Lattice-based cryptography

van de Pol, J.H.

Award date:
2011

Eindhoven University of Technology
Department of Mathematics and Computer Science

M. Sc. Thesis

Lattice-based cryptography

Author:
J.H. van de Pol

Supervisor:
Dr. B.M.M. de Weger

Advisors:
Ir. K.J.P.M. Poels
Dr. E. Jochemsz

July 18, 2011

Acknowledgements

I wrote this thesis during an internship at the NLNCSA. The NLNCSA supports the government in protecting sensitive information, for instance by evaluating and developing information security products, and giving advice about how to use these. First and foremost, I thank both of my advisors at the NLNCSA for their contributions to my thesis: I want to thank ir. K.J.P.M. Poels for noticing the small things that turned out to be not so small and I want to thank dr. E. Jochemsz for her endless supply of good advice and ideas. The both of them were always available to encourage me, to answer my questions or to proofread some part of my thesis I had just written. My thanks go out to everyone else at the NLNCSA as well, for the intellectually stimulating discussions and the (extremely high-level) chess games.

I also want to thank my supervisor at the university, dr. Benne de Weger, for introducing me to the field of lattices, for suggesting the subject of my thesis and for sharing his experience in cryptography. Additionally, I thank the other members of my assessment committee, prof. dr. Tanja Lange and dr. Rudi Pendavingh for their thought-provoking questions and useful comments. I would also like to thank prof. dr. Dan Bernstein for suggesting several interesting open questions to examine.

Last but not least, I want to thank my family for their continued support (not only during the writing of this thesis). I thank my father Roetert for the enjoyable bike rides and for his wisdom and good humor. I thank my mother Marie Josée for her her good care and for always being ready to help. Finally, I thank my sister Janna for the good times in Tallinn and for setting the bar high with her own thesis.

Without the aforementioned people, this thesis would not have been possible.

Contents

Chapter 1

Public Key Cryptography and Lattices

1.1 Introduction

One of the aims of cryptography is to protect information that is sent over an insecure channel. In 1976, Diffie and Hellman introduced the concept of public key cryptography, where cryptosystems are based on mathematically hard problems. Since then, several of such mathematical problems have been proposed as a basis for public key cryptography, with varying success. In 1996, Ajtai discovered that there are mathematical problems in the area of lattices that have some desirable properties with respect to cryptography. Since then, lattices have been used to construct several cryptosystems and other cryptographic applications. In this chapter, lattices are introduced and their connection with public key cryptography is examined.

First, a short introduction will be given on the concept of public key cryptography in Section 1.2. Specifically, the notions of computational security and computational complexity will be discussed. In Section 1.3 lattices are introduced and in Section 1.4 several classical lattice problems are given. Then, the process of lattice reduction is considered in Section 1.5 and some additional lattice problems that are relevant for cryptography are given in Section 1.6. Finally, Section 1.7 will show one of the earliest applications of lattices in cryptography, where lattice reduction is used to break knapsack-based cryptosystems.

1.2 Public Key Cryptography

1.2.1 Asymmetric cryptography

Until 1976, all known cryptographic systems were symmetric, in the sense that both sender and receiver of a message use the same key. Some systems, such as the one-time pad, achieved information theoretic security, which means that the system is secure even against adversaries with unlimited computing power. However, information theoretic security comes at a high cost in terms of the key length and required randomness. Furthermore, symmetric-key cryptography requires significant key-management.

In 1976, Diffie and Hellman published an asymmetric key cryptosystem known as the Diffie-Hellman key exchange [13], which could be used to securely agree on a symmetric key. They also introduced the notion of a trapdoor function, sometimes called a trapdoor one-way function. A one-way function f has the property that it is easy to compute $f(x)$ from x, but it is hard to compute x from $f(x)$. A trapdoor function f has the additional property that there is some secret information y, such that given y, it is easy to compute x from $f(x)$. These functions are the basis of asymmetric key cryptography.

Diffie and Hellman proposed something they called Public Key Cryptography. The idea is to create a pair of keys that are related mathematically, consisting of a private key and a public key. One possible application is that the sender of a message can use the public key of the receiver to encrypt his message and the receiver can then use his private key to decrypt the message. Everyone can have the public key without compromising the security of the corresponding private key, because deriving the private key from the public information is as hard as inverting a one-way function. Thus, every user of a system can publish his public key for all other users of the system. This provides a major advantage over symmetric cryptography, where a unique symmetric key pair has to be created and distributed for each pair of users wanting to communicate privately.

In their paper, Diffie and Hellman noted that public key cryptosystems would never attain security in the information theoretic sense, because the public and private keys are always mathematically related. They mentioned that, since the public information would always uniquely determine the secret information among the members of a finite set, an adversary with unlimited computation power would always be able to retrieve the secret information using an exhaustive search. Public key cryptosystems aimed for a more practical approach to security instead, as opposed to the security against some theoretical adversary with unlimited computing power.

1.2.2 Computational security

Rather than information theoretic security, public key cryptosystems and most symmetric cryptosystems aim to achieve security while taking the limits of computational power of adversaries into account. This is known as *computational security*, and it means that an adversary with bounded computation power cannot feasibly break the system. Thus, breaking the system should not be impossible, but rather computationally infeasible. This has led to the practice of basing the security of (public key) cryptosystems on the assumption that some mathematical problem (such as breaking the encryption) is computationally infeasible. Two good examples of cryptosystems based on such mathematical problems are RSA and ElGamal.

RSA

RSA is based on the observation that it is easy to multiply numbers, but hard to factor composite numbers. A private key in the RSA system consists of two large secret primes p and q, and a secret exponent d that is invertible mod $(p-1)(q-1)$. The public key consists of the composite number $N = pq$ and the public exponent $e = d^{-1} \bmod (p-1)(q-1)$. Messages m are represented by numbers in the ring $\mathbb{Z}/N\mathbb{Z}$, and they can be encrypted by exponentiating using the public exponent $c = m^e \bmod N$.

Note that $\phi(N) = (p-1)(q-1)$, where ϕ is Euler's totient function. In the ring $\mathbb{Z}/N\mathbb{Z}$, exponents can be taken modulo $\phi(N)$, since $a^{\phi(N)} \equiv a \bmod N$ for all invertible $a \in \mathbb{Z}/N\mathbb{Z}$. Therefore, the receiver of the ciphertext c can use the private key to compute

$$c^d = m^{ed} = m \pmod N,$$

because $e = d^{-1} \bmod \phi(N)$, and the message m is retrieved.

If an adversary is able to factor N, he can retrieve the secret primes p and q, and hence knows $\phi(N)$. He can then use $\phi(N)$ to invert the public exponent, thus retrieving the secret exponent. Note that this attack is sufficient to break the system, but not necessary. It may be that there exist attacks that manage to break the encryption or retrieve the secret exponent without learning the factorization of N. Although such attacks are not known at this time, this means that a stronger assumption is necessary to get computational security. Indeed, this assumption must also imply that N cannot feasibly be factored by an adversary. The *RSA-assumption* says that it is intractable to derive the message m from the public key (N, e) and the corresponding ciphertext $c = m^e \bmod N$. Under the RSA-assumption, the RSA cryptosystem is secure.

ElGamal

ElGamal encryption is based on the fact that it is easy to exponentiate in a finite cyclic group, but hard to compute the discrete logarithm. Hence, the underlying mathematical structure of the ElGamal system is the finite cyclic group $\mathbb{Z}_p$ for p prime. Let g be a generator of $\mathbb{Z}_p$. The private key will be a secret integer $1 < x < p-1$. The public key consists of $h = g^x \mod p$, the generator g and the prime p. To encrypt a message m, represented by a group element, the sender takes a random integer y that is coprime to $p-1$ as a blinding factor and computes

$$a = g^y \pmod p,$$
$$b = mh^y \pmod p,$$

and sends the pair (a, b) as the ciphertext. The receiver now computes

$$b/a^x = mh^y(g^{yx})^{-1} = mg^{xy}(g^{yx})^{-1} = m \pmod p,$$

retrieving the message m.

An adversary could try to retrieve the secret exponent x. If he is able to take the discrete logarithm in the group G, he can compute $x = \log_g(h)$ to find the private key. However, this attack is once again sufficient but not necessary. The adversary could also try to derive g^{xy} from g, g^x and g^y, which is either public information or sent over possibly insecure channels. The assumption that the adversary cannot derive g^{xy} from g, g^x and g^y is known as the *computational Diffie-Hellman assumption*. Under the computational Diffie-Hellman assumption, the ElGamal cryptosystem is secure.

These two examples show that even though a cryptosystem makes use of the intractability of certain problems, breaking the system is not always equivalent to solving these problems. Furthermore, the difficulty of the actual problems is not always well known, and hence the security of the system is not easily proven. Note that the lack of a proof of security for these systems does not imply that they are insecure, however. It merely shows that the security is not yet perfectly understood. In a practical sense, the security of these systems is based on the performance of the best known attacks against these systems.

An important variable for such systems is the *security parameter*. This security parameter is some variable in the cryptosystem that determines the "input size" of the system. For many systems, such as RSA, the security parameter is the key length in bits. Increasing the security parameter will generally increase the security of a cryptosystem, but it will decrease its efficiency as well, i.e., the keys will be longer and decryption and encryption may take longer. The efficiency of a cryptosystem is often considered with respect to the security parameter. The security parameter is chosen such that it is computationally infeasible to break the cryptosystem, when taking the limits of computation into account.

The *security* of a system gives a measure of this computational infeasibility. It is most commonly measured in bits, in which case it is called *bit security*. For a bit security of k bits, the "effort" that an adversary must invest into the system is equivalent to trying out all possible keys of length k bits, which leads to 2^k different keys. Thus, the adversary must put in an "effort" of at least 2^k to break the system. Here, effort is a vague term that includes time (possibly measured in number of operations on a computer), but can include things such as computer memory or other computer resources as well. Furthermore, probabilistic attacks on a system with security parameter k should not succeed with a probability higher than 2^{-k}.

It would be ideal if the 'best' attack on the cryptosystem consists of trying all possible private keys. To achieve this, other ways of breaking the system should be computationally infeasible. But when are mathematical problems such as 'breaking the encryption' computationally infeasible?

1.2.3 Complexity theory and cryptosystems

This is where the two disciplines of computational complexity theory and the analysis of algorithms come in. Computational complexity theory aims to divide computational problems in several

difficulty classes, whereas the analysis of algorithms focuses on improving algorithms (to solve these problems) in terms of running time and storage space. The following is an informal description of computational complexity theory and is by no means an extensive introduction into the field. The aim of this description is merely to provide some intuition on the notion of difficult problems. For a more detailed introduction to complexity theory, see [46] by Johnson or any undergraduate level textbook on the subject.

Two notable complexity classes from computational complexity theory are P, for polynomial, and NP, for non-deterministic polynomial. The class P consists of all problems that can be solved by a Deterministic Turing Machine in an amount of time that is bounded from above by a function that is polynomial in the length of its input. The class NP consists of all problems that are solvable by a Non-Deterministic Turing Machine, again with a time that is bounded by a function polynomial in the length of the input.

The class P is contained in the class NP. An important open question in complexity theory is whether these classes are in fact the same, or whether there exist problems in NP that are not in P. Informally, P is the class of "easy" problems, whereas NP might also contain some "hard" problems that are not in P. Karp introduced polynomial-time Karp reductions in [48], which made it possible to reduce one problem to another problem. Such reductions assume that there is some subroutine that solves the other problem, which can then be used to solve the original problem. These subroutines are called *oracles*. Because the reduction is polynomial-time, reducing a problem A to another problem B shows that if B is in P, then A must be as well. Intuitively, this means that problem A cannot be "harder" to solve than B, as solving B immediately gives a solution for A. Karp also showed that there was a subclass of NP, which he called NP-complete, such that for any problem in that class, all problems in NP can be reduced to that particular problem. This means that showing that any of these NP-complete problems is actually in P would immediately prove that $P = NP$.

The question whether $P = NP$ remains open, but many years of research have gone into it, and the conjecture is that $P \neq NP$. Until it is shown that $P = NP$, NP-complete problems are considered "hard" to solve. Therefore, if one was able to prove that (mathematically) breaking a cryptosystem is equivalent to solving some problem that is NP-complete, this should give a reasonable measure of security. Such proofs are called *security proofs*.

Unfortunately, it is not always straightforward to base a cryptosystem on a difficult mathematical problem. Although the problem is hard to solve, which provides one-wayness, the holder of the private key should still be able to retrieve the message by using some trapdoor. It is not always easy to incorporate such a trapdoor in hard problems. Furthermore, the notion of hard problems is based on worst-case analysis alone, whereas cryptography is affected by typical computational properties and costs. In other words: not every instance of a hard problem is necessarily hard, and a cryptographic system based on a subclass of a hard problem will not be secure if the particular subclass turns out to be easy to solve. A good example that suffered from these problems is the case of knapsack-based cryptosystems, based on a variant of the knapsack problem. Certain restrictions on this hard problem created some practical vulnerabilities. These vulnerabilities could be exploited using lattices, as will be shown after the introduction of lattices in Section 1.7.

Some problems have the property that not just worst-case instances are hard to solve. By showing that any worst-case instance can be reduced to some random instance, it follows that an average-case instance of the problem is at least as hard as any instance. This is also known as *random self-reducibility*. In 1996, Ajtai [1] showed that it was possible to use randomized reductions to establish a worst-case and average-case connection between certain lattice problems. This generated a lot of interest in lattice problems as a basis for cryptosystems, because average-case hardness is a desirable property for cryptosystems.

1.3 Lattices

This section is intended to give the definition and several properties of lattices. It is based on introductions to lattices by Micciancio [64], Nguyen [75] and Lenstra [57], although a few proofs

have been added. First some notation that is useful when talking about lattices will be introduced. This notation will then be used to formally define lattices. Afterwards, some distinctive properties of lattices will be given, as well as three natural ways to derive lattices from other lattices. Finally, a process called Gram-Schmidt orthogonalization is described, because it plays a role in the analysis of lattice reduction.

1.3.1 Notation

Let $\mathbb{R}^n$ be the n-dimensional Euclidean vector space with its usual topology. Bold letters denote column vectors, and the Euclidean inner product and corresponding norm are denoted by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i,$$
$$\|\mathbf{x}\| = \sqrt{x_1^2 + \ldots + x_n^2},$$

where $\mathbf{x} = (x_1, \ldots, x_n)^T$ and $\mathbf{y} = (y_1, \ldots, y_n)^T$ are vectors in $\mathbb{R}^n$. The distance between two vectors is defined as $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$, and the distance between a vector $\mathbf{x} \in \mathbb{R}^n$ and a subset $E \subset \mathbb{R}^n$ is defined as

$$\operatorname{dist}(\mathbf{x}, E) = \min_{\mathbf{y} \in E} \{d(\mathbf{x}, \mathbf{y})\}.$$

The unit vectors are denoted by $\mathbf{e}_1, \ldots, \mathbf{e}_n$, i.e., $\mathbf{e}_i$ is the vector with its i'th coordinate equal to one and all other coordinates equal to zero.

Let $\lfloor . \rfloor : \mathbb{R} \to \mathbb{Z}$ denote the floor function, which sends each real number x to the biggest integer smaller than or equal to x, and let $\lceil . \rceil : \mathbb{R} \to \mathbb{Z}$ denote the ceiling function, which sends each real number x to the smallest integer greater than or equal to x. Denote by $\lfloor . \rceil : \mathbb{R} \to \mathbb{Z}$ the round function, which is the function that sends each real number to the closest integer (rounding up if this is not unique). Furthermore, for vectors $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$, define $\lfloor \mathbf{x} \rceil = (\lfloor x_1 \rceil, \ldots, \lfloor x_n \rceil)^T$ (and define similar notation for the floor and ceiling functions).

The open ball of radius $r > 0$ centered at $\mathbf{x}$ is denoted by

$$\mathcal{B}(\mathbf{x}, r) = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{y}\| < r\}.$$

When $\mathbf{x} = \mathbf{0}$ and $r = 1$, this is the n-dimensional unit ball $\mathcal{B}(\mathbf{0}, 1)$. Its volume is denoted by

$$v_n = \frac{\pi^{n/2}}{\Gamma(n/2 + 1)}, \tag{1.1}$$

where $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ is the Gamma function, the extension of the factorial function to the real (and complex) numbers. Furthermore, let $\overline{\mathcal{B}(\mathbf{x}, r)}$ denote the closure of $\mathcal{B}(\mathbf{x}, r)$.

For any subset $S \subseteq \mathbb{R}^n$ of vectors, the linear span of S, or $\operatorname{span}(S)$, is the minimal subspace of $\mathbb{R}^n$ that contains S (or equivalently all linear combinations of the vectors in S). For any finite subset $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\} \subset \mathbb{R}^n$, let $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ be the set of all linear combinations of the $\mathbf{b}_i$'s with integral coefficients:

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m) := \left\{ \sum_{i=1}^{n} \lambda_i \mathbf{b}_i : \lambda_1, \ldots, \lambda_m \in \mathbb{Z} \right\}.$$

The half open parallelepiped spanned by the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$ is denoted by

$$\mathcal{P}(\mathbf{b}_1, \ldots, \mathbf{b}_m) = \left\{ \sum_{i=1}^{n} \lambda_i \mathbf{b}_i : 0 \leq \lambda_i < 1 \right\}.$$

Furthermore, the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{R}^n$ are called *linearly dependent* if there exist $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$, which do not all equal zero, such that

$$\sum_{i=1}^{m} \lambda_i \mathbf{b}_i = \mathbf{0},$$

and *linearly independent* otherwise.

