.$$

Subtracting $3\sqrt{2}$ times the second column from the first column accounts to multiplying on the right with

$$\mathbf{U} = \begin{bmatrix} 1 & -3\sqrt{2} \\ 0 & 1 \end{bmatrix} \in \mathcal{GL}_n(R).$$

[1]This is of course not the most general shape of a number ring, but for simplicity we will assume this for the moment.

The left column of $\mathbf{B} \cdot \mathbf{U}$ *is then*

$$\begin{bmatrix} 21+10\sqrt{2} \\ 8+12\sqrt{2} \end{bmatrix} - 3\sqrt{2} \begin{bmatrix} 2+3\sqrt{2} \\ 1+\sqrt{2} \end{bmatrix} = \begin{bmatrix} 21+10\sqrt{2} \\ 8+12\sqrt{2} \end{bmatrix} - \begin{bmatrix} 6\sqrt{2}+18 \\ 3\sqrt{2}+6 \end{bmatrix} = \begin{bmatrix} 3+4\sqrt{2} \\ 2+9\sqrt{2} \end{bmatrix}.$$

So, as the right column of $\mathbf{B}$ *keeps fixed,*

$$\mathbf{B} \cdot \mathbf{U} = \begin{bmatrix} 3+4\sqrt{2} & 2+3\sqrt{2} \\ 2+9\sqrt{2} & 1+\sqrt{2} \end{bmatrix}.$$

Remark 19. *Module lattices with bases* $\mathbf{B} \in R^{n\times n}$ *are called* free modules. *To simplify matters, we will focus on these particular module lattices. It is sufficient for grasping the idea of the reduction of [Lee+19].*

There are also module lattices that are not free modules – they have a pseudo-basis *instead of an ordinary basis, as explained in Section 3.4; these module lattices are omitted in this discussion for now. See for example [Coh12] for more information on pseudo-bases.*

8.2.3 LLL-revisited

In Sections 2.4 to 2.6, the LLL algorithm and its ingredients were explained. This LLL algorithm is a *basis reduction* algorithm for bases[2] in $\mathbb{Z}^{n\times n}$; almost all such basis reduction algorithms rely on an *exact-SVP oracle in a lower dimension.* For LLL, this exact-SVP oracle is called many times in the *dimension* ***two*** *projected lattice* of the lattice of the input basis[3].

The efficiency of LLL is partially due to the fact that we have a polynomial algorithm to solve exact-SVP in rank-two lattices in $\mathbb{R}^{2\times 2}$; this algorithm is known as *Lagrange reduction* and is in essence nothing else than the extended Euclidean division algorithm.

The full reason why LLL is efficient is also due to another important procedure in the LLL algorithm, known as *size reduction.* This is to avoid so-called 'coefficient explosion'; in other words, size reduction keeps the coefficients manageably small during the LLL reduction computation.

[2]We assume $\mathbb{Z}$ as a base ring, as bases with coefficients in $\mathbb{Q}$ can always be scaled up to be in $\mathbb{Z}$.

[3]For BKZ of block size β, the reduction algorithm relies instead of solving exact-SVP in *dimension* β *projected lattices* of the input basis' lattice

Summary 29 (Ingredients of the LLL algorithm). The LLL algorithm consists of three main components, with their own purpose.

(i) A rank 2 exact-SVP oracle. In the 'ordinary' $\mathbb{Z}$-case, this can be done efficiently by Lagrange reduction.

(ii) Size reduction. This keeps the coefficient manageably small during the computation.

(iii) Projection. This allows to project the lattice onto a two-dimensional lattice and use the rank 2 exact SVP oracle.

The 'outer loop' of LLL searches for projected 2-dimensional blocks of the basis that are not Lagrange reduced; and Lagrange-reduces them; this is always followed by a 'size reduction' step.

8.2.4 ModuleLLL on module bases: Reduction from SVP in high-rank module lattices to SVP in rank 2 module lattices

Module-LLL on module bases

The exact same ingredients as in Summary 29 are needed to do LLL-reduction on module bases. We go through the different ingredients one by one.

(i) The rank 2 exact-SVP oracle "A reduction from SVP in high-rank module lattices to SVP in rank 2 module lattices" exactly means that we may *assume* that we have an exact-SVP oracle for rank 2 module lattices. So, for the remainder of this text, we will assume that we have such an oracle.

In a later part of this text, namely Section 8.2.5, we show that exact-SVP for rank 2 module lattices (of a fixed number field K) can be reduced to CVP in a special lattice $\mathcal{L}_K$ only depending on K; a lattice that resembles a log-S-unit lattice.

(ii) Size reduction and (iii) projection The notion of size reduction and *projection* uses the fact that we can *Gram-Schmidt orthogonalize* a basis, which needs an adequate *norm notion* on the column vectors of a module basis.

So, suppose $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_{r'})$ is a module lattice basis. In other words, each $\mathbf{b}_j \in R^{r'}$ is a column vectors consisting of r' entries from the number ring R. Then are essentially two 'size notions' on a vector; one is the *(Minkowski) geometric norm*, the other is the *algebraic norm*.