For a subspace $E \subset \mathbb{R}^n$, denote its orthogonal complement by

$$E^{\perp} = \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{y} \rangle = 0, \text{ for all } \mathbf{y} \in E\}.$$

Now, let π_E denote the projection on the orthogonal complement $E^{\perp}$ of E. This is the unique linear map $\pi_E : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\begin{aligned} \pi_E(\mathbf{x}) &= \mathbf{x}, \quad \text{for all } \mathbf{x} \in E^{\perp}, \text{ and} \\ \pi_E(\mathbf{x}) &= \mathbf{0}, \quad \text{for all } \mathbf{x} \in E. \end{aligned}$$

Note that π_E is indeed a projection, since $\pi_E^2 = \pi_E$. For all $\mathbf{x} \in \mathbb{R}^n$, the projection can be written as $\pi_E(\mathbf{x}) = \mathbf{x} - \mathbf{y}$ for some $\mathbf{y} \in E$ (that depends on $\mathbf{x}$), due to the linearity of π_E.

Let $R^{n \times m}(R)$ be the set of $n \times m$ matrices with coefficients in the ring R. One matrix in particular is of interest: the *Gram matrix* of the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$ is the $m \times m$ matrix $\left(\langle \mathbf{b}_i, \mathbf{b}_j \rangle\right)_{ij}$ containing all pairwise inner products of the $\mathbf{b}_i$'s. Consider the matrix B that has the vectors $\mathbf{b}_i$ as its columns, i.e.,

$$B = \begin{pmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_m \\ | & | & & | \end{pmatrix}.$$

The Gram matrix of $\mathbf{b}_1, \ldots, \mathbf{b}_m$ can equivalently be written as $B^T B$. The determinant of the Gram matrix, denoted by $\Delta(\mathbf{b}_1, \ldots, \mathbf{b}_m) = \det(B^T B)$, is called the *Gram determinant* of the $\mathbf{b}_i$'s. The Gram determinant has several interesting properties. It is always nonnegative, and equal to zero if and only if the $\mathbf{b}_i$'s are linearly dependent. Additionally, it is invariant under any integral linear transformation of determinant ± 1, including permutation of the $\mathbf{b}_i$'s. Finally, it also has a geometric interpretation when the $\mathbf{b}_i$'s are linearly independent. In that case, $\sqrt{\Delta(\mathbf{b}_1, \ldots, \mathbf{b}_m)}$ is the m-dimensional volume of the parallelepiped $\mathcal{P}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ spanned by the $\mathbf{b}_i$'s.

1.3.2 Definition

Lattices are typically defined as a discrete subgroup of $\mathbb{R}^n$. The first step is to examine what discreteness means in this context.

Definition 1.1 (Discreteness)**.** *Let D be a subset of $\mathbb{R}^n$. Then D is called discrete when it has no accumulation points. This means that for all $\mathbf{x} \in D$, there exists an $r > 0$ such that $\mathcal{B}(\mathbf{x}, r) \cap D = \{\mathbf{x}\}$.*

Note that any subset of a discrete set is also discrete. It is now possible to define lattices:

Definition 1.2 (Lattice)**.** *A subgroup under addition of $(\mathbb{R}^n, +)$ that has the discreteness property is called a lattice.*

As any subgroup of a lattice also has the discreteness property by definition, it will itself be a lattice. Let $G \subseteq \mathbb{R}^n$ be an additive subgroup. If G has an accumulation point $\mathbf{x} \in G$ with an accompanying sequence $(\mathbf{x}_n)_{n=1}^{\infty} \subset L$ converging to $\mathbf{x}$, then the sequence $(\mathbf{x} - \mathbf{x}_n)_{n=1}^{\infty}$ is in G due to closure under addition. As this sequence converges to $\mathbf{0} \in G$, any additive subgroup of $\mathbb{R}^n$ has an accumulation point if and only if $\mathbf{0}$ is an accumulation point. Therefore, it is also possible to define lattices as nonempty subsets $L \subseteq \mathbb{R}^n$, closed under subtraction, such that there exists an $r > 0$ such that $\mathcal{B}(\mathbf{0}, r) \cap L = \{\mathbf{0}\}$. Due to the combination of their group structure and discreteness, lattices have the following properties:

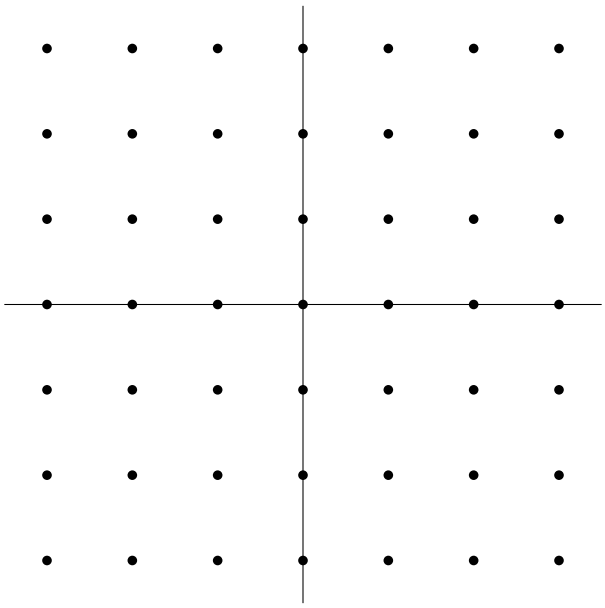

Figure 1.1: The lattice $\mathbb{Z}^n$, consisting of all vectors in $\mathbb{R}^2$ with integral coefficients.

Lemma 1.3. *Let $L \subset \mathbb{R}^n$ be a lattice. Then the following properties hold:*

(i) There exists an $r > 0$ such that for all $\mathbf{x} \in L$, there is no other element $\mathbf{y} \in L$ with $\|\mathbf{x} - \mathbf{y}\| < r$. Equivalently, there exists an $r > 0$ such that $\mathcal{B}(\mathbf{x}, r) \cap L = \{\mathbf{x}\}$ for all $\mathbf{x} \in L$.

(ii) L is a closed set, which means that L has no accumulation points outside of L (hence none at all).

(iii) If $S \subseteq \mathbb{R}^n$ is bounded, then $L \cap S$ is finite.

(iv) L is countable.

Although this gives a formal definition of a lattice, it does not show how lattices can be represented. The next step in working with lattices is to find a way to represent them.

1.3.3 Bases

In vector spaces, subspaces can be described using *bases*. It is also possible to define bases for lattices. Take m vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{R}^n$. Consider the set of integral linear combinations of these vectors, $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$. This is a subgroup of $\mathbb{R}^n$, but not necessarily discrete. For example, the set $\mathcal{L}((1), (\sqrt{2})) \subset \mathbb{R}$ is not discrete, because $\sqrt{2} \notin \mathbb{Q}$, which means it is possible to construct a sequence $(a_n \cdot 1 - b_n \cdot \sqrt{2})$ that converges to zero as $n \to \infty$. The following proposition gives sufficient conditions for discreteness:

Proposition 1.4. *The subgroup $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m) \subset \mathbb{R}^n$ is discrete (and therefore a lattice) if either*

(i) $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{Q}^n$, or

(ii) $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{R}^n$ are linearly independent.

Proof. The first case is trivial, as it is possible to compute the least common multiple l of the denominators of all vector entries, and the distance between two vectors is always at least $1/l$. For

the second case, let $L = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ and consider the parallelepiped

$$P = \left\{ \sum_{i=1}^{m} \lambda_i \mathbf{b}_i : |\lambda_i| < 1 \right\}.$$

Now take $\mathbf{y} \in L \cap P$. Since $\mathbf{y} \in L$, it is a linear combination of the $\mathbf{b}_i$'s with integral coefficients, but since $\mathbf{y} \in P$, it is also a linear combination of the $\mathbf{b}_i$'s with real coefficients in the open interval $(-1, 1)$:

$$\mathbf{y} = \sum_{i=1}^{m} \lambda_i \mathbf{b}_i, \quad \text{where } \lambda_i \in \mathbb{Z}, \text{ and}$$
$$\mathbf{y} = \sum_{i=1}^{m} \mu_i \mathbf{b}_i, \quad \text{where } \mu_i \in (-1, 1).$$

Subtracting $\mathbf{y}$ from itself gives the zero vector:

$$\mathbf{0} = \mathbf{y} - \mathbf{y} = \sum_{i=1}^{m} (\lambda_i - \mu_i) \mathbf{b}_i.$$

Because the $\mathbf{b}_i$'s are linearly independent, $\lambda_i - \mu_i = 0$ for all $1 \leq i \leq m$. But since λ_i is integral for all i and μ_i is in the open interval $(-1, 1)$, they must be zero. Hence, $\mathbf{y} = \mathbf{0}$, and thus $L \cap P = \{\mathbf{0}\}$. Since there exists an $r > 0$ such that $\mathcal{B}(\mathbf{0}, r) \subset P$ (e.g. $r = \frac{1}{2} \min\{\|\mathbf{b}_1\|, \ldots, \|\mathbf{b}_n\|\}$), $\mathbf{0}$ cannot be an accumulation point. This completes the proof. □

The fact that a finite set of vectors can describe a lattice, much like a finite set of vectors can describe a subspace of $\mathbb{R}^n$, gives rise to the following definition:

Definition 1.5. *A lattice L is said to be spanned by the generators $\mathbf{b}_1, \ldots, \mathbf{b}_m$ if $L = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$. If the $\mathbf{b}_i$'s are linearly independent as well, then $(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ is called a basis of L and for every $\mathbf{x} \in L$ there are unique integral coordinates $(x_1, \ldots, x_m)$ such that*

$$\mathbf{x} = \sum_{i=1}^{m} x_i \mathbf{b}_i.$$

Bases are useful when representing lattices, but they are generally not unique. Compare for instance Figures 1.2 and 1.3 to see two different bases of the lattice $\mathbb{Z}^n$. For a basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ of a lattice L, the matrix representation of this basis is defined as the matrix B that has the vectors $\mathbf{b}_i$ as its columns:

$$B = \begin{pmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_m \\ | & | & & | \end{pmatrix}.$$

For any basis matrix B, define $\mathcal{L}(B) = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$, $\mathcal{P}(B) = \mathcal{P}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ and $\text{span}(B) = \text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$. In the following, the word *basis* will be used interchangeably for both $\mathbf{b}_1, \ldots, \mathbf{b}_m$ and their corresponding basis matrix B.

Linear subspaces of $\mathbb{R}^n$ have a dimension, which signifies the number of vectors in a basis for the subspace, as well as the maximal number of independent vectors in this subspace. This definition is used to define the rank of a lattice.

Definition 1.6. *Let $L \subset \mathbb{R}^n$ be a lattice. The rank d of L is defined to be the dimension of the subspace* $\text{span}(L)$*, which is the minimal subspace containing L. If $n = d$, the lattice is called full-rank.*

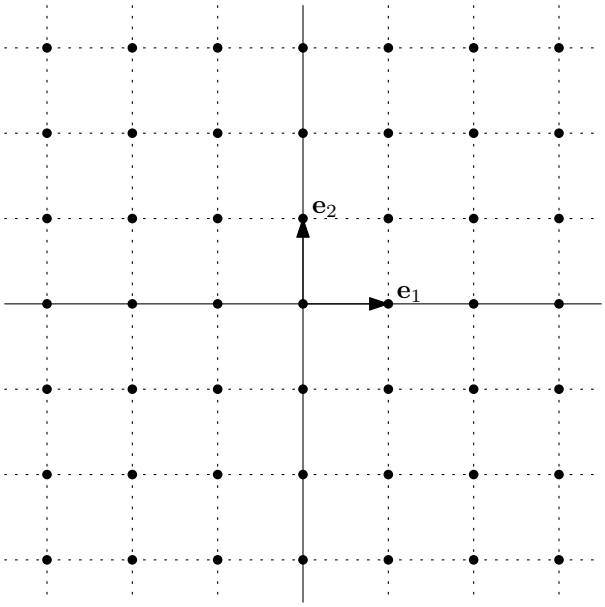

Figure 1.2: The lattice $\mathbb{Z}^n$ with two basis vectors $\mathbf{e}_1 = (1, 0)$ and $\mathbf{e}_2 = (0, 1)$.

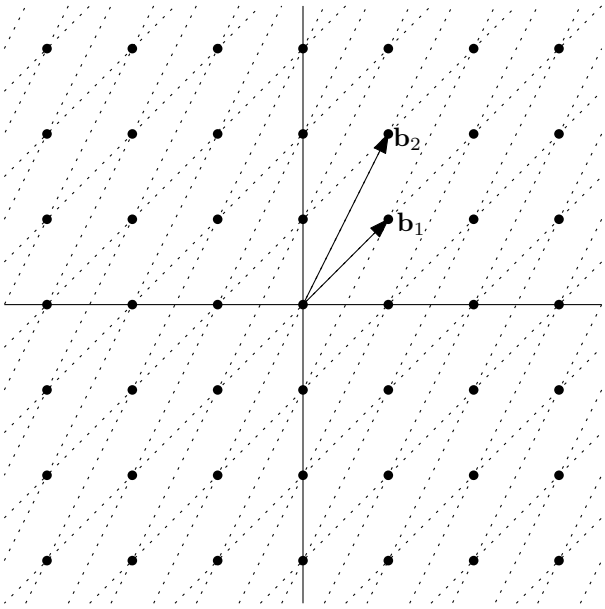

Figure 1.3: The lattice $\mathbb{Z}^n$, now with the different basis $\mathbf{b}_1 = (1, 1)$ and $\mathbf{b}_2 = (2, 1)$.

The words rank and dimension are sometimes used interchangeably for the rank of a lattice. To prevent confusion between the dimension of the space and the rank of the lattice, the word dimension will not be used to describe the rank of a lattice here. As with subspaces, the rank of a lattice is equal to the maximal number of linearly independent vectors in the lattice, as well as the number of vectors in a basis of L. Note that unlike in subspaces, not all maximal sets of linear independent vectors form a basis. The following proposition characterizes when independent vectors in the lattice form a basis.

Proposition 1.7. *Let $L \subset \mathbb{R}^n$ be a d-rank lattice, and let $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be d independent vectors in L, with matrix representation B. Then, B is a basis of L if and only if there are no non-zero lattice vectors $\mathbf{x} \in L$ such that $\mathbf{x} \in \mathcal{P}(B)$.*

Proof. ($\Longrightarrow$) Assume B is a basis of L and consider lattice vector $\mathbf{x} \in L$. Since B is a basis of L consisting of d independent vectors, $\mathbf{x}$ can be written as a unique linear combination of the $\mathbf{b}_i$'s with integral coefficients:

$$\mathbf{x} = \sum_{i=1}^{d} \lambda_i \mathbf{b}_i, \quad \lambda_i \in \mathbb{Z}.$$

If $\mathbf{x} \in \mathcal{P}(B)$, $0 \leq \lambda_i < 1$ by definition, and thus $\lambda_i = 0$ for all $1 \leq i \leq d$. Therefore, the only lattice point $\mathbf{x} \in \mathcal{P}(B)$ is the zero vector.

($\Longleftarrow$) For the other direction, it is shown that if B is not a basis, there must be a nonzero lattice vector $\mathbf{x} \in L$ such that $\mathbf{x} \in \mathcal{P}(B)$. Assume B is not a basis of L. There must be a lattice vector $\mathbf{y} \in L$ such that $\mathbf{y}$ is not a linear combination of the $\mathbf{b}_i$'s with integral coefficients. Since $\mathbf{b}_1, \ldots, \mathbf{b}_d$ are d linearly independent vectors in L, $\text{span}(L) = \text{span}(B)$. Thus, it is possible to write $\mathbf{y}$ as a linear combination of $\mathbf{b}_1, \ldots, \mathbf{b}_d$

$$\mathbf{y} = \sum_{i=1}^{d} \lambda_i \mathbf{b}_i,$$

where at least one λ_i is not integral. Now consider the vector

$$\mathbf{y}' = \sum_{i=1}^{d} \lfloor \lambda_i \rfloor \mathbf{b}_i.$$

As $\mathbf{y}'$ is an integral linear combination of d lattice vectors, it must itself again be a lattice vector. Since L is an additive subgroup, the difference between the lattice vectors $\mathbf{y}$ and $\mathbf{y}'$ is a lattice vector as well. This vector is of the form

$$\mathbf{x} = \mathbf{y} - \mathbf{y}' = \sum_{i=1}^{d} (\lambda_i - \lfloor \lambda_i \rfloor) \mathbf{b}_i.$$

Because $0 \leq \lambda_i - \lfloor \lambda_i \rfloor < 1$, the vector $\mathbf{x} \in \mathcal{P}(B)$. Furthermore, $\mathbf{x}$ cannot be the zero vector, since there is at least one λ_i that is not integral and thus $\lambda_i - \lfloor \lambda_i \rfloor > 0$ for some i. Therefore, $\mathbf{x} \in L$ is a nonzero lattice vector such that $\mathbf{x} \in \mathcal{P}(B)$, as desired. This completes the proof. □

Figure 1.4 shows an example where two independent lattice vectors taken from a lattice L do not form a basis of this lattice L. It also shows that the half-open parallelepiped $\mathcal{P}(B)$ contains a lattice vector, as shown by Proposition 1.7. The following proposition shows that it is always possible to obtain a basis from maximal sets of linearly independent vectors:

Proposition 1.8. *Let $L \subset \mathbb{R}^n$ be a lattice of rank d. If $\mathbf{c}_1, \ldots, \mathbf{c}_d \in L$ are d linearly independent lattice vectors, then there exists a lower triangular $d \times d$ matrix $(u_{ij}) \in \mathbb{R}^{d \times d}$ such that the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$, given by $\mathbf{b}_i = \sum_{j=1}^{i} u_{ij} \mathbf{c}_j$ form a basis of L, i.e., they are linearly independent and $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_d) = L$.*

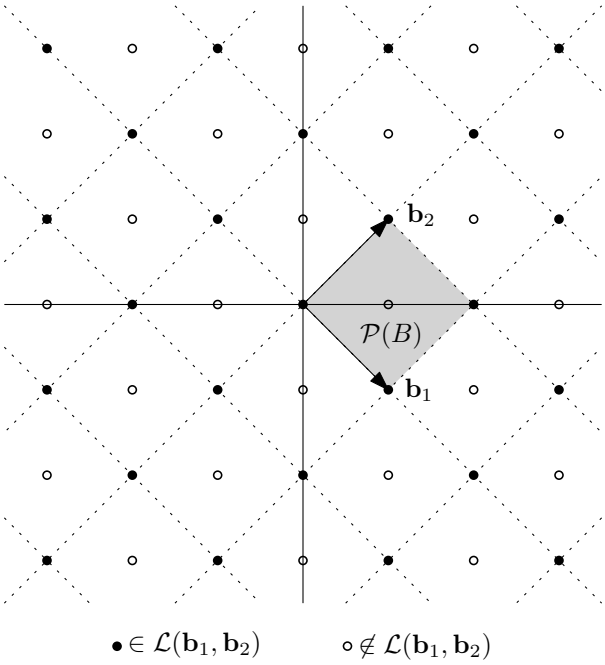

Figure 1.4: The independent vectors $\mathbf{b}_1 = (1, -1)$ and $\mathbf{b}_2 = (1, 1)$ span a lattice different from $\mathbb{Z}^2$.

Combining this proposition with the fact that every d-rank lattice has at least d independent vectors gives the following corollary:

Corollary 1.9. *Every lattice $L \subset \mathbb{R}^n$ has at least one basis.*

As a result, every d-rank lattice can be written as the integral linear combinations of d basis vectors. Furthermore, Proposition 1.4 and Corollary 1.9 provide yet another definition of lattices, i.e., as the integral linear combinations of a set of basis vectors. As mentioned, these bases are generally not unique. The following proposition shows the relation between different lattice bases:

Proposition 1.10. *Take any basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of a lattice $L \subset \mathbb{R}^n$, and any d lattice vectors $\mathbf{c}_1, \ldots, \mathbf{c}_d \in L$. Then there exists a unique square integral $d \times d$ matrix $U = (u_{ij})$ such that, for $1 \leq j \leq d$, $\mathbf{c}_j = \sum_{i=1}^{d} u_{ij}\mathbf{b}_i$, or equivalently $C = BU$. Furthermore, the $\mathbf{c}_j$'s form a basis of L if and only if the determinant of U is ± 1.*

Combining this proposition with the fact that there are infinitely many $d \times d$ matrices of determinant ± 1 when $d \geq 2$ gives that there are also infinitely many bases for each lattice when $d \geq 2$.

1.3.4 Volume

The last part of Proposition 1.10 states that any two lattice bases B and B' can be transformed into one another by multiplication with a suitable unimodular matrix U, i.e., an integral matrix of determinant ± 1. Therefore, the Gram determinant of both bases is equal (and hence for any two bases of the lattice), which suggests it might be interesting to consider the following lattice invariant:

Definition 1.11. *Let $L \subset \mathbb{R}^n$ be a d-rank lattice and let $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ be any basis. Then*

$$\mathrm{vol}(L) = \Delta(\mathbf{b}_1, \ldots, \mathbf{b}_d)^{1/2} = \sqrt{\det(B^T B)}$$

is called the volume or determinant of the lattice, which is independent of the choice of basis.

This quantity is called the volume due to the geometric interpretation of the Gram determinant: it denotes the d-dimensional volume of the parallelepiped $\mathcal{P}(B)$ spanned by the basis vectors of L. In the case that L is a full-rank lattice ($d = n$), the following properties of the volume hold:

Lemma 1.12. *Let $L \subset \mathbb{R}^n$ be a full-rank lattice. Then:*

(i) If $B \in \mathbb{R}^{n\times n}$ is a basis of L, then $\mathrm{vol}(L) = \sqrt{\det(B^T B)} = \sqrt{\det(B^T)\det(B)} = |\det(B)|$.

(ii) If $r > 0$, then

$$\lim_{r\to\infty} \frac{r^n v_n}{|\overline{\mathcal{B}(\mathbf{0}, r)} \cap L|} = \mathrm{vol}(L),$$

where v_n is defined as in (1.1).

Both these properties give rise to equivalent definitions of the volume of a full-rank lattice. Part (*ii*) of Lemma 1.12 shows that the ratio between the number of lattice vectors contained in $\overline{\mathcal{B}(\mathbf{0}, r)}$ and the volume $r^n v_n$ of this ball with radius r converges to the volume of L.

1.3.5 Quadratic forms

Historically, lattices were not studied in terms of subgroups of vector spaces, but instead in terms of quadratic forms. If $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ is a basis of a lattice $L \subset \mathbb{R}^n$, then

$$q(x_1, \ldots, x_d) = \left\| \sum_{i=1}^{d} x_i \mathbf{b}_i \right\|^2 \tag{1.2}$$

is a positive definite quadratic form over $\mathbb{R}^d$. Equivalently, given a positive quadratic form q over $\mathbb{R}^d$, Cholesky factorization of q returns d linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ satisfying (1.2) for all $\mathbf{x} = (x_1, \ldots, x_d)^T \in \mathbb{R}^d$.

1.3.6 Derived lattices

Let $L \subset \mathbb{R}^n$ be a lattice. It is possible to derive other lattices from L that are related.

Sublattices

Definition 1.13. *Let $L \subset \mathbb{R}^n$ be a lattice. A subset $M \subset L$ that is also a lattice is called a sublattice of L. Each sublattice of L must also be a subgroup of L, and each subgroup of L retains the discreteness property and is therefore a sublattice of L. M is called a full-rank sublattice of L if* $\dim(M) = \dim(L)$.

Note that unlike subspaces, a sublattice of L can have the same rank as L without being equal to L. Recall Figure 1.4, where the pictured lattice has rank 2, yet is a sublattice of $\mathbb{Z}^2$.

Now, since a sublattice M of L is also a subgroup, it is interesting to consider the quotient L/M. The number of cosets of M in L is called the group index and is denoted by $[L : M]$. The following lemma shows a relation between full-rank sublattices and a finite group index:

Lemma 1.14. *Let $L \subset \mathbb{R}^n$ be a lattice, and let M be a sublattice of L. Then M is full-rank if and only if the group index $[L : M]$ is finite, and furthermore*

$$\mathrm{vol}(M) = \mathrm{vol}(L)[L : M].$$

It is also interesting to consider sublattices that are not full-rank. In that case, their rank is strictly less than d and their group index is infinite.

Definition 1.15. *Let $L \subset \mathbb{R}^n$ be a lattice and let $M \subset L$ be a sublattice. If there is a linear subspace $E \subseteq \mathbb{R}^n$ such that $M = L \cap E$, then M is called a primitive or pure sublattice.*

Take any basis $\mathbf{B}$ of L, then $\mathbf{B} \cap E$ is a basis for M. Proposition 1.8 now implies the following lemma:

Lemma 1.16. *Let $L \subset \mathbb{R}^n$ be a lattice of rank d. A k-rank sublattice M of L is primitive if and only if every basis $\mathbf{B}$ of M can be augmented to a basis of L, by finding $d-k$ vectors that, when combined with the basis of M, form a basis of L.*

This definition can be extended to vectors, because it is possible to define lattices as the integral linear combination of a set of independent vectors. This gives rise to the following definition:

Definition 1.17. *Let $L \subset \mathbb{R}^n$ be a d-rank lattice. The vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m \in L$ are called primitive if and only if $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ is a primitive sublattice of L, i.e., if and only if there are vectors $\mathbf{b}_{m+1}, \ldots, \mathbf{b}_d$ such that $\mathbf{b}_1, \ldots, \mathbf{b}_d$ is a basis for L.*

Projected lattices

In $\mathbb{R}^n$, subspaces and their orthogonal complements can be used to define projections on these spaces. When projecting lattices on subspaces, the result is not necessarily discrete. It is possible to choose subspaces that guarantee discreteness when projecting the lattices, however. The result will then again be a lattice.

Lemma 1.18. *Let $L \subset \mathbb{R}^n$ be a d-rank lattice, and take an r-rank primitive sublattice $M \subset L$, where $1 \leq r \leq d$. Define the projection $\pi_M := \pi_{\mathrm{span}(M)}$ as the projection on the orthogonal complement of the smallest subspace containing M. Then $\pi_M(L) \subset \mathbb{R}^n$ is a $(d-r)$-rank lattice with volume $\mathrm{vol}(L)/\mathrm{vol}(M)$.*

Proof. Take a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_r)$ of M. This basis can be extended to a basis of L, since M is primitive by assumption. This gives rise to the existence of $\mathbf{b}_{r+1}, \ldots, \mathbf{b}_d \in L$ such that $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is a basis of L. Now, recall the definition of π_M and consider

$$\begin{aligned}\pi_M(L) = \pi_M(L \setminus M) &= \pi_M(\mathcal{L}(\mathbf{b}_{r+1}, \ldots, \mathbf{b}_d)) \\ &= \mathcal{L}(\pi_M(\mathbf{b}_{r+1}), \ldots, \pi_M(\mathbf{b}_d)),\end{aligned}$$

since $\pi_M(M) = \{\mathbf{0}\}$ by definition of π_M.

From the definition of π_M it also follows that for all i, there exists a vector $\mathbf{m}_i \in M$ such that $\pi_M(\mathbf{b}_i) = \mathbf{b}_i - \mathbf{m}_i$. Consider the following sum:

$$\sum_{i=r+1}^{d} \lambda_i \pi_M(\mathbf{b}_i) = \sum_{i=r+1}^{d} \lambda_i \mathbf{b}_i - \lambda_i \mathbf{m}_i.$$

As $\mathbf{m}_i \in M$, it is a linear combination of the vectors $\mathbf{b}_1, \ldots \mathbf{b}_r$, and since the basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ are linearly independent, this sum is equal to the zero vector if and only if the λ_i are all zero. Therefore, $\pi_M(\mathbf{b}_{r+1}), \ldots, \pi_M(\mathbf{b}_d)$ are linearly independent. As a result, the subgroup $\pi_M(L) = \mathcal{L}(\pi_M(\mathbf{b}_{r+1}), \ldots, \pi_M(\mathbf{b}_d))$ is a $(d-r)$-rank lattice due to Proposition 1.4. The proof of the statement on the volume of $\pi_M(L)$ becomes trivial once Gram-Schmidt vectors are introduced (see Section 1.3.7) and is omitted here. □

This lemma leads to the following corollary, which will be useful when studying lattice reduction:

Corollary 1.19. *Take a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of a lattice $L \subset \mathbb{R}^n$. Now define M_i as the sublattice spanned by the primitive basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$, for $1 \leq i \leq d$. Furthermore, define $\pi_i := \pi_{M_i} = \pi_{\mathrm{span}(\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\})}$, for $1 \leq i \leq d$. Then $\pi_i(L)$ is a $(d-i+1)$-rank lattice of volume $\mathrm{vol}(L)/\mathrm{vol}(\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}))$. Note that π_1 is the identity.*

This corollary is often used in lattice reduction, because projection reduces the rank, which allows for induction on the lattice rank. In fact, applying π_2 (the projection on the orthogonal complement of the first basis vector $\mathbf{b}_1$) reduces the rank by 1. In Euclidean vector spaces with the normal inner product, the map π_2 is defined by

$$\pi_2(\mathbf{u}) = \mathbf{u} - \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|^2} \mathbf{b}_1.$$

To see that the resulting vector is in the orthogonal complement of $\operatorname{span}(\mathbf{b}_1)$, compute the inner product

$$\langle \mathbf{u} - \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|^2} \mathbf{b}_1, \mathbf{b}_1 \rangle = \langle \mathbf{u}, \mathbf{b}_1 \rangle - \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|^2} \langle \mathbf{b}_1, \mathbf{b}_1 \rangle = \langle \mathbf{u}, \mathbf{b}_1 \rangle - \langle \mathbf{b}_1, \mathbf{u} \rangle = 0.$$

Now, let $\mathbf{u} \in L$ be a lattice vector and let $\mathbf{v} = \pi_2(\mathbf{u})$. Then $\mathbf{v}$ can be lifted back to L as follows:

$$\begin{aligned} \mathbf{u}' &= \mathbf{v} + \left(\frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|^2} - \left\lfloor \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|^2} \right\rceil \right) \mathbf{b}_1 \\ &= \mathbf{u} - \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|^2} \mathbf{b}_1 + \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|^2} \mathbf{b}_1 - \left\lfloor \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|^2} \right\rceil \mathbf{b}_1 \\ &= \mathbf{u} - \left\lfloor \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|^2} \right\rceil \mathbf{b}_1. \end{aligned} \tag{1.3}$$

Note that $\left\lfloor \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|^2} \right\rceil \in \mathbb{Z}$ and $\left| \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|} - \left\lfloor \frac{\langle \mathbf{b}_1, \mathbf{u} \rangle}{\|\mathbf{b}_1\|} \right\rceil \right| \leq 1/2$ by definition of the round function. Thus, the result is a nonzero vector $\mathbf{u}' \in L$ such that

$$\begin{aligned} \mathbf{v} &= \pi_2(\mathbf{u}'), \\ \|\mathbf{u}'\|^2 &\leq \|\mathbf{v}\|^2 + (\|\mathbf{b}_1\|/2)^2, \text{ and} \\ |\langle \mathbf{b}_1, \mathbf{u}' \rangle| &\leq \|\mathbf{b}_1\|^2/2. \end{aligned}$$

Using this procedure, it is possible to derive a somewhat short vector $\mathbf{u}'$ in L from any short vector $\mathbf{v}$ in $\pi_2(L)$. Figure 1.5 shows a 2-dimensional example of this lifting procedure, with the basis $\mathbf{b}_1 = (1, 2)$ and $\mathbf{b}_2 = (-1, 2)$. The projection $\pi_2(\mathbf{b}_2)$ is lifted back to the shorter lattice vector $\mathbf{u}' = (-2, 0)$.

Dual lattice

Besides projective lattices and sublattices, it is also possible to consider the dual lattice.

Definition 1.20. *For a lattice $L \subset \mathbb{R}^n$, the dual lattice $L^\times$ is defined as*

$$L^\times = \{\mathbf{y} \in \operatorname{span}(L) : \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}, \text{ for all } \mathbf{x} \in L\}.$$

In the case that L is full-rank, the dual lattice consists of all vectors $\mathbf{y}$ such that for every $\mathbf{x} \in L$, the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ is an integer. Let B be a basis of L. Any lattice vector can be written as $\mathbf{x} = B\mathbf{x}'$, for $\mathbf{x}' \in \mathbb{Z}^n$. Conversely, each $\mathbf{x}' \in \mathbb{Z}^n$ gives rise to a lattice vector. Given a vector $\mathbf{y}$ in the dual, the inner product becomes

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle B\mathbf{x}', \mathbf{y} \rangle = \langle \mathbf{x}', B^T\mathbf{y} \rangle \in \mathbb{Z}, \quad \text{for all } \mathbf{x}' \in \mathbb{Z}^n.$$

Now, by taking $\mathbf{x}' = \mathbf{e}_i$, where $\mathbf{e}_i$ is the i'th unit vector, it immediately follows that this inner product is integral if and only if every entry of $B^T\mathbf{y}$ is integral. Therefore, $\mathbf{y}' = B^T\mathbf{y}$ is an integral vector, and equivalently, $\mathbf{y} = (B^T)^{-1}\mathbf{y}'$ for some integral vector $\mathbf{y}'$. This proves that $L^\times \subseteq \left\{(B^T)^{-1}\mathbf{y}' | \mathbf{y}' \in \mathbb{Z}^n\right\}$. Conversely, every $\mathbf{y}' \in \mathbb{Z}^n$ gives rise to a vector in the dual, since for any $\mathbf{y}' \in \mathbb{Z}^n$ the following equality holds:

$$\langle \mathbf{x}', \mathbf{y}' \rangle = \langle \mathbf{x}', (B^T)(B^T)^{-1}\mathbf{y}' \rangle = \langle B\mathbf{x}', (B^T)^{-1}\mathbf{y}' \rangle = \langle \mathbf{x}, \mathbf{y} \rangle.$$

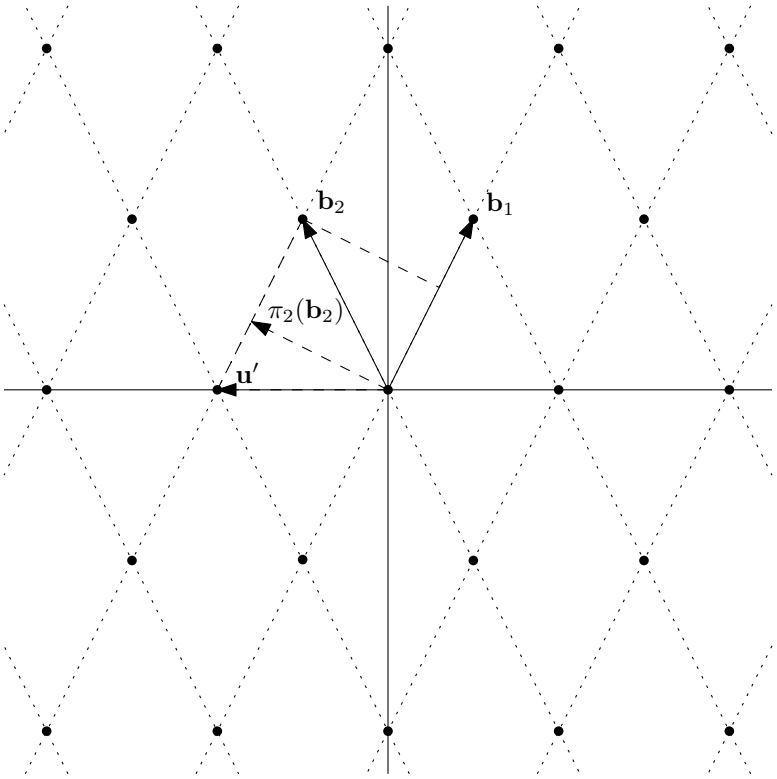

Figure 1.5: The projection $\pi_2(\mathbf{b}_2)$ is lifted to L, where $\mathbf{b}_1 = (2, -1)$ and $\mathbf{b}_2 = (2, 1)$.