Minkowski geometric norm The Minkowski geometric norm is defined via the concatenated Minkowski embedding. Recall the Minkowski embedding $R \to K_{\mathbb{R}}$, where each element $\alpha \in R$ is mapped to the vector $(\sigma(\alpha))_\sigma$ for each embedding $\sigma : K \to \mathbb{C}$ into the complex numbers. The *geometric norm* $\|\alpha\|$

of such an element $\alpha \in R$ is then defined as the geometric norm of the vector $(\sigma(\alpha))_\sigma$. Concretely,

$$\|\alpha\|^2 = \sum_{\sigma : K \to \mathbb{C}} |\sigma(\alpha)|^2,$$

with its associated inner product, for $\alpha, \beta \in R$.

$$\langle \alpha, \beta \rangle = \sum_{\sigma : K \to \mathbb{C}} \sigma(\alpha)\overline{\sigma(\beta)}$$

The Minkowski geometric norm on a R-vector $\mathbf{b} = (\beta_1, \ldots, \beta_{r'})^T \in R^{r'}$ is then defined by the following rule

$$\|\mathbf{b}\|_K^2 = \sum_{j=1}^{r'} \sum_{\sigma : K \to \mathbb{C}} |\sigma(\beta_j)|^2 = \sum_{j=1}^{r'} \|\beta_j\|^2.$$

which again defines an inner product in the following sense, for $\mathbf{c} = (\gamma_1, \ldots, \gamma_{r'})^T \in R^{r'}$:

$$\langle \mathbf{b}, \mathbf{c} \rangle_K := \sum_{j=1}^{r'} \sigma(\beta_j)\overline{\sigma(\gamma_j)}.$$

Algebraic norm Apart from the *geometric norm* there is also an *algebraic norm* defined on elements $\alpha \in R$ as follows:

$$N(\alpha) = |R/(\alpha)|,$$

where $|R/(\alpha)|$ is the number of elements in the quotient ring $R/(\alpha)$. This algebraic norm can then be generalized for R-vectors $\mathbf{b} = (\beta_1, \ldots, \beta_{r'})^T \in R^{r'}$ by taking the algebraic norm of the norm $\|\mathbf{b}\|_K \in K_\mathbb{R}$.

$$N(\mathbf{b}) = N(\|\mathbf{b}\|_K).$$

One can show that this norm, in some way, has a reasonably interplay with the inner product $\langle \cdot, \cdot \rangle_K$.

Gram-Schmidt orthogonalization Using this inner product, one can define a Gram-Schmidt orthogonalized basis, which in essence can be used in the same way to apply size reduction and projection. The 'Lovász' condition will need the notion of the algebraic norm.

Remark 20. *In complete generality, this is a slightly more technical to get totally right, especially in the case of pseudo-bases. In both papers [Lee+19; MS19] quite some work is dedicated to make the Gram-Schmidt orthogonalization and LLL (and BKZ) reduction work.*

The 'loop' part In the loop part, the LLL algorithm projects parts of the module basis to a rank-2 basis, and checks whether this rank 2 basis is well-reduced enough. In the 'ordinary' LLL algorithm this check is carried out by checking the *Lovász condition.* For the module case this not fundamentally different.

Summary 30 (LLL over modules). The Minkowski geometric norm in combination with the algebraic norm allows to define LLL reduction of module bases in a sound way.
We have [Lee+19, Theorem 3.9]: γ-SVP in rank 2 modules (over K) allows to solve $(2\gamma|\Delta_K|^{1/t})^{2r'-1}$-SVP in rank r' modules, where $t = [K : \mathbb{Q}]$ and Δ_K is the discriminant of K.

8.2.5 Quantumly reducing exact-SVP in rank 2 module lattices to CVP in a log-S-unit like lattice

The rank 2 SVP step in the module-LLL algorithm can be thought of as 'Euclidean division' in the number field K. Most number fields *cannot have* Euclidean division in the 'classical' way, so, the 'division' is relaxed into a slightly different requirement: Given $\alpha, \beta \in R$, we would like to find $x, y \in R$ such that

$$\|x\alpha + y\beta\| \leq c \cdot \|\alpha\|, \tag{8.1}$$

where $\|y\| \leq C$. Here, $c, C \in \mathbb{R}$ are some small constants, depending on the number field.

Essentially, Equation (8.1) boils down to finding $x, y \in R$ where $x\alpha \approx y\beta$, which, by defining an adequate *logarithmic map* $\overline{\mathrm{Log}}$, can be rephrased into

$$\overline{\mathrm{Log}}(x\alpha) \approx \overline{\mathrm{Log}}(y\beta) \Longleftrightarrow \overline{\mathrm{Log}}(x/y) \approx \overline{\mathrm{Log}}(\beta/\alpha).$$

This logarithmic map is then an 'extension' of the map $\mathrm{Log}_{(\mathrm{S})}$, used for the logarithmic S-units (where S is a set of prime ideals). Recall that $\mathrm{Log}_{(\mathrm{S})} : \mathcal{O}_{K,S}^\times \to H \times \mathbb{Z}^S$ sends $\alpha \longmapsto ((\log|\sigma(\alpha)|)_\sigma, (v_\mathfrak{p}(\alpha))_{\mathfrak{p}\in S})$, where σ ranges over the embeddings $\sigma : K \to \mathbb{C}$. In this extended Logarithm, the $\log|\sigma(\alpha)|$ is replaced by the logarithm of $\sigma(\alpha)$ *without the absolute values*, i.e., the complex logarithm, which lands into $(\mathbb{R} + i\mathbb{R})/(2\pi i\mathbb{Z})$. This yields the map

$$\overline{\mathrm{Log}} : \mathcal{O}_{K,S}^\times \to \overline{H} \times \mathbb{Z}^S,$$

where $\overline{H} := \{(x_\sigma)_\sigma \in \prod_\sigma (\mathbb{R} + i\mathbb{R})/(2\pi i\mathbb{Z}) \mid \sum_\sigma x_\sigma = 0\}$.

Essentially, this rephrases into a CVP-problem where the lattice is $\overline{\mathrm{Log}}(\mathcal{O}_{K,S}^\times) \subseteq \overline{H} \times Z^S$, and where the target is some adequate definition of $\overline{\mathrm{Log}}(\beta/\alpha)$. This definition will only consists of a $\overline{H}$-part, namely, $(\log(\sigma(\beta/\alpha)))_\sigma \in \overline{H}$; the $\mathbb{Z}^S$-part will be zero.