This proves that $\left\{(B^T)^{-1}\mathbf{y}' : \mathbf{y}' \in \mathbb{Z}^n\right\} \subseteq L^\times$. Thus, the dual is spanned by the columns of $(B^T)^{-1}$. It immediately follows that $(L^\times)^\times = L$ and that the rank of $L^\times$ is the same as that of L. A similar argument applies when L is not full-rank, but then the dual is spanned by $B(B^TB)^{-1}$, which is the right inverse of B^T when the columns of B are independent (recall that the Gram matrix B^TB has a non-zero determinant if and only if the columns of B are independent).

Dual lattices are volume-reversing, as seen in the following lemma:

Lemma 1.21. *Let $L \subset \mathbb{R}^n$ be a lattice of rank d. Then $L^\times \subset \mathbb{R}^n$ is a d-rank lattice of volume* $\mathrm{vol}(L)^{-1}$.

Proof. As the dual lattice is a lattice that is spanned by $B(B^TB)^{-1}$, its volume is equal to $\det((B(B^TB)^{-1})^TB(B^TB)^{-1})$ by Lemma 1.12.

$$\begin{aligned}\det\left((B(B^TB)^{-1})^TB(B^TB)^{-1}\right) &= \det\left(((B^TB)^{-1})^TB^TB(B^TB)^{-1}\right)\\ &= \det\left(((B^TB)^{-1})^T\right) = \det(B^TB)^{-1}.\end{aligned}$$

By Lemma 1.12, this is equal to $\mathrm{vol}(L)^{-1}$, as desired. Furthermore, the rank of $L^\times$ is the same as that of L, since the columns of $B(B^TB)^{-1}$ are independent. □

If $L \subset \mathbb{Z}^n$ is an integral lattice of rank $d < n$, it is possible to define the orthogonal lattice, which is similar to the dual lattice of L:

Definition 1.22. *For a lattice $L \subset \mathbb{R}^n$, the orthogonal lattice is defined as*

$$L^\perp = \{\mathbf{y} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{y}\rangle = 0, \text{ for all } \mathbf{x} \in L\}.$$

Note that, unlike in the dual lattice, the vectors in the orthogonal lattice are not restricted to the span of L. The orthogonal lattice appears in the analysis of several lattice-based cryptosystems, as will be seen in Chapter 2.

1.3.7 Gram Schmidt Orthogonalization

The Gram-Schmidt process is an iterative method to orthonormalize the basis of a vector space. This is done by taking a basis vector and scaling it such that it has length one. All subsequent vectors are iteratively projected on the orthogonal complement of the span of the previous vectors and scaled to length one. When the vectors are not scaled during the procedure, the resulting basis will be orthogonal rather than orthonormal. In lattices, the projection of a lattice vector on the orthogonal complement of another lattice vector is not necessarily in the lattice, and it is generally not possible to scale a vector to length one. This means that applying the Gram-Schmidt process to a lattice basis will generally not result in a set of basis vectors that still span the same lattice. However, the Gram-Schmidt process is still interesting to consider in the case of lattices, as will be shown in Section 1.5.

Define the Gram Schmidt Orthogonalization (GSO) of a basis $\{\mathbf{b}_1, \dots \mathbf{b}_d\}$ of a lattice L using the following iterative formula:

$$\mathbf{b}_1^* := \mathbf{b}_1$$
$$\mathbf{b}_i^* := \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b}_j^*, \quad \text{where } \mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2} \text{ for all } 1 \leq j < i \leq d.$$

In order to orthonormalize this basis, all vectors $\mathbf{b}_i^*$ can be divided by their length. Figure 1.6 shows the result of applying the GSO-procedure with normalization to the basis $\mathbf{b}_1 = (1, 2)$ and $\mathbf{b}_2 = (1, 0)$. Note that applying the GSO-procedure to a lattice basis does not result in a basis for the lattice (even without normalization).

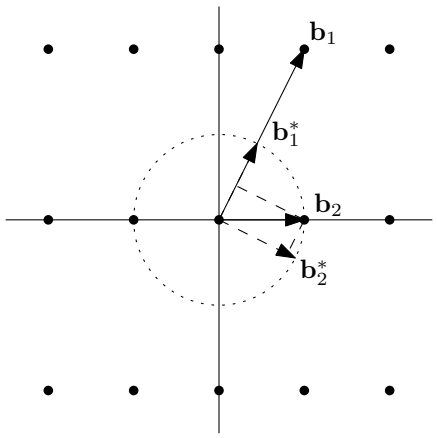

Figure 1.6: The normalized result of the GSO-procedure applied to $\mathbf{b}_1 = (1, 2)$ and $\mathbf{b}_2 = (1, 0)$.

Consider the following three matrices:

$$B = \begin{pmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_d \\ | & | & & | \end{pmatrix}, \quad B^* = \begin{pmatrix} | & | & & | \\ \mathbf{b}_1^* & \mathbf{b}_2^* & \cdots & \mathbf{b}_d^* \\ | & | & & | \end{pmatrix}, \quad \mu = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \mu_{2,1} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \mu_{d,1} & \cdots & \mu_{d,d-1} & 1 \end{pmatrix}.$$

Using these matrices, the GSO of a basis B can be described using the matrix equality $B = B^*\mu^T$. Thus, the GSO of a basis of the lattice L satisfies

$$\begin{aligned}\mathrm{vol}(L) &= \sqrt{\det(B^T B)} = \sqrt{\det(\mu(B^*)^T B^* \mu^T)} \\ &= \sqrt{\det(\mu)^2 \det((B^*)^T B^*)} = \sqrt{\det((B^*)^T B^*)}.\end{aligned}$$

The Gram matrix $(B^*)^T B^*$ is a diagonal matrix with entries $\langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle = \|\mathbf{b}_i^*\|^2$. The non-diagonal entries are zero, because the columns of B^* are pairwise orthogonal. Thus, the Gram determinant is given by $\det((B^*)^T B^*) = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|^2$. Therefore, for any basis B of the lattice L, the volume of L is given by

$$\mathrm{vol}(L) = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|, \tag{1.4}$$

where $\mathbf{b}_i^*$ are the GSO vectors of the basis B.

Note that the GSO depends on the order of the $\mathbf{b}_i$. Although the resulting vectors are generally not in the lattice (except for $\mathbf{b}_1^*$, which is in the lattice by design), an adaptation of the GSO can still be used to "reduce" a basis in a certain sense. This reduction uses the lifting procedure discussed in the section on projected lattices, as will be shown in Section 1.5, where lattice reduction is introduced.

1.4 Classical lattice problems

This section contains two important classical lattice problems. As the computational aspects of these problems are important, the actual representation of lattices becomes an issue, such as the representation of a lattice in a computer. Lattices are defined in the Euclidean vector space $\mathbb{R}^n$. However, from a computational viewpoint it is customary to consider lattices where all lattice vectors have rational coefficients. Any rational lattice can be transformed to an integral lattice by multiplying with a suitable integral factor (such as the least common multiple of all denominators). Thus, without loss of generality, lattice problems will only be defined in terms of integral lattices, such that the representation becomes a basis matrix of integers.

1.4.1 SVP

Let $L \subset \mathbb{Z}^n$ be a non-trivial lattice, i.e., it contains at least one nonzero vector $\mathbf{x}$. Now, the set $S = \overline{\mathcal{B}(\mathbf{0}, \|\mathbf{x}\|)}$ is bounded, and therefore, by Lemma 1.3, $L \cap S$ is finite. Note that $\mathbf{x} \in L \cap S$, and thus $L \cap S$ contains at least one nonzero vector. It follows that the set of norms of nonzero vectors in $L \cap S$ is finite and nonempty, and thus there exists a nonzero vector $\mathbf{u} \in L$ such that

$$\|\mathbf{u}\| = \min_{\mathbf{v} \in L \setminus \{\mathbf{0}\}} \|\mathbf{v}\|. \tag{1.5}$$

This gives rise to the *shortest vector problem* (SVP):

Definition 1.23 (Shortest Vector Problem (SVP))**.** *Given a basis B of a d-rank lattice $L \subseteq \mathbb{Z}^n$, find a nonzero vector $\mathbf{u} \in L$ such that $\|\mathbf{u}\| = \min_{\mathbf{v} \in L \setminus \{\mathbf{0}\}} \|\mathbf{v}\|$.*

Note that the shortest vector in a lattice is not unique. Indeed, if $\mathbf{u}$ is a nonzero shortest vector, then so is $-\mathbf{u}$. It is even possible to have several linearly independent shortest vectors. In the following, a "shortest vector" will always mean a nonzero vector, as $\mathbf{0}$ is trivially the shortest vector in a lattice.

In 1910, Minkowski created the *Geometry of Numbers* [73], a field intended to be a bridge between quadratic forms and Diophantine approximation. A central problem in this field was to prove the existence of short nonzero vectors in a lattice. Van Emde Boas proved in 1981 [15] that

SVP is NP-hard in the ℓ_∞ norm, but NP-hardness in the ℓ_p-norm for $p < \infty$ was not proven until Ajtai showed it in [2] for the ℓ_2 norm under randomized reductions. For most applications the shortest vector problem is defined using the ℓ_2 norm.

Often, applications do not require an actual shortest vector, but a vector that is "short enough". Hence, it is interesting to consider the *approximate shortest vector problem*:

Definition 1.24 (Approximate Shortest Vector Problem (aSVP))**.** *Given a basis B of a d-rank lattice $L \subseteq \mathbb{Z}^n$ and an approximation factor $\gamma \geq 1$, find a nonzero vector $\mathbf{u} \in L$ such that $\|\mathbf{u}\| \leq \gamma \min_{\mathbf{v} \in L \setminus \{\mathbf{0}\}} \|\mathbf{v}\|$.*

It is currently known that SVP is NP-hard (under quasi-polynomial time reductions) to approximate for factors $\gamma = 2^{\log(n)^{1/2-\varepsilon}}$, where $\varepsilon > 0$ is an arbitrarily small constant (see [49]).

1.4.2 CVP

Another problem that was studied in the language of quadratic forms is the *closest vector problem* (CVP), which can be seen as the inhomogeneous version of SVP:

Definition 1.25 (Closest Vector Problem (CVP))**.** *Given a basis B of a d-rank lattice $L \subseteq \mathbb{Z}^n$ and a target vector $\mathbf{x} \in \operatorname{span}(L)$ (not necessarily in the lattice L), find a lattice vector $\mathbf{u} \in L$ such that $\|\mathbf{x} - \mathbf{u}\| = \operatorname{dist}(\mathbf{x}, L)$.*

In 1981, van Emde Boas included it in his paper on hard lattice problems [15], where he called it the "Nearest Vector Problem". He showed that it is NP-hard to solve CVP exactly. As with SVP, it is not always necessary to retrieve the actual closest vector, but sometimes a vector that is "close enough" suffices. This gives rise to the *approximate closest vector problem*:

Definition 1.26 (Approximate Closest Vector Problem (aCVP))**.** *Given a basis B of a d-rank lattice $L \subseteq \mathbb{Z}^n$, a target vector $\mathbf{x} \in \mathbb{R}^n$ (not necessarily in the lattice L) and an approximation factor $\gamma \geq 1$, find a lattice vector $\mathbf{u} \in L$ such that $\|\mathbf{x} - \mathbf{u}\| \leq \gamma \operatorname{dist}(\mathbf{x}, L)$.*

It is known that CVP is NP-hard to approximate within any constant factor, as well as some slowly increasing (sub-polynomial) function in the lattice rank (see [6, 14]).

1.4.3 Connections between SVP and CVP

Reducing SVP to CVP

The closest vector problem has been regarded as being harder than the shortest vector problem (in the same dimension). This was first formalized by Henk [34], and later expanded by Goldreich et al. [31]. They showed, using elementary results on lattices, that an oracle that solves CVP can be used to solve SVP in the same dimension for a wide variety of norms. Furthermore, their result preserves approximation factors.

They reduce SVP to CVP as follows. Given a basis $B = [\mathbf{b}_1, \ldots, \mathbf{b}_d]$, define the basis $B^{(j)} = [\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}, 2\mathbf{b}_j, \mathbf{b}_{j+1}, \ldots, \mathbf{b}_d]$. Consider the instances $(B^{(j)}, \mathbf{b}_j)$ of the closest vector problem, which are defined by the basis $B^{(j)}$ and the target vector $\mathbf{b}_j$. For the reduction, the instances $(B^{(j)}, \mathbf{b}_j)$ are given to the CVP-oracle, which is a subroutine that solves CVP. The oracle will return the vectors $\mathbf{v}_j$, the lattice vectors in $\mathcal{L}(B^{(j)})$ that are closest to $\mathbf{b}_j$. Now, the shortest of the vectors $\mathbf{v}_1 - \mathbf{b}_1, \ldots, \mathbf{v}_n - \mathbf{b}_n$ will be a shortest vector of the lattice $\mathcal{L}(B)$.

The proof of validity for their reduction starts out with the following observation:

Proposition 1.27. *Let $B = [\mathbf{b}_1, \ldots, \mathbf{b}_d]$ be a basis for a lattice L, and let $\mathbf{u} = \sum_{i=1}^{d} \lambda_i \mathbf{b}_i$ be a shortest nonzero vector. Then, at least one of the $\lambda_i \in \mathbb{Z}$ is odd for $1 \leq i \leq d$.*

Proof. Assume the λ_i are all even, i.e., $\lambda_i = 2\lambda_i'$ for some integer λ_i'. Now consider the vector

$$\mathbf{u}' = \frac{1}{2}\mathbf{u} = \sum_{i=1}^{d} \lambda_i' \mathbf{b}_i.$$

As the λ_i''s are all integer, $\mathbf{u}'$ is a nonzero lattice vector with norm $\frac{1}{2}\|\mathbf{u}\|$. This contradicts the assumption that $\mathbf{u}$ is a shortest nonzero vector in L. □

The next proposition shows that any feasible solution to the SVP instance corresponds to a feasible solution of one of the CVP-instances. $(B^{(j)}, \mathbf{b}_j)$.

Proposition 1.28. *Let* $\mathbf{u} = \sum_{i=1}^{d} \lambda_i \mathbf{b}_i$ *be a lattice vector in* $\mathcal{L}(B)$ *such that* λ_j *is odd. Then* $\mathbf{v} = \mathbf{u} + \mathbf{b}_j$ *is a lattice vector in* $\mathcal{L}(B^{(j)})$ *such that the distance* $\operatorname{dist}(\mathbf{v}, \mathbf{b}_j) = \|\mathbf{u}\|$.

Proof. Consider the vector $\mathbf{v}$:

$$\mathbf{v} = \mathbf{u} + \mathbf{b}_j = (\lambda_j + 1)\mathbf{b}_j + \sum_{i \neq j} \lambda_i \mathbf{b}_i = \frac{\lambda_j + 1}{2}(2\mathbf{b}_j) + \sum_{i \neq j} \lambda_i \mathbf{b}_i.$$

Because λ_j is odd, $\frac{\lambda_j + 1}{2}$ is an integer, and hence $\mathbf{v}$ is in $\mathcal{L}(B^{(j)})$. Furthermore $\operatorname{dist}(\mathbf{v}, \mathbf{b}_j) = \|\mathbf{v} - \mathbf{b}_j\| = \|\mathbf{u}\|$. □

Finally, they show that any feasible solution of the CVP-instances $(B^{(j)}, \mathbf{b}_j)$ corresponds to a nonzero vector in $\mathcal{L}(B)$ as well:

Proposition 1.29. *Let* $\mathbf{v} = \lambda_j(2\mathbf{b}_j) + \sum_{i \neq j} \lambda_i \mathbf{b}_i$ *be a vector in* $\mathcal{L}(B^{(j)})$. *Then* $\mathbf{u} = \mathbf{v} - \mathbf{b}_j$ *is a* **nonzero** *lattice vector in* $\mathcal{L}(B)$.

Proof. Consider the vector $\mathbf{u}$:

$$\mathbf{u} = \mathbf{v} - \mathbf{b}_j = \sum_{i=1}^{d} \lambda_j \mathbf{b}_j - \mathbf{b}_j = (2\lambda_j - 1)\mathbf{b}_j + \sum_{i \neq j} \lambda_i \mathbf{b}_i.$$

Because $2\lambda_j - 1$ is an odd integer and hence not equal to zero, it follows that $\mathbf{u}$ is a nonzero vector in $\mathcal{L}(B)$. □

Now, let $\mathbf{u}$ be a shortest vector in $\mathcal{L}(B)$. By Proposition 1.27 it must have an odd coefficient with respect to the basis B. Thus, by Proposition 1.28, there is an instance $(B^{(j)}, \mathbf{b}_j)$ of the closest vector problem that has a solution $\|\mathbf{v}\|$ of distance at most $\|\mathbf{v} - \mathbf{b}_j\| = \|\mathbf{u}\|$ to $\mathbf{b}_j$. Therefore, each shortest vector in $\mathcal{L}(B)$ gives rise to a solution of a CVP-instance $(B^{(j)}, \mathbf{b}_j)$, such that the length of the shortest vector is equal to the distance between the target vector and this CVP solution. It follows that among all CVP-instances $(B^{(j)}, \mathbf{b}_j)$, there exists at least one solution whose distance to the target does not exceed the length of the shortest vector in $\mathcal{L}(B)$.

By Proposition 1.29, every solution $\mathbf{v}_j$ of every CVP-instance $(B^{(j)}, \mathbf{b}_j)$ gives rise to a vector $\mathbf{u}$ in the lattice $\mathcal{L}(B)$, such that the norm of $\mathbf{u}$ is the same as the distance of $\mathbf{v}_j$ to $\mathbf{b}_j$. This means that the solutions to the CVP-instances $(B^{(j)}, \mathbf{b}_j)$ cannot have a distance to their target that is smaller than the length of a shortest vector in $\mathcal{L}(B)$. Combining these results shows that the shortest of the vectors $\mathbf{v}_1 - \mathbf{b}_1, \ldots, \mathbf{v}_n - \mathbf{b}_n$ will be a shortest vector of the lattice $\mathcal{L}(B)$. Thus, SVP can be reduced to CVP using this method. This means that, in terms of computational complexity, the closest vector problem is as least as hard as the shortest vector problem.

Embedding technique

Although SVP can be reduced to CVP, there also exists a heuristic method to convert instances of CVP to an instance of SVP in a lattice of similar rank in a similar dimension. This method, known as the "embedding technique" (see [30]), allows for reasonable approximations to CVP.

The embedding technique works as follows. Take an instance of the CVP, with a basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ for the lattice L and with the target vector $\mathbf{c}$. Now construct the $(n+1)$-rank lattice L' using the columns of the following matrix as basis vectors:

$$B' = \begin{pmatrix} | & | & & | \\ \mathbf{c} & \mathbf{b}_1 & \cdots & \mathbf{b}_n \\ | & | & & | \\ 1 & 0 & \cdots & 0 \end{pmatrix}.$$

The determinant of B' is the same as the determinant of the basis B of L. Thus, the volume of the lattice L' is the same as that of L. Furthermore, the rank is nearly the same, since the rank of L is n and the rank of L' is $n+1$. The expectation is that the length of the shortest lattice vector is approximately the same in both lattices. This expectation will be made more concrete in the next section on short vectors. Now, consider the vector that is closest to $\mathbf{c}$ as an integral linear combination of the $\mathbf{b}_i$'s, $\mathbf{x} = \sum_i \lambda_i \mathbf{b}_i$. The idea is that $\mathbf{c} - \mathbf{x}$ will have relatively small entries (depending on how close $\mathbf{c}$ is to the lattice), and therefore $(\mathbf{c} - \mathbf{x}, 1)$ will be a short vector in L'. However, this is merely an efficient heuristic technique, and does not provide a reduction from CVP to SVP.

1.4.4 Theoretical results on short vectors

As mentioned in the SVP description, every lattice has a shortest nonzero vector $\mathbf{u} \in L$. Minkowski defined the length of one of the shortest nonzero vectors as the *first minimum* of the lattice L, denoted by $\lambda_1(L)$. It was also mentioned that this vector is not necessarily unique; if $\mathbf{u}$ is a solution, then so is $-\mathbf{u}$. This means that a second-to-shortest nonzero vector is not necessarily a useful concept. Therefore, Minkowski included linear independence in his definition of the successive minima of a lattice:

Definition 1.30. *Let $L \subset \mathbb{R}^n$ be a d-rank lattice. Define for $1 \leq i \leq d$ the i^{th} minimum as*

$$\lambda_i(L) = \min_{\substack{\mathbf{u}_1, \ldots, \mathbf{u}_i \in L \\ \text{independent}}} \max_{1 \leq j \leq i} \|\mathbf{u}_j\|.$$

Thus, the first minimum $\lambda_1(L)$ is, as before, the length of one of the shortest nonzero vectors from the lattice. The second minimum is obtained by taking two independent lattice vectors $\mathbf{u}_1, \mathbf{u}_2$ such that the longest of the two is as short as possible, its length being the second minimum. In general, the j'th minimum is obtained by taking j independent lattice vectors $\mathbf{u}_1, \ldots, \mathbf{u}_j$ such that the longest of these j vectors is as short as possible. It follows that these minima are nondecreasing, i.e., $\lambda_1(L) \leq \ldots \leq \lambda_d(L)$.

Now, by definition there will always be linearly independent vectors that reach the minima simultaneously. But these vectors do not always form a lattice basis when the dimension is 4 or greater. On top of that, when the dimension is 5 or greater, a basis reaching the minima simultaneously need not exist.

Let $L \subset \mathbb{Z}^n$ be a full-rank lattice. Recall that, by Lemma 1.12(ii), for $r > 0$

$$\lim_{r \to \infty} \frac{r^n v_n}{|\overline{\mathcal{B}(\mathbf{0}, r)} \cap L|} = \text{vol}(L).$$

Rewriting this limit shows that, heuristically:

$$\frac{r^n v_n}{\text{vol}(L)} \approx |\overline{\mathcal{B}(\mathbf{0}, r)} \cap L|. \tag{1.6}$$

Now, it is interesting to consider for which r the left-hand side of (1.6) is one, i.e., when there is expected to be only one lattice vector in the n-dimensional ball of radius r. Using Stirling's formula to approximate v_n gives the solution

$$r \approx \sqrt{\frac{n}{2\pi \mathrm{e}}}(\mathrm{vol}(L))^{1/n}.$$

This gives rise to the definition of the *expected shortest length* of a nonzero vector in a lattice L:

$$\sigma(L) = \sqrt{\frac{n}{2\pi \mathrm{e}}}(\mathrm{vol}(L))^{1/n}. \tag{1.7}$$

Note that the previous argument only holds heuristically. It is based on the Gaussian Heuristic, which says that for any set $C \subset \mathbb{R}^n$, the expected number of lattice points in $L \cap C$ is $\mathrm{vol}(C)/\mathrm{vol}(L)$. Furthermore, the word *expected* is not meant in the probabilistic sense, but heuristically.

It is also interesting to consider how long these shortest vectors of a lattice can be. Is it possible to somehow bound the length of a shortest vector? Let $L \subset \mathbb{R}^n$ be a d-rank lattice. The length of the shortest vector $\lambda_1(L)$ and the volume of the lattice $\mathrm{vol}(L)$ are both *homogeneous* functions of L. Indeed, if L is replaced by tL, the length of the shortest vector becomes $\lambda_1(tL) = |t|\lambda_1(L)$, whereas for the volume $\mathrm{vol}(tL) = |t|^d \mathrm{vol}(L)$. Therefore, the quantity $\lambda_1(L)$ has the same degree as $\mathrm{vol}(L)^{1/d}$. Hermite [35] was the first to prove the fact that $\lambda_1(L)/\mathrm{vol}(L)^{1/d}$ could be upper bounded, resulting in the following definition:

Definition 1.31. *Hermite's constant γ_d is the supremum of $\lambda_1(L)^2/\mathrm{vol}(L)^{2/d}$ over all d-rank lattices L.*

The powers of two used in this definition stem from the fact that Hermite looked at quadratic forms rather than lattices. Using Hermite's constant, it is possible to upper bound the length of the shortest vector in any d-rank lattice L:

$$\lambda_1(L) \leq \sqrt{\gamma_d}\,\mathrm{vol}(L)^{1/d}. \tag{1.8}$$

It is known that for all d, there exists a lattice reaching equality in (1.8) for Hermite's constant γ_d. These lattices are called *critical*. However, it turns out that finding exact values for γ_d is a very difficult problem. The only known values for γ_d are for $1 \leq d \leq 8$ and $d = 24$. Additionally, all critical lattices are known for each of these ranks, up to scaling and isometry.

It is hard to give the explicit upper bound on the shortest length of a vector as in (1.8) when γ_d is only known for nine different values of d. Fortunately, it is possible to give rather good asymptotical bounds on γ_d. Hermite himself gave an upper bound on γ_d, as seen in the following theorem:

Theorem 1.32 (Hermite's inequality)**.** *Hermite's constant for lattices of rank d satisfies the upper bound*

$$\gamma_d \leq \gamma_2^{d-1}, \tag{1.9}$$

for all integers $d \geq 2$, and where $\gamma_2 = \sqrt{4/3}$.

Proof. The proof goes by induction. For the base case, consider $d = 2$, where (1.9) is trivially satisfied. To see that $\gamma_2 \leq \sqrt{4/3}$, consider a two-dimensional lattice L spanned by $\mathbf{b}_1$ and $\mathbf{b}_2$. Assume without loss of generality that $\|\mathbf{b}_1\| = \lambda_1(L)$ and $\langle \mathbf{b}_1, \mathbf{b}_2 \rangle \leq \|\mathbf{b}_1\|^2/2$. If the second condition is not satisfied, replace $\mathbf{b}_2$ by the vector obtained when lifting $\pi_2(\mathbf{b}_2)$ back to L, as in Equation (1.3). Now consider the GSO-vectors $\mathbf{b}_1^* = \mathbf{b}_1$ and $\mathbf{b}_2^* = \mathbf{b}_2 - \frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle}{\|\mathbf{b}_1\|^2}\mathbf{b}_1$. The following inequalities hold:

$$\begin{aligned}\|\mathbf{b}_1\|^2 \leq \|\mathbf{b}_2\|^2 &= \|\mathbf{b}_2^* + \frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle}{\|\mathbf{b}_1\|^2}\mathbf{b}_1\|^2 \leq \|\mathbf{b}_2^*\|^2 + \left|\frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle}{\|\mathbf{b}_1\|^2}\right|^2 \|\mathbf{b}_1\|^2 \\ &\leq \|\mathbf{b}_2^*\|^2 + \|\mathbf{b}_1\|^2/4.\end{aligned}$$

By Equation (1.4), the volume of the lattice L is given by $\text{vol}(L) = \|\mathbf{b}_1\|\|\mathbf{b}_2^*\|$. Thus, rewriting this expression gives

$$\lambda_1(L) = \|\mathbf{b}_1\| \leq \sqrt{4/3}\|\mathbf{b}_2^*\| = \sqrt{4/3}\,\text{vol}(L)/\|\mathbf{b}_1\|.$$

It now follows that

$$\lambda_1(L)^2/\,\text{vol}(L)^{2/2} \leq \sqrt{4/3},$$

for every lattice L of rank 2. Thus, $\gamma_2 \leq \sqrt{4/3}$ by definition of Hermite's constant.

To see that $\gamma_2 \geq \sqrt{4/3}$, consider the hexagonal lattice spanned by $\mathbf{b}_1 = (1, 0)$ and $\mathbf{b}_2 = (\frac{1}{2}, \frac{1}{2}\sqrt{3})$. Then $\text{vol}(L) = \frac{1}{2}\sqrt{3}$ and $\lambda_1(L) = \lambda_2(L) = 1$. Thus, $\lambda_1(L)^2/\,\text{vol}(L) = 1/(1/2\sqrt{3}) = \sqrt{4/3}$, as desired.

The proof to complete the induction is similar to the proof of $\gamma_2 \leq \sqrt{4/3}$. Assume the induction hypothesis (1.9) is true for rank $d-1$. Consider a d-rank lattice $L \subset \mathbb{R}^n$ and one of its shortest nonzero vectors $\mathbf{b}_1$. Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ be any basis for L containing $\mathbf{b}_1$. Now consider the projected lattice $L' = \pi_2(L)$ and its shortest vector $\mathbf{b}_2'$. Note that the Gram-Schmidt Orthogonalization of the basis $\{\pi_2(\mathbf{b}_2), \ldots, \pi_2(\mathbf{b}_d)\}$ of L' satisfies that $(\pi_2(\mathbf{b}_i))^* = \mathbf{b}_i^*$ for all $2 \leq i \leq d$, because the first step of the GSO in L is to project these vectors on the orthogonal complement of the span of $\mathbf{b}_1$ anyway. By Equation (1.4) and the fact that $\mathbf{b}_1 = \mathbf{b}_1^*$, the volume of L' satisfies

$$\text{vol}(L') = \prod_{i=2}^{d} \|\mathbf{b}_i^*\| = \text{vol}(L)/\|\mathbf{b}_1\|.$$

As L' is a $(d-1)$-rank lattice, the definition of Hermite's constant gives that

$$\|\mathbf{b}_2'\| = \lambda_1(L') \leq \sqrt{\gamma_{(d-1)}}\,\text{vol}(L')^{1/(d-1)} \leq (\gamma_2)^{(d-2)/2}\,\text{vol}(L')^{1/(d-1)},$$

where the last inequality follows from the induction hypothesis. Recall from Equation (1.3) that $\mathbf{b}_2'$ can be lifted back to L by adding the appropriate scalar multiple of $\mathbf{b}_1$ resulting in a nonzero vector $\mathbf{b}_2 \in L$ such that $\|\mathbf{b}_2\|^2 \leq \|\mathbf{b}_2'\|^2 + \|\mathbf{b}_1\|^2/4$. Now $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$, because $\mathbf{b}_1$ is a shortest vector in L by assumption. Since $\text{vol}(L') = \text{vol}(L)/\|\mathbf{b}_1\|$, it follows that

$$\begin{aligned}\|\mathbf{b}_1\| \leq \sqrt{4/3}\|\mathbf{b}_2'\| &\leq \gamma_2^{d/2}\,\text{vol}(L')^{1/(d-1)} \\ &= \gamma_2^{d/2}(\text{vol}(L)/\|\mathbf{b}_1\|)^{1/(d-1)}.\end{aligned}$$

This implies

$$\lambda_1(L) = \|\mathbf{b}_1\| \leq \gamma_2^{(d-1)/2}\,\text{vol}(L)^{1/d}.$$

From here it is easily seen that

$$\lambda_1(L)^2/\,\text{vol}(L)^{2/d} \leq \gamma_2^{d-1},$$

for any d-rank lattice L, and specifically for a critical one, where $\lambda_1(L)^2/\,\text{vol}(L)^{2/d} = \gamma_d$. This proves the desired result. □

This bound is exponential in the dimension d. Since then, both linear upper and lower bounds have been found for γ_d. The first linear upper bound was shown by Minkowski, as a consequence of his convex body theorem.

Theorem 1.33 (Minkowski's Convex Body Theorem)**.** *Let $L \subset \mathbb{R}^n$ be an n-rank lattice. Let $C \subset \mathbb{R}^n$ be a measurable convex set, that is symmetric with respect to $\mathbf{0}$ and of measure strictly greater than $2^n\,\text{vol}(L)$. Then $C \cap L$ contains at least one nonzero vector.*

The proof of this theorem makes use of a generalization of the pigeonhole principle. As a corollary of this theorem, an upper bound on the length of the shortest vector is obtained:

Corollary 1.34. *Let $L \subset \mathbb{R}^n$ be a d-dimensional lattice and let v_d be defined as in* (1.1)*. Then L contains a nonzero vector* $\mathbf{x}$ *such that*

$$\lambda_1(L) \leq \|\mathbf{x}\| \leq 2\left(\frac{\mathrm{vol}(L)}{v_d}\right)^{1/d}.$$

Rewriting this expression gives that

$$\frac{\lambda_1(L)^2}{\mathrm{vol}(L)^{2/d}} \leq \frac{4}{(v_d)^{2/d}}.$$

Since this holds for all d-rank lattices L, it specifically holds for a critical lattice of rank d, and thus

$$\gamma_d \leq \frac{4}{(v_d)^{2/d}},$$

for $d \geq 1$. Now, using known properties of v_d, the following linear bound can be derived:

$$\gamma_d \leq 1 + \frac{d}{4}, \tag{1.10}$$

for all $d \geq 1$.

These bounds can be used to give an upper bound for the length λ_1 of the shortest vector in a lattice, as in (1.8). The next question is whether λ_2 can be bounded as well. Unfortunately, this is not the case, as can be seen by the following example:

Example 1.35. *Consider the lattice L spanned by the columns of the following matrix:*

$$\begin{pmatrix} \varepsilon & 0 \\ 0 & \frac{1}{\varepsilon} \end{pmatrix}.$$

For $\varepsilon \leq 1$, the length of the shortest vector is $\lambda_1(L) = \varepsilon$, whereas the length of the second shortest vector (linearly independent from the first) is $\lambda_2(L) = 1/\varepsilon$, which tends to infinity for $\varepsilon \to 0$. Thus, $\lambda_2(L)$ can be arbitrarily large compared the volume of L, which is 1 in this case. Additionally, this example shows that $\lambda_1(L)$ can be arbitrarily small compared to the upper bound of Hermite's constant.