By heuristic reasoning and technical arguments, one can show that solving CVP in this sort-of log-S-unit lattice allows to solve exact SVP in rank 2 module lattices. Note however, that translating the Logarithm back into the original space requires taking exponents. So, in order to have a good SVP result, the CVP-distance in the log space needs to be *extremely small*, essentially optimal. The quantum part of the reduction essentially comes down to computing class groups and unit groups and logarithms therein.

For now, there is no algorithm yet that allows for computing Closest Vectors in this Logarithmic S-unit like lattices efficiently; which makes this reduction essentially a theoretical one. The running time for our current CVP algorithms are too large for this reduction to be useful in terms of finding short vectors in module-lattices.

Summary 31 (Rank 2 Module-SVP reduces to CVP in a K-specific lattice). By defining an adequate logarithmic function $\overline{\text{Log}}$ on the S-units, one can translate SVP in rank 2 module lattices over K into a CVP instance (with very small approximation factor) in a logarithmic S-unit like lattice, fixed for K. No algorithm exists yet that allows to solve such CVP instances in an efficient enough way in order for this module reduction to beat the generic lattice algorithms for SVP.

8.3 Discussion

Computational gap between rank 2 and rank 1 module lattices

For about a decade, the general consensus among lattice-based cryptographers is that there is a computational barrier between SVP in rank 1 module lattices (ideal lattices) and rank ≥ 2 module lattices. The belief in this computational gap is partially caused by the works described in this chapter: Ideal lattices are somehow 'weaker' than generic lattices[4] due to the quantum attack, but for module lattices no such attack is known. The reduction described in this section from high-rank module lattices to rank 2 module lattices suggests that there is indeed 'something happening' in between rank 1 and rank 2.

Space of lattices An additional argument for why rank 1 and rank ≥ 2 module lattices are different, is by considering the *space of all rank r' module lattices* for a fixed number field. This space turns out to be a homogeneous topological space, but for the case $r' = 1$ (ideal lattices), this space is *a compact Abelian group*, known as the *Arakelov class group.*

For $r' \geq 2$, this topological space is non-compact and not Abelian. This is not an argument of why no attack could ever happen on these module lattices, but it is an indication of a fundamental difference between those two types of lattices over a number field K.

[4]But, for small crypto-relevant approximation factors, we still have no algorithm better than the generic ones.

Bibliography

[ABD16] Martin Albrecht, Shi Bai, and Léo Ducas. “A subfield lattice attack on overstretched NTRU assumptions”. In: *Annual International Cryptology Conference*. Springer. 2016, pp. 153–178 (cit. on pp. 75, 89).

[AFG13] Martin R Albrecht, Robert Fitzpatrick, and Florian Göpfert. “On the efficacy of solving LWE by reduction to unique-SVP”. In: *International Conference on Information Security and Cryptology*. Springer. 2013, pp. 293–310 (cit. on p. 68).

[AG11] Sanjeev Arora and Rong Ge. “New algorithms for learning in presence of errors”. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2011, pp. 403–415 (cit. on p. 76).

[Agg+20] Divesh Aggarwal et al. “Slide reduction, revisited—filling the gaps in SVP approximation”. In: *Annual International Cryptology Conference*. Springer. 2020, pp. 274–295 (cit. on p. 81).

[Alb+14] Martin Albrecht et al. “Algebraic algorithms for LWE problems”. In: (2014) (cit. on p. 77).

[Alb+18] Martin R. Albrecht et al. “Estimate All the {LWE, NTRU} Schemes!” In: *Security and Cryptography for Networks*. Ed. by Dario Catalano and Roberto De Prisco. Cham: Springer International Publishing, 2018, pp. 351–367. ISBN: 978-3-319-98113-0 (cit. on pp. 38, 48, 52, 84, 90).

[Alb+19] Martin R Albrecht et al. “The general sieve kernel and new records in lattice reduction”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019, pp. 717–746 (cit. on p. 79).

[Alb+20a] Martin R Albrecht et al. “Estimating quantum speedups for lattice sieves”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020, pp. 583–613 (cit. on pp. 83, 84, 86, 87, 90).

[Alb+20b] Martin R Albrecht et al. “Faster Enumeration-based Lattice Reduction: Root Hermite Factor $k^{1/(2k)}$ in Time $k^{k/8+o(k)}$”. In: *Annual International Cryptology Conference*. Springer. 2020, pp. 186–212 (cit. on p. 80).

[Alb+21] Martin R Albrecht et al. “Lattice reduction with approximate enumeration oracles”. In: *Annual International Cryptology Conference*. Springer. 2021, pp. 732–759 (cit. on pp. 81, 89).

[Alb17] Martin R Albrecht. “On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2017, pp. 103–129 (cit. on p. 70).

[Alk+16] Erdem Alkim et al. “Post-quantum Key {Exchange—A} New Hope”. In: *25th USENIX Security Symposium (USENIX Security 16)*. 2016, pp. 327–343 (cit. on pp. 68, 70).

[Aon+16] Yoshinori Aono et al. “Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2016, pp. 789–819 (cit. on p. 81).

[App+09] Benny Applebaum et al. “Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems”. In: *CRYPTO*. 2009, pp. 595–618 (cit. on p. 45).

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. “On the concrete hardness of learning with errors”. In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203 (cit. on p. 90).

[BDF20] Koen de Boer, Léo Ducas, and Serge Fehr. “On the Quantum Complexity of the Continuous Hidden Subgroup Problem”. In: *EUROCRYPT*. Springer International Publishing, 2020, pp. 341–370. ISBN: 978-3-030-45724-2 (cit. on pp. 92, 105).