Although it is not possible to bound any other successive minima using the volume of L, Minkowski showed it is possible to bound the geometric mean of the successive minima:

Theorem 1.36 (Minkowski's Second Theorem)**.** *Let $L \subset \mathbb{R}^n$ be a d-dimensional lattice. Then*

$$\left(\prod_{i=1}^{r} \lambda_i(L)\right)^{1/r} \leq \sqrt{\gamma_d}\,\mathrm{vol}(L)^{1/d},$$

for every $1 \leq r \leq d$.

1.5 Lattice basis reduction

Minkowski's results on short vectors in a lattice of the previous section do not have constructive proofs. They do not provide a procedure to retrieve such short vectors. There are several algorithms that solve the exact version of SVP, retrieving an actual shortest vector. However, these algorithms perform some variation of a complete enumeration of short vectors. As such, they do not run in polynomial time for varying lattice rank.

As opposed to Minkowski's result, the proof of Hermite's inequality does give some constructive pointers to find short vectors, but its approximation factor is exponential in the rank. In 1982, Lenstra, Lenstra and Lovász published the Lenstra-Lenstra-Lovász algorithm (LLL), which is a basis reduction algorithm for lattices, based on Hermite's inequality. As a byproduct of reducing the lattice basis, the LLL algorithm finds a short vector that approximates the shortest nonzero vector with an exponential approximation factor, such as in Hermite's inequality. The LLL algorithm was a huge breakthrough in lattice basis reduction, because it runs in polynomial time as a function of the rank of the lattice. This section provides an introduction to lattice reduction, and specifically to the LLL algorithm.

Lattices are generally represented by a basis. In $\mathbb{R}^n$, there are orthonormal bases, i.e., bases $(\mathbf{e}_1, \ldots, \mathbf{e}_n)$ where all vectors have norm $\|\mathbf{e}_i\| = 1$ and they are pairwise orthogonal, i.e., $\langle \mathbf{e}_i, \mathbf{e}_j \rangle = 0$ if $i \neq j$. It is interesting to consider orthogonal bases for lattices as well. Lattice basis reduction algorithms aim to give a basis that is reduced in the sense that its basis vectors are close to orthogonal. Since it is not immediately clear how to define what "close to orthogonal" means, bases that consist of short vectors will be considered instead. The reason for the use of short vectors will become apparent in this section. Algorithms for SVP and CVP often use basis reduction algorithms as a preparation step, because reduced bases are easier to work with.

1.5.1 Orthogonality

An attractive quality of vector space bases is orthogonality. In lattices, it is not always possible to orthogonalize the basis. Recall that for any basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ the volume of the lattice always satisfies $\mathrm{vol}(L) \leq \prod_{i=1}^{d} \|\mathbf{b}_i\|$, with equality if and only if all the basis vectors are orthogonal to each other. This gives rise to the following definition:

Definition 1.37. *The orthogonality defect of the basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ of a lattice L is denoted by $\prod_{i=1}^{d} \|\mathbf{b}_i\| / \mathrm{vol}(L) \geq 1$. Equality holds if and only if the basis is orthogonal.*

It is easy to see that decreasing the length of the vectors in a basis also decreases the orthogonality defect of the lattice.

1.5.2 Reduction notions

There is no single definition of what a reduced basis is. Several reduction notions will be defined here.

Size-reduction

The first notion is intuitively very easy to achieve. However, it is rather weak, in the sense that by itself, it does not offer very good bounds on the orthogonality defect or length of the basis vectors.

Definition 1.38. *A basis $\{\mathbf{b}_1, \ldots \mathbf{b}_d\}$ of a lattice $L \subset \mathbb{R}^n$ is called size-reduced if the GSO-coefficients μ_{ij} satisfy*

$$|\mu_{ij}| \leq \frac{1}{2},$$

for all $1 \leq j < i \leq d$.

An equivalent geometric interpretation of this definition is that $\mathbf{b}_i$ is already as close to the orthogonal complement of $\mathbf{b}_j^*$ as possible, for all $j < i$. As a result, when projecting $\mathbf{b}_i$ on the orthogonal complement of $\mathbf{b}_j^*$, less than half the vector $\mathbf{b}_j^*$ needs to be subtracted or added. This basically means that this basis is the closest basis to this particular GSO, while still spanning the same lattice. Furthermore, since $\mathbf{b}_i = \mathbf{b}_i^* + \sum_{j<i} \mu_{ij} \mathbf{b}_j^*$, the square of the norm of $\mathbf{b}_i$ will satisfy

$$\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_i\|^2 \leq \|\mathbf{b}_i^*\|^2 + \left(\frac{1}{2}\right)^2 \sum_{j=1}^{i-1} \|\mathbf{b}_j^*\|^2,$$

by the triangle inequality. An algorithm to size-reduce bases will be given later. Many other reduction notions include size-reduction as a requirement in their definition.

Lagrange-reduction

A special case of the basis reduction problem is the case where $d = 2$. In the late 18th century, an algorithm (which has been attributed to both Lagrange [52] and Gauss [22]) was invented to reduce the basis for lattices of rank $d = 2$ in terms of quadratic forms. This algorithm returns a basis that satisfies the following reduction notion:

Definition 1.39. *Let $L \subset \mathbb{R}^n$ be a 2-dimensional lattice. A basis $\{\mathbf{b}_1, \mathbf{b}_2\}$ of L is said to be Lagrange-reduced if $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$ and $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \|\mathbf{b}_1\|^2/2$.*

It is easy to check whether a given basis satisfies this notion. A geometric interpretation of this definition is that in the plane spanned by $\mathbf{b}_1$ and $\mathbf{b}_2$, the vector $\mathbf{b}_1$ is within the circle centered at the origin with radius $\|\mathbf{b}_2\|$, and the angle between the two vectors is in the interval $(\pi/3, 2\pi/3)$ modulo π. Note that the condition on the inner product is equivalent to size-reduction.

It turns out that this reduction notion also minimizes the length of the vectors.

Proposition 1.40. *Let $L \subset \mathbb{R}^n$ be a 2-dimensional lattice. The basis $\{\mathbf{b}_1, \mathbf{b}_2\}$ of L is Lagrange-reduced if and only if $\|\mathbf{b}_1\| = \lambda_1(L)$ and $\|\mathbf{b}_2\| = \lambda_2(L)$.*

By definition, there always exist vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ that simultaneously reach the minima $\lambda_1(L)$ and $\lambda_2(L)$. Furthermore, because the space is 2-dimensional, these vectors also form a basis of the lattice. Assume they do not, then there must be lattice vector $\mathbf{x} = x_1\mathbf{b}_1 + x_2\mathbf{b}_2$ where x_1 and x_2 are not integral. But then the vector $(x_1 - \lfloor x_1 \rceil)\mathbf{b}_1 + (x_2 - \lfloor x_2 \rceil)\mathbf{b}_2$ is shorter than $\mathbf{b}_2$ by the fact that $\mathbf{b}_1$ and $\mathbf{b}_2$ are linearly independent and the triangle inequality. Thus, there always exist bases that simultaneously reach the minima, and by Proposition 1.40, there always exist Lagrange-reduced bases.

The algorithm that achieves this reduction notion is a generalization of Euclid's algorithm for computing the greatest common divisor to a 2-dimensional vector space. It will be explained in the next section about reduction algorithms.

HKZ-reduction

The next reduction notion is very strong, but rather hard to achieve:

Definition 1.41 (HKZ-reduction)**.** *A basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ of a lattice $L \subset \mathbb{R}^n$ is called Hermite-Korkine-Zolotarev-reduced (HKZ-reduced) if it is size-reduced and*

$$\|\mathbf{b}_i^*\| = \lambda_1(\pi_i(L)), \tag{1.11}$$

for all $1 \leq i \leq d$, where $\mathbf{b}_i^*$ is the GSO-projection of $\mathbf{b}_i$.

This is a natural definition, since $\mathbf{b}_i^* \in \pi_i(L)$ is a nonzero vector in the projected lattice by design of the GSO procedure. Furthermore, $\mathbf{b}_d^*$ is the shortest vector of $\pi_d(L)$, because all vectors in $\pi_d(L)$ are multiples of $\mathbf{b}_d$ in the original lattice: the contributions of the other $\mathbf{b}_i$'s have been projected onto $\mathbf{0}$.

HKZ-reduced bases have the following interesting properties:

Theorem 1.42. *Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ be an HKZ-reduced basis of a lattice $L \subset \mathbb{R}^n$. Then*

$$\frac{4}{i+3} \leq \left(\frac{\|\mathbf{b}_i\|}{\lambda_i(L)}\right)^2 \leq \frac{i+3}{4},$$

for all $1 \leq i \leq d$.

For a proof of this theorem, see the work by Mahler [63] for a proof of the upper bound and the work by Lagarias, Lenstra and Schnorr [50] for a proof of the lower bound. The proofs combine several observations on the lengths of projected vectors, as well as some properties of size-reduced bases.

Another nice property of HKZ-reduced bases is that for any HKZ-reduced basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$, the projected partial basis $\{\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_j)\}$ is HKZ-reduced for all $1 \leq i \leq j \leq d$. This allows for properties of lower-dimensional HKZ-reduced bases to be transferred to higher dimensions. However, these properties come at a cost. The requirement (1.11) demands that each GSO projection must be a shortest vector in some projected lattice. This suggests that every vector $\mathbf{b}_i$ requires some instance of SVP to be solved in some projected lattice. Getting $\mathbf{b}_1$ to satisfy the requirement is already an SVP-instance in the original lattice. Therefore, computing a basis that satisfies this reduction notion is at least as hard as solving the shortest vector problem.

Hermite's reduction notions

Hermite devised two algorithms to reduce a basis, with a different reduction notion for both. Both algorithms appeared in letters to Jacobi [35]. The first algorithm he created obtains a basis that satisfies the following two properties:

- The basis is size-reduced.
- The GSO vectors satisfy $\|\mathbf{b}_i^*\| \leq (4/3)^{(d-i)/4} \operatorname{vol}(\pi_i(L))^{1/(d-i+1)}$ for every $1 \leq i \leq d$, which is equivalent to Hermite's inequality (1.9) in the projected lattice $\pi_i(L)$.

This is not a very strong reduction notion. The following example shows that the orthogonality defect of a basis satisfying this reduction notion is unbounded when the rank of the lattice is $d \geq 3$.

Example 1.43. *Let $0 < \varepsilon < 1$ and let L be the lattice spanned by the basis*

$$B = \begin{pmatrix} 1 & 1/2 & 1/2 \\ 0 & \varepsilon & \varepsilon/2 \\ 0 & 0 & 1/\varepsilon \end{pmatrix}.$$

To see that this basis satisfies the reduction notion, first compute the GSO:

$$\begin{aligned}
\mathbf{b}_1^* &= \mathbf{b}_1 = (1,0,0), \\
\mathbf{b}_2^* &= \mathbf{b}_2 - \frac{\langle \mathbf{b}_1^*, \mathbf{b}_2 \rangle}{\|\mathbf{b}_1^*\|^2} \mathbf{b}_1^* = (1/2, \varepsilon, 0) - \frac{1}{2}(1,0,0) = (0, \varepsilon, 0), \\
\mathbf{b}_3^* &= \mathbf{b}_3 - \frac{\langle \mathbf{b}_1^*, \mathbf{b}_3 \rangle}{\|\mathbf{b}_1^*\|^2} \mathbf{b}_1^* - \frac{\langle \mathbf{b}_2^*, \mathbf{b}_3 \rangle}{\|\mathbf{b}_2^*\|^2} \mathbf{b}_2^* = (1/2, \varepsilon/2, 1/\varepsilon) - \frac{1}{2}(1,0,0) - \frac{\varepsilon^2}{2\varepsilon^2}(0,\varepsilon,0) = (0,0,1/\varepsilon).
\end{aligned}$$

It follows that the basis is size-reduced, because $|\mu_{ij}| = 1/2$ for $1 \leq j < i \leq 3$. Now, by (1.4), the volumes of the projected lattices can be computed by multiplying the norms of the GSO-vectors:

$$\begin{aligned}
\operatorname{vol}(L) &= \|\mathbf{b}_1^*\| \cdot \|\mathbf{b}_2^*\| \cdot \|\mathbf{b}_3^*\| = 1 \cdot \varepsilon \cdot 1/\varepsilon = 1, \\
\operatorname{vol}(\pi_2(L)) &= \|\mathbf{b}_2^*\| \cdot \|\mathbf{b}_3^*\| = \varepsilon \cdot 1/\varepsilon = 1, \\
\operatorname{vol}(\pi_3(L)) &= \|\mathbf{b}_3^*\| = 1/\varepsilon.
\end{aligned}$$

Note that the volume of L does not depend on ε. The second requirement of the reduction notion can now be verified:

$$\begin{aligned}
\|\mathbf{b}_1^*\| &= 1 \leq (4/3)^{2/4} \cdot 1^{1/3} = \sqrt{4/3}, \\
\|\mathbf{b}_2^*\| &= \varepsilon \leq (4/3)^{1/4} \cdot 1^{1/2} = (4/3)^{1/4}, \\
\|\mathbf{b}_3^*\| &= 1/\varepsilon \leq (4/3)^0 \cdot (1/\varepsilon)^1 = 1/\varepsilon.
\end{aligned}$$

This holds for all $0 < \varepsilon < 1$, and if ε tends to 0, the second basis vector will lie close to parallel to the first basis vector, and the orthogonality defect becomes

$$\|\mathbf{b}_1\| \cdot \|\mathbf{b}_2\| \cdot \|\mathbf{b}_3\| / \operatorname{vol}(L) = 1 \cdot \sqrt{1/4 + \varepsilon^2} \cdot \sqrt{1/4 + \varepsilon^2/4 + 1/\varepsilon^2}/1 \to \infty,$$

as $\varepsilon \to 0$.

Hermite noticed that his algorithm did not correspond with the aforementioned 2-dimensional algorithm that satisfies Lagrange's reduction notion (Definition 1.39). He created a second algorithm, which obtains a basis satisfying the following properties:

- The basis is size-reduced.
- The GSO-vectors satisfy that $\pi_i(\mathbf{b}_i)$ is the smallest vector in the projected basis of $\pi_i(L)$, i.e.,

$$\|\pi_i(\mathbf{b}_i)\| \leq \|\pi_i(\mathbf{b}_j)\|, \tag{1.12}$$

  for all $1 \leq i < j \leq d$.

The requirement (1.12) is much weaker than the requirement (1.11) from Definition 1.41. In (1.11), the projection of the i'th basis vector $\pi_i(\mathbf{b}_i)$ must be at least as short as all vectors in the projected lattice $\pi_i(L)$, whereas in (1.12), $\pi_i(\mathbf{b}_i)$ only needs to be at least as short as the projection of the other basis vectors $\pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_d)$. This reduction notion guarantees that the GSO-vectors satisfy

$$\begin{aligned}
\|\mathbf{b}_i^*\|^2 &= \|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2 \\
&= \|\pi_{i+1}(\mathbf{b}_{i+1}) + \mu_{i+1,i}\mathbf{b}_i^*\|^2 \\
&= \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2\|\mathbf{b}_i^*\|^2 \\
&\leq \|\mathbf{b}_{i+1}^*\|^2 + \frac{1}{4}\|\mathbf{b}_i^*\|^2, \\
\|\mathbf{b}_i^*\|^2 &\leq \frac{4}{3}\|\mathbf{b}_{i+1}^*\|^2.
\end{aligned} \tag{1.13}$$

From (1.13), it follows by induction that

$$\|\mathbf{b}_i^*\|^2 \leq \left(\frac{4}{3}\right)^{j-i} \|\mathbf{b}_j^*\|^2, \tag{1.14}$$

for all $1 \leq i \leq j \leq d$. Equation (1.14) leads to the following bound on the norms of the vectors of the reduced basis:

$$\begin{aligned}
\|\mathbf{b}_i\|^2 &= \|\mathbf{b}_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{ij}^2 \|\mathbf{b}_j^*\|^2 \\
&\leq \|\mathbf{b}_i^*\|^2 + \sum_{j=1}^{i-1} \frac{1}{4}\left(\frac{4}{3}\right)^{i-j} \|\mathbf{b}_i^*\|^2 \\
&= \left(1 + \frac{1}{4}\frac{(4/3)^i - 4/3}{4/3 - 1}\right)\|\mathbf{b}_i^*\|^2 = \left(\frac{4}{3}\right)^{i-1} \|\mathbf{b}_i^*\|^2.
\end{aligned} \tag{1.15}$$

Using the inequality in (1.15), the orthogonality defect of a basis of a lattice L that satisfies this reduction notion can be bounded:

$$\begin{aligned}
\prod_{i=1}^{d} \|\mathbf{b}_i\| / \operatorname{vol}(L) &\leq \prod_{i=1}^{d} \left(\frac{4}{3}\right)^{(i-1)/2} \|\mathbf{b}_i^*\| / \operatorname{vol}(L) \\
&= \left(\frac{4}{3}\right)^{\sum_{i=1}^{d}(i-1)/2} = \left(\frac{4}{3}\right)^{d(d-1)/4}.
\end{aligned}$$

Although this is a suitable reduction notion that gives a bound on the orthogonality defect, Hermite's algorithms are not known to run in polynomial time as a function of the lattice rank. Hendrik Lenstra [56] proposed a reduction algorithm that relaxed the second requirement of Hermite's reduction notion:

$$c\|\pi_i(\mathbf{b}_i)\| \leq \|\pi_i(\mathbf{b}_j)\|,$$

where $c \in (1/4, 1)$ is a constant. Hendrik Lenstra proved in [56] that his algorithm runs in polynomial time for any fixed dimension. In a letter to Hendrik[1], Lovász proposed a modification which guarantees a polynomial running time for any dimension. Recall that the reasoning in Equations (1.13), (1.14) and (1.15) used only the inequalities $\|\mathbf{b}_i^*\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2$ and $|\mu_{ij}| \leq 1/2$. Lovász' proposal was to replace the inequalities $c\|\pi_i(\mathbf{b}_i)\| \leq \|\pi_i(\mathbf{b}_j)\|$ by the single inequality $c\|\pi_i(\mathbf{b}_i)\| \leq \|\pi_i(\mathbf{b}_{i+1})\|$, for $1 \leq i \leq d$. This leads to the following reduction notion.