[Bec+16] Anja Becker et al. “New directions in nearest neighbor searching with applications to lattice sieving”. In: *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2016, pp. 10–24 (cit. on p. 82).

[Ber+18] Daniel J. Bernstein et al. “NTRU Prime: Reducing Attack Surface at Low Cost”. In: *Selected Areas in Cryptography – SAC 2017*. Ed. by Carlisle Adams and Jan Camenisch. Cham: Springer International Publishing, 2018, pp. 235–260. ISBN: 978-3-319-72565-9 (cit. on p. 56).

[BGJ15] Anja Becker, Nicolas Gama, and Antoine Joux. “Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search.” In: *IACR Cryptol. ePrint Arch.* 2015 (2015), p. 522 (cit. on p. 82).

[BL16] Anja Becker and Thijs Laarhoven. “Efficient (ideal) lattice sieving using cross-polytope LSH”. In: *International Conference on Cryptology in Africa.* Springer. 2016, pp. 3–23 (cit. on p. 82).

[BLS16] Shi Bai, Thijs Laarhoven, and Damien Stehlé. “Tuple lattice sieving”. In: *LMS Journal of Computation and Mathematics* 19.A (2016), pp. 146–162 (cit. on p. 82).

[Bos+18] Joppe Bos et al. “CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM”. In: *2018 IEEE European Symposium on Security and Privacy (EuroS & P).* 2018, pp. 353–367. DOI: `10.1109/EuroSP.2018.00032` (cit. on p. 38).

[BS15] J.-F. Biasse and F. Song. *A note on the quantum attacks against schemes relying on the hardness of finding a short generator of an ideal in* $\mathbb{Q}(\zeta_{2^n})$. Tech. rep. 2015-12. Revision of September 28th 2015. The University of Waterloo, 2015 (cit. on p. 92).

[BSW18] Shi Bai, Damien Stehlé, and Weiqiang Wen. “Measuring, simulating and exploiting the head concavity phenomenon in BKZ”. In: Springer, 2018, pp. 369–404 (cit. on p. 88).

[CDW17] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. “Short Stickelberger class relations and application to Ideal-SVP”. In: *EUROCRYPT.* Springer. 2017, pp. 324–348 (cit. on pp. 37, 92, 97, 99).

[CDW21] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. “Mildly Short Vectors in Cyclotomic Ideal Lattices in Quantum Polynomial Time”. In: *J. ACM* 68.2 (Jan. 2021). ISSN: 0004-5411. DOI: `10.1145/3431725`. URL: `https://doi.org/10.1145/3431725` (cit. on pp. 92, 96–98, 100).

[CGS14] Peter Campbell, Michael Groves, and Dan Shepherd. *Soliloquy: A Cautionary Tale.* ETSI 2nd Quantum-Safe Crypto Workshop. 2014 (cit. on p. 37).

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. “An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low-level encoding of zero”. In: *LMS Journal of Computation and Mathematics* 19.A (2016), pp. 255–266 (cit. on p. 75).

[CL21] André Chailloux and Johanna Loyer. “Lattice sieving via quantum random walks”. In: *International Conference on the Theory and Application of Cryptology and Information Security.* Springer. 2021, pp. 63–91 (cit. on p. 83).

[CN11] Yuanmi Chen and Phong Q. Nguyen. “BKZ 2.0: Better Lattice Security Estimates”. In: *ASIACRYPT.* 2011, pp. 1–20 (cit. on p. 88).

[Coh12] Henri Cohen. *Advanced topics in computational number theory.* Vol. 193. Springer Science & Business Media, 2012 (cit. on pp. 39, 111).

[Cra+16] Ronald Cramer et al. “Recovering short generators of principal ideals in cyclotomic rings”. In: *EUROCRYPT*. Springer. 2016, pp. 559–585 (cit. on pp. 37, 92, 98–100).

[Dac+20] Dana Dachman-Soled et al. “LWE with side information: attacks and concrete security estimation”. In: *Annual International Cryptology Conference*. Springer. 2020, pp. 329–358 (cit. on pp. 68, 69).

[DP23a] Léo Ducas and Ludo N Pulles. “Accurate score prediction for dual-sieve attacks”. In: *Cryptology ePrint Archive* (2023) (cit. on p. 71).

[DP23b] Léo Ducas and Ludo N Pulles. “Does the dual-sieve attack on learning with errors even work?” In: *Annual International Cryptology Conference*. Springer. 2023, pp. 37–69 (cit. on p. 71).

[DPW19] Léo Ducas, Maxime Plançon, and Benjamin Wesolowski. “On the Shortness of Vectors to be found by the Ideal-SVP Quantum Algorithm”. In: *CRYPTO*. Springer. 2019, pp. 322–351 (cit. on p. 92).

[DSW21] Léo Ducas, Marc Stevens, and Wessel van Woerden. “Advanced lattice sieving on GPUs, with tensor cores”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2021, pp. 249–279 (cit. on pp. 84, 86, 87).

[Duc18] Léo Ducas. “Shortest vector from lattice sieving: a few dimensions for free”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2018, pp. 125–145 (cit. on p. 83).

[Duc22] Léo Ducas. “Estimating the Hidden Overheads in the BDGL Lattice Sieving Algorithm”. In: *International Conference on Post-Quantum Cryptography*. Springer. 2022, pp. 480–497 (cit. on pp. 85, 87).

[DW21] Léo Ducas and Wessel van Woerden. “NTRU Fatigue: How Stretched is Overstretched?” In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2021, pp. 3–32 (cit. on pp. 76, 89).

[Eis+14] Kirsten Eisenträger et al. “A quantum algorithm for computing the unit group of an arbitrary degree number field”. In: *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. ACM. 2014, pp. 293–302 (cit. on pp. 37, 92, 99, 105).

[EJK20] Thomas Espitau, Antoine Joux, and Natalia Kharchenko. “On a dual/hybrid approach to small secret LWE”. In: *International Conference on Cryptology in India*. Springer. 2020, pp. 440–462 (cit. on p. 70).

[FPS22] Joël Felderhoff, Alice Pellet-Mary, and Damien Stehlé. “On Module Unique-SVP and NTRU”. In: Springer-Verlag, 2022 (cit. on pp. 37, 59).

[Gen09] Craig Gentry. “A fully homomorphic encryption scheme”. `crypto.stanford.edu/craig`. PhD thesis. Stanford University, 2009 (cit. on p. 92).

[GJ21] Qian Guo and Thomas Johansson. “Faster dual lattice attacks for solving LWE with applications to CRYSTALS”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2021, pp. 33–62 (cit. on pp. 70, 75, 89).

[GMW21] Qian Guo, Erik Mårtensson, and Paul Stankovski Wagner. “On the sample complexity of solving LWE using BKW-style algorithms”. In: *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2021, pp. 2405–2410 (cit. on p. 72).

[GN08] Nicolas Gama and Phong Q Nguyen. “Predicting lattice reduction”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2008, pp. 31–51 (cit. on pp. 67, 68).

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. “Lattice Enumeration Using Extreme Pruning”. In: *EUROCRYPT*. 2010, pp. 257–278 (cit. on p. 81).

[Gu19] Chunsheng Gu. “Integer Version of Ring-LWE and Its Applications”. In: *Security and Privacy in Social Networks and Big Data*. Ed. by Weizhi Meng and Steven Furnell. Singapore: Springer Singapore, 2019, pp. 110–122. ISBN: 978-981-15-0758-8 (cit. on p. 46).

[Hei21] Max Heiser. “Improved Quantum Hypercone Locality Sensitive Filtering in Lattice Sieving”. In: *Cryptology ePrint Archive* (2021) (cit. on p. 83).

[HK17] Gottfried Herold and Elena Kirshanova. “Improved algorithms for the approximate k-list problem in Euclidean norm”. In: *IACR International Workshop on Public Key Cryptography*. Springer. 2017, pp. 16–40 (cit. on p. 82).

[HKL18] Gottfried Herold, Elena Kirshanova, and Thijs Laarhoven. “Speed-ups and time–memory trade-offs for tuple lattice sieving”. In: *IACR International Workshop on Public Key Cryptography*. Springer. 2018, pp. 407–436 (cit. on p. 82).

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. “Analyzing blockwise lattice algorithms using dynamical systems”. In: Springer, 2011, pp. 447–464 (cit. on p. 29).

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. “NTRU: A Ring-Based Public Key Cryptosystem”. In: *ANTS*. Ed. by Joe Buhler. Vol. 1423. Lecture Notes in Computer Science. Springer, 1998, pp. 267–288. ISBN: 3-540-64657-4 (cit. on pp. 41, 56, 59).

[HR17] Michael Hamburg and Arnab Roy. “Integer Module LWE key exchange and encryption: The three bears (draft)”. In: 2017 (cit. on p. 46).

[HS07] Guillaume Hanrot and Damien Stehlé. “Improved analysis of Kannan’s shortest lattice vector algorithm”. In: *Annual international cryptology conference*. Springer. 2007, pp. 170–186 (cit. on p. 121).

[HS08] Guillaume Hanrot and Damien Stehlé. “Worst-case Hermite-Korkine-Zolotarev reduced lattice bases”. In: *arXiv preprint arXiv:0801.3331* (2008) (cit. on p. 121).

[HS10] Guillaume Hanrot and Damien Stehlé. “A complete worst-case analysis of Kannan’s shortest lattice vector algorithm”. In: *Manuscript* (2010). Full version of [HS07; HS08], pp. 1–34 (cit. on p. 80).

[KF17] Paul Kirchner and Pierre-Alain Fouque. “Revisiting lattice attacks on overstretched NTRU parameters”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2017, pp. 3–26 (cit. on p. 89).

[KL21] Elena Kirshanova and Thijs Laarhoven. “Lower bounds on lattice sieving and information set decoding”. In: *Annual International Cryptology Conference*. Springer. 2021, pp. 791–820 (cit. on p. 85).

[Laa15] Thijs Laarhoven. “Sieving for shortest vectors in lattices using angular locality-sensitive hashing”. In: *Annual Cryptology Conference*. Springer. 2015, pp. 3–22 (cit. on p. 82).

[Laa16a] Thijs Laarhoven. “Search problems in cryptography: from fingerprinting to lattice sieving”. English. PhD Thesis. PhD thesis. Mathematics and Computer Science, Feb. 2016. ISBN: 978-90-386-4021-1 (cit. on p. 83).

[Laa16b] Thijs Laarhoven. “Sieving for closest lattice vectors (with preprocessing)”. In: *CoRR* abs/1607.04789 (2016). URL: `http://arxiv.org/abs/1607.04789` (cit. on pp. 99, 103).

[Lee+19] Changmin Lee et al. “An LLL algorithm for module lattices”. In: *ASIACRYPT*. Springer. 2019, pp. 59–90 (cit. on pp. 37, 93, 109–111, 113, 114).

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra Jr., and László Lovász. “Factoring polynomials with rational coefficients”. In: *Mathematische Annalen* 261.4 (Dec. 1982), pp. 515–534 (cit. on p. 25).

[LMV15] Thijs Laarhoven, Michele Mosca, and Joop Van De Pol. “Finding shortest lattice vectors faster using quantum search”. In: *Designs, Codes and Cryptography* 77.2 (2015), pp. 375–400 (cit. on p. 83).