Definition 1.44. *Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ be a basis of a lattice $L \subset \mathbb{R}^n$. It is said to be LLL-reduced with factor $\delta \in (\frac{1}{4}, 1)$ if it is size-reduced and*

$$\delta\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^* + \mu_{i+1,i}\mathbf{b}_i^*\|^2, \tag{1.16}$$

for all $1 < i \leq d$.

The inequality (1.16) is called Lovász' condition, and is equivalent to $\delta\|\pi_i(\mathbf{b}_i)\| \leq \|\pi_i(\mathbf{b}_{i+1})\|$. Its purpose stems from the fact that Gram-Schmidt Orthogonalization depends on the order of the vectors in the basis. If the vectors $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ would be swapped, then only $\mathbf{b}_i^*$ and $\mathbf{b}_{i+1}^*$ would change, and the new $\mathbf{b}_i^{*'} = \mathbf{b}_{i+1}^* + \mu_{i+1,i}\mathbf{b}_i^*$. Thus, the left-hand side of (1.16) is δ times the norm of the 'old' $\mathbf{b}_i^*$ (before swapping), whereas the right-hand side is the norm of the 'new' $\mathbf{b}_i^{*'}$ (after swapping). This means that if Lovász' condition does not hold, swapping $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ would significantly reduce the norm of $\mathbf{b}_i^*$. On the other hand, if the condition is satisfied, swapping will not improve the norm of $\mathbf{b}_i^*$ much. The most natural value for δ is 1, since that will lead to the best reduction. Unfortunately, it is unknown if a basis satisfying that reduction notion can be computed in polynomial time.

Another equivalent formulation of (1.16) is

$$(\delta - \mu_{i+1,i}^2)\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2,$$

which implies that $\mathbf{b}_{i+1}^*$ is not much shorter than $\mathbf{b}_i^*$ for each i. As size-reduction implies that $\mu_{i+1,i}^2 < 1/4$, this formulation yields the inequality

$$\begin{aligned}(\delta - \frac{1}{4})\|\mathbf{b}_i^*\|^2 \leq (\delta - \mu_{i+1,i}^2)\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2,\\ \|\mathbf{b}_i^*\|^2 \leq \left(\delta - \frac{1}{4}\right)^{-1}\|\mathbf{b}_{i+1}\|^2.\end{aligned} \tag{1.17}$$

This leads to the following theorem:

Theorem 1.45. *Let $\frac{1}{4} < \delta \leq 1$, and take $\alpha = 1/(\delta - \frac{1}{4})$. If the basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ of a lattice $L \subset \mathbb{R}^n$ is LLL-reduced with factor δ, then*

(i) $\|\mathbf{b}_1\| \leq \alpha^{(d-1)/4}(\mathrm{vol}(L))^{1/d}$.

(ii) $\|\mathbf{b}_i\| \leq \alpha^{(d-1)/2}\lambda_i(L)$, $1 \leq i \leq d$.

(iii) $\prod_{i=1}^d \|\mathbf{b}_i\| \leq \alpha^{d(d-1)/4}\,\mathrm{vol}(L)$.

This result shows the connection between an LLL-reduced basis and Hermite's inequality (1.9). The closer δ is to 1, the better Hermite's inequality is approximated, because the constant $4/3$ becomes $4/3+\varepsilon$ for $\delta < 1$. The proof of Theorem 1.45 follows the same reasoning as the derivation of Equations (1.13), (1.14) and (1.15), where $\alpha = 4/3$ was used, which is the value that arises when $\delta = 1$. The same results hold for $\delta \in (1/4, 1)$, by replacing (1.13) by (1.17).

[1]For a detailed description of the history of the LLL algorithm, see [101].

1.5.3 Lattice reduction algorithms

As seen in the section on lattice reduction, there is not one single notion of a reduced lattice basis. Therefore, there are also several different algorithms, each trying to reach a different reduction notion. This section will focus on the LLL-algorithm and its reduction notions. First, an algorithm will be given to size-reduce a basis, based on the Gram-Schmidt process. Then Lagrange's reduction algorithm will be described, which is the 2-dimensional vector space equivalent of Euclid's algorithm for the greatest common divisor. Finally, the LLL-algorithm will be given, which can be seen as an improved version of Hermite's algorithms.

Size-reduction

The Gram-Schmidt Orthogonalization process is already quite close to a basis reduction algorithm. The only problem is that the resulting vectors are not necessarily in the lattice. By slightly tweaking the process, it is possible to create an algorithm that achieves the size-reduction notion. It takes a basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ as input and the output will be a size-reduced basis.

Algorithm 1 Size-reduction algorithm

```
Compute the GSO coefficients μ_ij of {b_1, ..., b_d}.
for i = 2 to d do
  for j = i − 1 down to 1 do
    b_i ← b_i − ⌈μ_ij⌋ b_j
    for k = 1 to j do
      μ_ik ← μ_ik − ⌈μ_ij⌋ μ_jk
    end for
  end for
end for
```

Algorithm 1 is equivalent to an algorithm that performs the following steps:

1. Compute the GSO coefficients μ_{ij} of $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$.
2. Project all basis vectors $\mathbf{b}_i$ on their GSO vectors $\mathbf{b}_i^*$, iterating from $i = 1$ to d.
3. Lift all GSO vectors back to lattice vectors using the lifting procedure in Equation (1.3), iterating from $i = d$ to 1.

However, for efficiency, algorithm 1 only iterates over all vectors $\mathbf{b}_i$ once from $i = 1$ to d, and lifts them back immediately after projecting. This means that when the algorithm is done with the vector $\mathbf{b}_i$, it will generally not be equal to the GSO vector $\mathbf{b}_i^*$. However, the $\mathbf{b}_i$ will still be used by the algorithm to reduce $\mathbf{b}_j$'s for $j > i$ in later iterations. Thus, the GSO coefficients μ_{ik} need to be adapted after $\mathbf{b}_i$ is reduced, to correct for the difference between $\mathbf{b}_i$ and $\mathbf{b}_i^*$. Note that the algorithm changes the basis vectors, but not their GSO-projections.

Example 1.46. *This example shows Algorithm 1 in action. It is applied to the lattice basis spanned by the columns of the matrix*

$$\begin{pmatrix} 3 & 2 & 1 \\ 0 & 3 & 2 \\ 0 & 0 & 3 \end{pmatrix}.$$

First, the GSO-coefficients are computed. They are shown in the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 \\ 2/3 & 1 & 0 \\ 1/3 & 2/3 & 1 \end{pmatrix}.$$

The size-reduction begins, for $i = 2$ and $j = 1$, $\mathbf{b}_2 = (2, 3, 0)$ is reduced with respect to $\mathbf{b}_1$:

$$\mathbf{b}_2 \leftarrow \mathbf{b}_2 - \lceil \mu_{2,1} \rfloor \mathbf{b}_1 = (2, 3, 0) - 1 \cdot (3, 0, 0) = (-1, 3, 0).$$

Thus, $\mathbf{b}_2$ becomes $(-1, 3, 0)$. Next, the matrix of GSO-coefficients is modified, which results in

$$\begin{pmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ 1/3 & 2/3 & 1 \end{pmatrix}.$$

The value $\mu_{2,1}$ is now $-1/3$. This signifies that $\mathbf{b}_1$ was subtracted $1/3$ times too many from $\mathbf{b}_2$, with respect to the real GSO-coefficients. Since $\mathbf{b}_2$ is now size-reduced with respect to $\mathbf{b}_1$, the algorithm continues with $\mathbf{b}_3 = (1, 2, 3)$. First, $\mathbf{b}_3$ is size-reduced with respect to $\mathbf{b}_2$ ($i = 3$, $j = 2$):

$$\mathbf{b}_3 \leftarrow \mathbf{b}_3 - \lceil \mu_{3,2} \rfloor \mathbf{b}_2 = (1, 2, 3) - 1 \cdot (-1, 3, 0) = (2, -1, 3).$$

Thus, $\mathbf{b}_3$ becomes $(2, -1, 3)$. The GSO-coefficients are modified again, which results in

$$\begin{pmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ 2/3 & -1/3 & 1 \end{pmatrix}.$$

Note that the value $\mu_{3,1}$ has changed into $2/3$. This happened because $\mathbf{b}_2$ was subtracted once from $\mathbf{b}_3$, but $\mathbf{b}_1$ was subtracted $1/3$ times too many from $\mathbf{b}_2$. As a net result, $\mathbf{b}_1$ was added $1/3$ times to $\mathbf{b}_3$. Additionally, the value $\mu_{3,2}$ is now $-1/3$. As before, this signifies that $\mathbf{b}_2$ was subtracted from $\mathbf{b}_2$ more than necessary with respect to the GSO-coefficients. Finally, $\mathbf{b}_3$ gets size-reduced with respect to $\mathbf{b}_1$.

$$\mathbf{b}_3 \leftarrow \mathbf{b}_3 - \lceil \mu_{3,1} \rfloor \mathbf{b}_1 = (2, -1, 3) - 1 \cdot (3, 0, 0) = (-1, -1, 3).$$

The GSO-coefficients are changed one last time, resulting in

$$\begin{pmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ -1/3 & -1/3 & 1 \end{pmatrix},$$

which shows that the basis is indeed size-reduced. The algorithm now returns the size-reduced basis, resulting in the following basis matrix:

$$\begin{pmatrix} 3 & -1 & -1 \\ 0 & 3 & -1 \\ 0 & 0 & 3 \end{pmatrix}.$$

Lagrange-reduction

Lagrange's algorithm is the vector space equivalent of Euclid's algorithm for computing the greatest common divisor of two natural numbers. The idea is to take the biggest vector and keep removing multiples of the smallest vector until it cannot become smaller, then switch the vectors and repeat the process. Equivalently, this is the repeated application of GSO algorithm that was described previously, while switching the two vectors after every reduction step.

This algorithm returns a basis satisfying Lagrange's reduction notion as in Definition 1.39. As shown before, $\mathbf{u} - \left\lceil \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{v}\|^2} \right\rfloor \mathbf{v}$ is merely the projection of $\mathbf{u}$ onto the orthogonal complement of $\mathbf{v}$, and lifted back to L with the shortest scalar multiple of $\mathbf{v}$ such that it is a lattice point.

Algorithm 2 Lagrange's lattice reduction algorithm

if $\|\mathbf{u}\| < \|\mathbf{v}\|$ **then**
  swap $\mathbf{u}$ and $\mathbf{v}$
end if
repeat
  $\mathbf{r} \leftarrow \mathbf{u} - \left\lceil \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{v}\|^2} \right\rfloor \mathbf{v}$
  $\mathbf{u} \leftarrow \mathbf{v}$
  $\mathbf{v} \leftarrow \mathbf{r}$
until $\|\mathbf{u}\| < \|\mathbf{v}\|$

LLL

The LLL-algorithm produces a basis that satisfies the LLL-reduction notion as in Definition 1.44. Hermite had attempted to create similar algorithms to reduce the basis of lattices in accordance with Hermite's inequality (1.9). However, his reduction notions were slightly different and it is unknown whether his algorithms are polynomial time for varying dimensions. What sets the LLL-algorithm apart from Hermite's algorithms is that LLL relaxes Hermite's inequality and that LLL swaps basis vectors one at a time, using Lovász' condition (1.16). The following algorithm is LLL in its simplest form. It takes a lattice basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ as its input.

Algorithm 3 LenstraLenstraLovász lattice reduction algorithm

Size-reduce $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ (using algorithm 1)
if there exists an index i violating Lovász' condition (1.16) **then**
  swap $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ and return to the first step
end if

Theorem 1.47. *Let $\delta \in (\frac{1}{4}, 1)$. For $\mathbf{b}_i \in \mathbb{Q}^n$, Algorithm 3 computes a basis that is LLL-reduced with factor δ in time that is polynomial in the lattice rank d, the space dimension n and the maximal bit-length of the $\mathbf{b}_i$'s.*

This theorem can be proven by analyzing what happens during the vector swaps. A sketch of the proof will be given here, rather than the full proof, which can be found in the original paper [54]. Assume for simplicity that the $\mathbf{b}_i \in \mathbb{Z}^n$ for all i. When $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ are swapped, only their GSO vectors change. Let $\mathbf{c}_i^*$ and $\mathbf{c}_{i+1}^*$ be the new GSO vectors after swapping. Because the product of the length GSO vectors is equal to the lattice volume, it follows that

$$\|\mathbf{c}_i^*\| \cdot \|\mathbf{c}_{i+1}^*\| = \|\mathbf{b}_i^*\| \cdot \|\mathbf{b}_{i+1}^*\|.$$

Lovász' condition was violated, which implies that $\|\mathbf{c}_i^*\|^2 < \delta \|\mathbf{b}_i^*\|^2$. Combining these facts gives

$$\|\mathbf{c}_i^*\|^{2(d-i+1)} \cdot \|\mathbf{c}_{i+1}^*\|^{2(d-i)} < \delta \|\mathbf{b}_i^*\|^{2(d-i+1)} \cdot \|\mathbf{b}_{i+1}^*\|^{2(d-i)}.$$

Now consider the quantity

$$D = \|\mathbf{b}_1^*\|^{2d} \|\mathbf{b}_2^*\|^{2(d-1)} \cdot \ldots \cdot \|\mathbf{b}_d^*\|^2,$$

which decreases by a factor $\delta < 1$ every time vectors are swapped. Furthermore, D is the product of d Gram determinants D_i, where $D_i = \Delta(\mathbf{b}_1, \ldots, \mathbf{b}_i)$ for $1 \leq i \leq d$. It follows that D is an integer (since $\mathbf{b}_i \in \mathbb{Z}^n$) and can therefore only decrease a logarithmic number of times in the initial value of D. This means that only that many swaps can happen, which is upper bounded by $(\max\{\|\mathbf{b}_1\|, \ldots, \|\mathbf{b}_d\|\})^{2d}$. It now remains to be shown that the size of the rational coefficients μ_{ij} and $\|\mathbf{b}_i^*\|^2$ can be bounded. This can be done by closely examining the D_i.

The LLL-algorithm was a breakthrough in lattice reduction, since it was the first algorithm provably running in polynomial time, even in the dimension of the lattice. Several improvements

have been made to the algorithm since, but those will not be described here. The performance of LLL is remarkably good in the lower dimensions, and especially when the shortest vector is much shorter than the expected shortest length $\sigma(L)$ as defined in Equation (1.7). For instance, in lattices where the *lattice gap* $\lambda_2(L)/\lambda_1(L)$ is big, the shortest vector will generally be much shorter than the expected length $\sigma(L)$. The LLL-algorithm will perform above expectation in such lattices. In fact, the LLL-algorithm has performed much better in practice than the theoretical bounds of Theorem 1.45 would suggest. This has caused some commotion in the field of cryptography, where lattices have been used to break systems due to the unexpectedly good performance of lattice reduction algorithms such as LLL in the lower dimensions.

Example 1.48. *This example shows Algorithm 3 in action for $\delta = 3/4$. The algorithm is applied to the lattice spanned by the columns of the following basis matrix:*

$$\begin{pmatrix} 2 & 0 & -4 \\ 4 & 2 & -2 \\ -2 & 2 & 2 \end{pmatrix}$$

The first step is to size-reduce the basis using Algorithm 1. The result is the following matrix:

$$\begin{pmatrix} 2 & 0 & -2 \\ 4 & 2 & 2 \\ -2 & 2 & 0 \end{pmatrix}$$

Lovász' condition does not hold for $i = 1$, since

$$\|\mathbf{b}_2^* + \mu_{2,1}\mathbf{b}_1^*\|^2 = \|\mathbf{b}_2\|^2 = 8 < 18 = \frac{3}{4}\|\mathbf{b}_1^*\|^2.$$

Thus, columns 1 and 2 are swapped, resulting in the basis matrix

$$\begin{pmatrix} 0 & 2 & -2 \\ 2 & 4 & 2 \\ 2 & -2 & 0 \end{pmatrix}.$$

This basis is already size-reduced, but Lovász' condition is violated for $i = 2$, since

$$\|\mathbf{b}_3^* + \mu_{3,2}\mathbf{b}_2^*\|^2 = 6 < 33/2 = \frac{3}{4}\|\mathbf{b}_2^*\|^2.$$

Thus, columns 2 and 3 are swapped, resulting in the basis matrix

$$\begin{pmatrix} 0 & -2 & 2 \\ 2 & 2 & 4 \\ 2 & 0 & -2 \end{pmatrix}.$$

This basis is size-reduced and does not violate Lovász' condition. Thus, it is LLL-reduced with factor $3/4$.