[LN20a] Jianwei Li and Phong Q Nguyen. “A complete analysis of the BKZ lattice reduction algorithm”. In: *Cryptology ePrint Archive* (2020) (cit. on p. 81).

[LN20b] Jianwei Li and Phong Q. Nguyen. “A complete analysis of the BKZ lattice reduction algorithm”. In: *Cryptology ePrint Archive* (2020) (cit. on p. 29).

[LS15] Adeline Langlois and Damien Stehlé. "Worst-Case to Average-Case Reductions for Module Lattices". In: *Des. Codes Cryptography* 75.3 (June 2015), pp. 565–599. ISSN: 0925-1022. DOI: 10.1007/s10623-014-9938-4. URL: https://doi.org/10.1007/s10623-014-9938-4 (cit. on p. 37).

[MAT22] MATZOV. *Report on the Security of LWE: Improved Dual Lattice Attack.* Tech. rep. 2022. DOI: https://doi.org/10.5281/zenodo.6493704 (cit. on pp. 70, 71, 75, 84, 86, 87, 89).

[Moo+20] Dustin Moody et al. *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process.* en. July 2020. DOI: https://doi.org/10.6028/NIST.IR.8309 (cit. on pp. 47, 52).

[MR09] Daniele Micciancio and Oded Regev. "Lattice-based cryptography". In: *Post-quantum cryptography.* Springer, 2009, pp. 147–191 (cit. on p. 70).

[MS19] Tamalika Mukherjee and Noah Stephens-Davidowitz. "Lattice Reduction for Modules, or How to Reduce ModuleSVP to ModuleSVP". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 1142 (cit. on pp. 37, 39, 93, 109, 110, 113).

[Nat17] National Institute of Standards and Technology. *Post-Quantum Cryptography Standardization.* 2017. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization (cit. on p. 47).

[Neu13] Jürgen Neukirch. *Algebraic number theory.* Vol. 322. Springer Science & Business Media, 2013 (cit. on p. 94).

[Ngu09] Phong Q Nguyen. "Hermite's constant and lattice algorithms". In: *The LLL Algorithm.* Springer, 2009, pp. 19–69 (cit. on p. 80).

[NS13] J. Neukirch and N. Schappacher. *Algebraic Number Theory.* Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2013. ISBN: 9783662039830. URL: https://books.google.nl/books?id=hS3qCAAAQBAJ (cit. on pp. 37, 38).

[NV08] Phong Q. Nguyen and Thomas Vidick. "Sieve Algorithms for the Shortest Vector Problem Are Practical". In: *Journal of Mathematical Cryptology* (2008). To appear. (cit. on p. 82).

[Pei09] Chris Peikert. "Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem: Extended Abstract". In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing.* STOC '09. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 333–342. ISBN: 9781605585062. DOI: 10.1145/1536414.1536461. URL: https://doi.org/10.1145/1536414.1536461 (cit. on p. 43).

[PHS19] Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. “Approx-SVP in ideal lattices with pre-processing”. In: *EUROCRYPT*. Springer. 2019, pp. 685–716 (cit. on pp. 37, 92, 99, 102, 104).

[PP19] Chris Peikert and Zachary Pepin. “Algebraically Structured LWE, Revisited”. In: *Theory of Cryptography*. Ed. by Dennis Hofheinz and Alon Rosen. Cham: Springer International Publishing, 2019, pp. 1–23. ISBN: 978-3-030-36030-6 (cit. on pp. 45, 46, 53).

[PS21] Alice Pellet-Mary and Damien Stehlé. “On the hardness of the NTRU problem”. In: Springer-Verlag, 2021. DOI: `10.1007/978-3-030-92062-3_1` (cit. on pp. 37, 59).

[PV21] Eamonn W Postlethwaite and Fernando Virdia. “On the success probability of solving unique SVP via BKZ”. In: *IACR International Conference on Public-Key Cryptography*. Springer. 2021, pp. 68–98 (cit. on pp. 68, 69).

[Rav+21] Prasanna Ravi et al. “Lattice-Based Key-Sharing Schemes: A Survey”. In: *ACM Comput. Surv.* 54.1 (Jan. 2021). ISSN: 0360-0300. DOI: `10.1145/3422178`. URL: `https://doi.org/10.1145/3422178` (cit. on p. 48).

[Reg09] Oded Regev. “On Lattices, Learning with Errors, Random Linear Codes, and Cryptography”. In: *J. ACM* 56.6 (Sept. 2009). ISSN: 0004-5411. DOI: `10.1145/1568318.1568324`. URL: `https://doi.org/10.1145/1568318.1568324` (cit. on pp. 41, 43, 44).

[Sch87] Claus-Peter Schnorr. “A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms”. In: *Theor. Comput. Sci.* 53 (1987), pp. 201–224 (cit. on p. 28).

[SG10] Michael Schneider and Nicolas Gama. *Darmstadt SVP Challenges*. `https://www.latticechallenge.org/svp-challenge/index.php`. Accessed: 16-02-2022. 2010 (cit. on p. 79).

[SH95] Claus-Peter Schnorr and Horst Helmut Hörner. “Attacking the Chor-Rivest cryptosystem by improved lattice reduction”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1995, pp. 1–12 (cit. on p. 80).

[WEB23] Andreas Wiemers, Stephan Ehlen, and Kaveh Bashiri. “A remark on the Independence Heuristic in the Dual Attack”. In: *Cryptology ePrint Archive* (2023) (cit. on p. 71).

[YD17] Yang Yu and Léo Ducas. “Second order statistical behavior of LLL and BKZ”. In: Springer, 2017, pp. 3–22 (cit. on p. 88).

Part IV

Appendix

How to use estimation scripts

1 Usage of estimator scripts

We will shortly discuss the usage of the state-of-the-art estimator scripts from https://github.com/malb/lattice-estimator/ . These scripts estimate the cost of the (hybrid) primal and dual attacks, and the cost of Arora-Ge and BKW attacks. The hybrid dual attack estimator does incorporate the latest SVP cost model from MATZOV, but does not yet contain the latest dual attack improvements of that same work.

1.1 Getting started

- Obtain and install the latest version of SageMath (https://www.sagemath.org/). Most package managers contain it.
- Obtain the lattice estimator files (https://github.com/malb/lattice-estimator/archive/refs/heads/main.zip), and extract to a folder.
- Start sage in that folder, and import the estimator scripts as follows:

[1]:
```
from estimator import *
```

2 Obtaining estimates

The security estimate of schemes is based on the attack with the lowest (time) complexity. This means that it can be hard to obtain a single formula for the hardness of a certain LWE instance. The estimation scripts for each attack can be complex, as they often have to optimize over many parameters involved in the attack.

2.1 Primal and dual attack

Let us look at the estimates for the Frodo640 parameters. The estimator contains these parameters. Recall that a small secret LWE instance with m samples, can be turned into a regular LWE instance with m+n samples, or vice versa. The estimator follows the schemes and uses the small secret way of expressing the parameters.

[2]:
```
from estimator.schemes import Frodo640
print(Frodo640)
display("Primal attack", LWE.primal_usvp(Frodo640))
display("Dual attack", LWE.dual(Frodo640))
```

```
LWEParameters(n=640, q=32768, Xs=D(σ=2.80), Xe=D(σ=2.80), m=656, tag='Frodo640')
```

```
'Primal attack'

rop: 2^166.5, red: 2^166.5, δ: 1.003473, β: 486, d: 1285, tag: usvp

'Dual attack'

rop: 2^173.1, mem: 2^104.0, m: 656, β: 506, d: 1296, : 1, tag: dual
```

These two estimates (primal_usvp, dual) are the most basic, and they correspond to the primal and the dual attack. We now consider a few more refinements. First, primal_bdd considers the primal attack that runs BKZ and then runs a final higher dimensional SVP call. This generally improves the complexity a bit, at the cost of a higher memory usage (See Section . . .). Secondly, dual_hybrid does the same for the dual attack, in addition to combining it with a hybrid guessing step. Recall that the hybrid attack is mostly useful in combination with the dual attack, as the precomputation for computing short dual vectors only has to be done once.

```
[3]: display("Primal attack (final SVP call)", LWE.primal_bdd(Frodo640))
     display("Hybrid dual attack", LWE.dual_hybrid(Frodo640))
```

```
'Primal attack (final SVP call)'

rop: 2^163.0, red: 2^162.1, svp: 2^161.9, β: 470, η: 504, d: 1293, tag: bdd

'Hybrid dual attack'

rop: 2^171.2, mem: 2^167.0, m: 656, β: 499, d: 1285, : 1, ζ: 11, tag: dual_hybrid
```

For the Frodo640 parameters the hybrid attack is not very effective, because the error/secret distribution is quite wide. Lastly, we consider the FFT improvement by GJ. This is currently hidden behind a flag. Computing this more complex estimate can take longer than usual.

```
[4]: display("Hybrid dual attack with FFT", LWE.dual_hybrid(Frodo640, fft=True))
```

```
'Hybrid dual attack with FFT'

rop: 2^169.2, mem: 2^166.3, m: 656, β: 492, t: 91, d: 1285, : 1, ζ: 11, tag:␣
 ↪dual_hybrid
```

Overall we see that the lowest time complexity is achieved by the primal attack that uses a final large SVP call. Here 'rop' represents an estimate for the needed gate count.

2.2 Arora-Ge and BKW attacks

For most schemes there are too little samples to make the Arora-Ge (and Gröbner basis extension) or the BKW attacks work efficiently. Still one can obtain an estimate for them.

```
[5]: display("Arora-Ge + Grobner bases", LWE.arora_gb(Frodo640))
     display("BKW", LWE.coded_bkw(Frodo640)) # slow
```

```
'Arora-Ge + Grobner bases'

rop: 2^inf, dreg: 2^inf, tag: arora-gb

'BKW'
```

```
rop: 2^241.1, m: 2^228.5, mem: 2^229.5, b: 15, t1: 2, t2: 20, : 14, #cod: 555,␣
↪#top: 0, #test: 56, tag: coded-bkw
```

3 Changing parameters and their influence

As a proof of concept we will change individual parameters of Frodo640 and see what kind of influence this has on the attack costs. For simplicity we will focus on the dual_hybrid attack (without FFT).

Let us first recompute the initial estimate:

[6]:
```
display("Initial estimate:", LWE.dual_hybrid(Frodo640))
```

```
'Initial estimate:'
```

```
rop: 2^171.2, mem: 2^167.0, m: 656, β: 499, d: 1285, : 1, ζ: 11, tag: dual_hybrid
```

Decreasing the length n of the secret makes the problem easier:

[7]:
```
display("Smaller secret dimension",
        LWE.dual_hybrid(Frodo640.updated(n=500)))
```

```
'Smaller secret dimension'
```

```
rop: 2^133.3, mem: 2^128.8, m: 606, β: 363, d: 1097, : 1, ζ: 9, tag: dual_hybrid
```

Alternatively, decreasing the norm of the error/secret will result in an easier problem.

[8]:
```
display("Smaller error/secret",
        LWE.dual_hybrid(Frodo640.updated(Xs=ND.DiscreteGaussian(1.4), Xe=ND.
 ↪DiscreteGaussian(1.4))))
```

```
'Smaller error/secret'
```

```
rop: 2^147.6, mem: 2^141.2, m: 656, β: 414, d: 1285, : 1, ζ: 11, tag: dual_hybrid
```

One can also use many other distributions for the secret and error. We do recommend to keep them identical. For example we can consider a ternary distribution. Due to the small entropy per coefficient the hybrid part of the attack becomes more important.

[9]:
```
display("Ternary error/secret (no hybrid)",
        LWE.dual(Frodo640.updated(Xs=ND.Uniform(-1,1), Xe=ND.Uniform(-1,1))))
display("Ternary error/secret (hybrid)",
        LWE.dual_hybrid(Frodo640.updated(Xs=ND.Uniform(-1,1), Xe=ND.
 ↪Uniform(-1,1))))
```

```
'Ternary error/secret (no hybrid)'
```

```
rop: 2^134.6, mem: 2^77.0, m: 617, β: 367, d: 1257, : 1, tag: dual
```

```
'Ternary error/secret (hybrid)'
```

```
rop: 2^128.9, mem: 2^125.1, m: 593, β: 346, d: 1200, : 1, ζ: 33, tag: dual_hybrid
```

Lastly, increasing the number of samples does not always make the problem (significantly) easier for the primal or dual attack. The attack reports to only use 754 out of the 2000 samples.

```
[10]: display("More samples",
              LWE.dual_hybrid(Frodo640.updated(m=2000)))
```

```
'More samples'

rop: 2^170.4, mem: 2^167.0, m: 754, β: 495, d: 1383, : 1, ζ: 11, tag: dual_hybrid
```

However, adding more samples, while making the secret/error small, might make the Arora-Ge and BKW attacks better.

```
[11]: display("Grobner bases, many samples",
              LWE.arora_gb(Frodo640.updated(Xs=ND.Uniform(-1,1), Xe=ND.Uniform(-1,1),␣
       ↪m=2**24)))
      display("BKW, many samples",
              LWE.coded_bkw(Frodo640.updated(Xs=ND.Uniform(-1,1), Xe=ND.Uniform(-1,1),␣
       ↪m=2**120))) # slow
```

```
'Grobner bases, many samples'

rop: 2^65.5, dreg: 4, mem: 2^65.5, t: 1, m: 2^24.0, tag: arora-gb

'BKW, many samples'

rop: 2^166.7, m: 2^153.8, mem: 2^154.8, b: 10, t1: 4, t2: 23, : 9, #cod: 536, #top:
 ↪ 0, #test: 68, tag: coded-bkw
```

4 All estimates

To finalize, there exists a command to obtain many of the estimates in one go. This is however slow, includes some less useful attacks, and does not consider all the flags (e.g. fft=True).

```
[13]: display(LWE.estimate(Frodo640))
```

```
bkw                  :: rop: 2^241.1, m: 2^228.5, mem: 2^229.5, b: 15, t1: 2,
t2: 20, : 14, #cod: 555, #top: 0, #test: 56, tag: coded-bkw
usvp                 :: rop: 2^166.5, red: 2^166.5, δ: 1.003473, β: 486, d:
1285, tag: usvp
bdd                  :: rop: 2^163.0, red: 2^162.1, svp: 2^161.9, β: 470, η:
504, d: 1293, tag: bdd
bdd_hybrid           :: rop: 2^163.0, red: 2^162.1, svp: 2^161.9, β: 470, η:
504, ζ: 0, |S|: 1, d: 1297, prob: 1, : 1, tag: hybrid
bdd_mitm_hybrid      :: rop: 2^353.8, red: 2^353.8, svp: 2^208.2, β: 485, η:
2, ζ: 0, |S|: 1, d: 1297, prob: 2^-185.3, : 2^187.5, tag: hybrid
dual                 :: rop: 2^173.1, mem: 2^104.0, m: 656, β: 506, d: 1296,
: 1, tag: dual
dual_hybrid          :: rop: 2^171.2, mem: 2^167.0, m: 656, β: 499, d: 1285,
: 1, ζ: 11, tag: dual_hybrid
```

```
{'arora-gb': rop: 2^inf, dreg: 2^inf, tag: arora-gb,
 'bkw': rop: 2^241.1, m: 2^228.5, mem: 2^229.5, b: 15, t1: 2, t2: 20, : 14, #cod:␣
 ↪555, #top: 0, #test: 56, tag: coded-bkw,
 'usvp': rop: 2^166.5, red: 2^166.5, δ: 1.003473, β: 486, d: 1285, tag: usvp,
 'bdd': rop: 2^163.0, red: 2^162.1, svp: 2^161.9, β: 470, η: 504, d: 1293, tag:␣
 ↪bdd,
 'bdd_hybrid': rop: 2^163.0, red: 2^162.1, svp: 2^161.9, β: 470, η: 504, ζ: 0, |S|:
 ↪ 1, d: 1297, prob: 1, : 1, tag: hybrid,
 'bdd_mitm_hybrid': rop: 2^353.8, red: 2^353.8, svp: 2^208.2, β: 485, η: 2, ζ: 0,␣
 ↪|S|: 1, d: 1297, prob: 2^-185.3, : 2^187.5, tag: hybrid,
 'dual': rop: 2^173.1, mem: 2^104.0, m: 656, β: 506, d: 1296, : 1, tag: dual,
 'dual_hybrid': rop: 2^171.2, mem: 2^167.0, m: 656, β: 499, d: 1285, : 1, ζ: 11,␣
 ↪tag: dual_hybrid,
 'dual_mitm_hybrid': rop: 2^1866.0, mem: 2^1865.0, m: 139, k: 315, : 2, β: 79, d:␣
 ↪166, ζ: 613, tag: dual_mitm_hybrid}
```

5 MATZOV estimate

For the MATZOV dual attack estimate one use the code attached to https://arxiv.org/pdf/2205.13983.pdf. For slightly less optimal estimates one can just use LWE.dual_hybrid.