1.5.4 Babai's methods for CVP

Lattice reduction methods provide a basis of reasonably short vectors and can hence be used to approximate the shortest vector problem. However, it is also possible to use the results of lattice reduction to create algorithms that approximate the closest vector problem. Babai published two methods to approximate CVP in 1986 [7], the rounding method and the nearest plane algorithm. They are both approximation algorithms for CVP with provable approximation factors thanks to the LLL algorithm. The first step of both algorithms is to apply LLL to the given basis in order to reduce the length of the basis vectors. The underlying methods can be applied using any basis for the lattice, however. The nearest plane algorithm gives a better approximation factor than

the rounding method, but the rounding method is simpler. Only the rounding method will be described here.

Let $L \subset \mathbb{Z}^n$ be a d-rank lattice with basis B, and let $\mathbf{x} \in \text{span}(L)$ be the target vector. The rounding method can concisely be described by the computation

$$\mathbf{u} = B\lfloor B^{-1}\mathbf{x} \rceil.$$

The idea of this method is as follows: since $\mathbf{x} \in \text{span}(L)$ and B is a basis for L, it can be written as a unique linear combination of the basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$:

$$\mathbf{x} = \sum_{i=1}^{d} \lambda_i \mathbf{b}_i,$$

where $\lambda_i \in \mathbb{R}$. Now, the coordinate vector of $\mathbf{x}$ with respect to the basis B is given by $B^{-1}\mathbf{x} = (\lambda_1, \ldots, \lambda_d)$. In Babai's rounding method, each λ_i is rounded to the nearest integer $\lfloor \lambda_i \rceil$. The resulting lattice vector $\mathbf{u}$ is computed by taking the linear combination of the $\mathbf{b}_i$'s with coefficients $\lfloor \lambda_i \rceil$:

$$\mathbf{u} = \sum_{i=1}^{d} \lfloor \lambda_i \rceil \mathbf{b}_i = B\lfloor B^{-1}\mathbf{x} \rceil.$$

Babai showed that if the basis is LLL-reduced, then the rounding method finds an approximate solution to the CVP-instance within an approximation factor of $\gamma(d) = 1 + 2d(9/2)^{d/2}$. This is a reasonable approximation factor, although it is still exponential in the lattice rank.

To show that Babai's rounding method does not always find the right answer when the basis is bad, consider the following example. Let $L = \mathbb{Z}^2$ be the integral lattice in dimension 2. Now consider the CVP-instance given by the basis $\mathbf{b}_1 = (1, 1)$ and $\mathbf{b}_2 = (1, 0)$ of L and the target vector $\mathbf{x} = (13/8, 1/4)$. Now, $B^{-1}\mathbf{x} = (1/4, 11/8)$, or equivalently, $\frac{1}{4}\mathbf{b}_1 + \frac{11}{8}\mathbf{b}_2 = \mathbf{x}$. Rounding these coordinates gives $\lfloor B^{-1}\mathbf{x} \rceil = (0, 1)$, which leads to the lattice vector $0\mathbf{b}_1 + 1\mathbf{b}_2 = (1, 0)$. However, the closest lattice point to $\mathbf{x}$ is $2\mathbf{b}_2 = (2, 0)$. Figure 1.7 depicts this example.

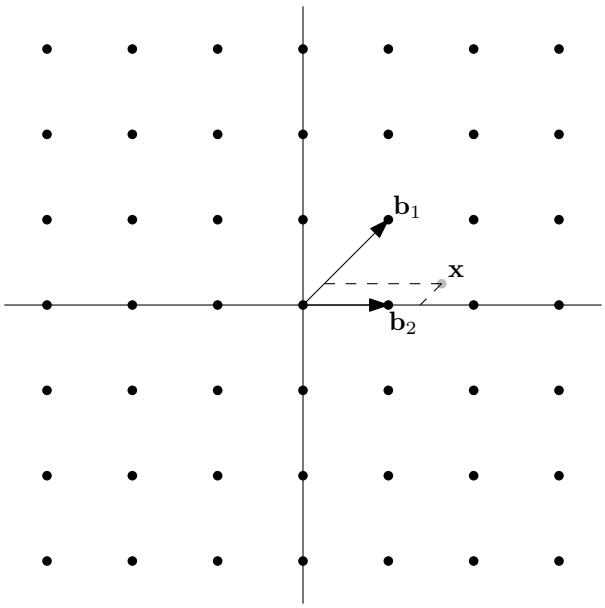

Figure 1.7: Example of Babai's rounding method.

1.6 Additional lattice problems

In his worst-case to average-case reductions, Ajtai [1] introduced three lattice problems, such that a solution to the shortest vector problem in a certain class of random lattices implies a solution to any instance of these three problems. Those problems will be introduced in this section, as well as the class of random lattices that was used. Recall that these worst-case to average-case reductions are desirable properties for cryptosystems based on such problems. Additionally, the Hermite Shortest Vector Problem will be described here, as an alternative to SVP.

1.6.1 Hermite Shortest Vector Problem

Recall that the aim of SVP is to find a shortest nonzero vector in a lattice. But given a vector, how does one check whether this is the shortest? To circumvent this issue, the Hermite Shortest Vector Problem does not aim to find a short vector relative to the shortest nonzero vector, but instead relative to the volume, as in the definition of Hermite's constant (Definition 1.31):

Definition 1.49 (Hermite Shortest Vector Problem (HSVP))**.** *Given a basis B of a d-rank lattice $L \subseteq \mathbb{Z}^n$ and an approximation factor $\gamma > 0$, find a non-zero lattice vector $\mathbf{v}$ such that the norm of $\mathbf{v}$ satisfies $\|\mathbf{v}\| \leq \gamma(\mathrm{vol}(L))^{1/d}$.*

Since the volume of a lattice is easily computed, verifying a solution becomes easy as well. The definition of Hermite's constant guarantees a solution to this problem for approximation factor $\gamma \geq \gamma_d$, where γ_d is Hermite's constant for rank d as in Definition 1.31.

Recall from Theorem 1.45(i) that the LLL-algorithm finds a basis such that the first vector satisfies $\|\mathbf{b}_1\| \leq \alpha^{(d-1)/4}(\mathrm{vol}(L))^{1/d}$ for some constant $\alpha > 4/3$. Therefore, the LLL-algorithm can solve HSVP for approximation factor $\gamma = \alpha^{(d-1)/4}$.

1.6.2 Shortest Length Problem

As mentioned in the description of HSVP, it is not trivial to check whether a nonzero vector is actually the shortest vector. If the length of a shortest nonzero vector is known, checking whether a vector is a shortest vector becomes trivial. Is it possible to determine the first minimum $\lambda_1(L)$ of a lattice L? This question gives rise to the first problem in Ajtai's paper:

Definition 1.50 (Shortest Length Problem (SLP))**.** *Given a basis B of a d-rank lattice $L \subseteq \mathbb{Z}^n$ and an approximation factor $\gamma > 0$, find a value $\lambda(L)$ such that $\lambda_1(L) \leq \lambda(L) \leq \gamma\lambda_1(L)$.*

Taking $\gamma = 1$ as the approximation factor gives the exact version of the problem. In Ajtai's paper, the approximation factor γ is polynomial (in n and d).

1.6.3 Unique Shortest Vector Problem

The second problem is a special case of the shortest vector problem that puts certain restrictions on the lattice L.

Definition 1.51 (Unique Shortest Vector Problem (uSVP))**.** *Given a basis B of a full-rank lattice $L \subset \mathbb{Z}^n$ and a gap factor γ, find the shortest non-zero lattice vector $\mathbf{v} \in L$, where $\mathbf{v}$ is unique in the sense that any other vector $\mathbf{x} \in L$ with $\|\mathbf{x}\| \leq \gamma\|\mathbf{v}\|$ is an integral multiple of $\mathbf{v}$.*

Equivalently, this is the regular shortest vector problem restricted to the lattices where the gap $\lambda_2(L)/\lambda_1(L) > \gamma$, where $\lambda_i(L)$ is the i'th successive minimum of L. In his works, Ajtai considers the unique shortest vector problem with polynomial gap of the form n^c for some integer c.

1.6.4 Shortest Basis Problem

The third problem is similar to a basis reduction problem. Let $\mathcal{B}(L)$ be the set of all bases for the lattice L.

Definition 1.52 (Shortest Basis Problem (SBP))**.** *Given a d-rank lattice $L \subset \mathbb{Z}^n$ and an approximation factor $\gamma > 0$, find a basis $B = [\mathbf{b}_1, \ldots, \mathbf{b}_d]$, such that*

$$\max_{1 \leq i \leq d} \|\mathbf{b}_i\| \leq \gamma \max_{1 \leq i \leq d} \|\mathbf{b}'_i\|,$$

for all bases $B' = [\mathbf{b}'_1, \ldots, \mathbf{b}'_d] \in \mathcal{B}(L)$.

The exact version of the problem is obtained by taking $\gamma = 1$. In Ajtai's paper, γ is a polynomial approximation factor (in n and d).

1.6.5 Modular lattices

Ajtai's class of random lattices consists of so-called *modular* or *q-ary* lattices L, which satisfy $q\mathbb{Z}^n \subseteq L \subseteq \mathbb{Z}^n$ for some integer q. Now, this is not a useful class of lattices by itself, as every lattices is modular for some q. Specifically, when q is an integer multiple of the volume of L, then L is q-ary. In the following, lattices L that are modular for $q << \mathrm{vol}(L)$ will be considered.

Modular lattices have the following property. If $\mathbf{x} \equiv \mathbf{y} \bmod q$, then $\mathbf{x} \in L$ if and only if $\mathbf{y} \in L$. In his paper, Ajtai only considers lattices of the form

$$\Lambda_q^{\perp}(A) = \{\mathbf{x} \in \mathbb{Z}^m : A\mathbf{x} = 0 \pmod q\},$$

where A is an $n \times m$ integral matrix with coefficients modulo q. Here, m is used to denote the number of columns rather than d, since it need not be smaller than n for modular lattices. The columns of the matrix need not even be linearly independent. For $m \geq n$, the lattice $\Lambda_q^{\perp}(A)$ is a full-rank lattice in $\mathbb{Z}^m$. It is also possible to consider the modular lattice spanned by the rows of A:

$$\Lambda_q(A) = \left\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x} = A^T\mathbf{y} \pmod q \text{ for some } \mathbf{y} \in \mathbb{Z}^n\right\}.$$

These lattices are dual in the sense that $\Lambda_q^{\perp}(A) = q \cdot \Lambda_q(A)^{\times}$ and $\Lambda_q(A) = q \cdot \Lambda_q^{\perp}(A)^{\times}$. Finding a short (not necessarily shortest) in the lattice $\Lambda_q^{\perp}(A)$ was later named the *Small Integer Solution* problem (SIS), but Ajtai did not explicitly define it as a separate problem. The SIS problem will be formally introduced in Chapter 3.

1.7 Knapsack-based cryptography

To show the strength of the LLL algorithm, as well as the pitfalls of building trapdoors in hard problems, this section will give an example of a type of cryptosystem that was broken by LLL using the weaknesses created by the trapdoors. This example appeared in [18] and [40]. In 1978, Merkle and Hellman published an article [33], attempting to base a cryptosystem on the subset sum problem, which is known to be NP-complete. The subset sum problem is to find a subset of a given set of positive integers $\{z_1, \ldots, z_n\}$, such that the elements in the subset sum up to some given integer S. This problem is a special case of the knapsack problem, and knapsack-based cryptosystems are generally based on the subset sum problem.

The idea was to encrypt a binary message vector $\mathbf{m}$ by taking the inner product with the vector $\mathbf{z} = (z_1, \ldots, z_n)$, and to let the resulting integer $S = \langle \mathbf{m}, \mathbf{z} \rangle$ be the ciphertext. But now it becomes difficult to retrieve the $\mathbf{m}$ from the integer S, and it might not even be a unique solution to the subset sum problem. Merkle and Hellman needed to create a trapdoor to address these issues, which is generally a difficult task when turning hard problems into cryptosystems.

For their trapdoor, Merkle and Hellman decided to restrict the subset sum problem to an easier class of instances. Instead of using general integers z_i, they used a *super-increasing* sequence of

integers $\mathbf{r} = \{r_1, \ldots, r_n\}$, where super-increasing means that $r_{i+1} \geq 2r_i$ for all $1 \leq i < n$. A consequence of this lower bound is that for all i,

$$r_i \geq 2r_{i-1} \geq r_{i-1} + 2r_{i-2} \geq \ldots \geq \sum_{j=1}^{i-1} r_j + r_1 > \sum_{j=1}^{i-1} r_j.$$

Since for all i, r_i is greater than the sum of all previous elements, the subset sum problem becomes easy to solve. For a target integer S, just find the biggest r_i such that $r_i \leq S$. Since $\sum_{j=1}^{r-1} r_j < r_i \leq S$, r_i must be used in the sum to get S. Just set $\mathbf{m}_i = 1$ and continue the procedure with $S - r_i$. Thus, anyone who possesses the sequence $\mathbf{r}$ can use this trapdoor to decrypt the message $\mathbf{m}$.

The cryptosystem was constructed as follows: Take a secret super-increasing sequence $\mathbf{r}$, as well as two secret integers A and B, such that $B > 2r_n$ and $\gcd(A, B) = 1$. Since $B > 2r_n$, B will also be greater than the sum of any subset of the r_i's, and therefore greater than any possible solution of the subset sum problem. Next, the sequence $\mathbf{r}$ is transformed into the public key by calculating $\mathbf{Z} = A\mathbf{r} \mod B$. The public key is basically a new instance of the subset sum problem with $Z_i = Ar_i \mod B$ for all i, where $\mathbf{Z}$ is generally not super-increasing.

Messages are now encrypted into this public subset sum instance as before. The ciphertext becomes the integer $S = \langle \mathbf{m}, \mathbf{Z} \rangle$, where $\mathbf{m}$ is the binary message vector. To break the encryption, one would need to solve this instance of the subset sum problem. To decrypt a ciphertext S, it is multiplied by $A^{-1} \mod B$ to obtain $S' = A^{-1}\langle \mathbf{m}, A\mathbf{r} \rangle \equiv \langle \mathbf{m}, \mathbf{r} \rangle \mod B$. Because it is known that the solution $S = \langle \mathbf{m}, \mathbf{r} \rangle < B$, the exact value of $\langle \mathbf{m}, \mathbf{r} \rangle$ is obtained by taking $0 \leq S' \leq B - 1$. Since $\mathbf{r}$ is super-increasing, $\mathbf{m}$ can easily be recovered as described.

The transformation of the private key to the public key can be adapted at will. The version described here is sometimes called the single-iterated Merkle-Hellman knapsack cryptosystem, because only one pair (A, B) was used, but it is possible to transform the instance several times, each time picking a B_i greater than the maximal sum of the subset sum instance $i - 1$. Other transformations can be used as well, but generally modular linear operations are used to keep the running time of the decryption low.

The density of a sequence $\mathbf{x} = (x_1, \ldots, x_n)$ is defined as $n/\log(\max_i x_i)$, and the density of the knapsack is defined as the density of the sequence of the corresponding subset sum problem. The bigger the value of r_n in a super-increasing sequence, the lower the density of the knapsack. For security reasons, the smallest value of the private key r_1 must not be taken too small, or else combinatorial attacks can be used to break the system. Furthermore, a bigger r_1 results in a bigger r_n and hence in a knapsack of lower density. Now consider what happens to the density of the knapsack when being transformed using an iteration in the Merkle-Hellman system. The values $0 \leq \mathbf{x}_1, \ldots, \mathbf{x}_n \leq B_i$ have already had $i - 1$ transformations and are roughly uniformly distributed over the interval $[0, B_i)$. Since B_{i+1} needs to be bigger than the sum of the x_i's, the expectation is that $B_{i+1} > \frac{n}{2} B_i$. This means that each iteration is expected to lower the density of the knapsack, so they cannot continue indefinitely if the density is to be kept high.

This is particularly interesting because it turns out that low-density knapsacks are vulnerable to an attack using lattices. Consider an instance of a low-density knapsack cryptosystem with public instance $(Z_1, \ldots, Z_n)$. The Z_i are expected to be reasonably high-valued due to the low density of the knapsack. Let $\mathbf{m} = (m_1, \ldots, m_n)$ be a binary message vector with ciphertext $S = \langle \mathbf{m}, \mathbf{Z} \rangle$. It is now possible to construct the lattice spanned by the columns of the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ Z_1 & Z_2 & Z_3 & \cdots & Z_n & S \end{pmatrix}. \tag{1.18}$$

Note that multiplying the vector $(m_1, m_2, \ldots, m_n, -1)$ with the matrix in (1.18) results in the vector $\mathbf{v} = (m_1, m_2, \ldots, m_n, \langle \mathbf{m}, \mathbf{Z} \rangle - S) = (m_1, m_2, \ldots, m_n, 0)$ by design. This vector has norm $\|\mathbf{v}\| \leq \sqrt{n}$, because $\mathbf{m}$ is a binary vector. Thus, $\mathbf{v}$ is very likely to be the shortest vector in the lattice (which has volume S) since it is very unlikely that a vector as short as $\mathbf{v}$ is a linear combination of vectors that are as long as the columns of the matrix in (1.18).

The next step in the attack is to reduce the lattice and hope the correct short $(0, 1)$-vector is found. It is possible that there are many very small non-binary vectors in this lattice, sometimes called *parasitic solutions*. Lagarias and Odlyzko showed in [51] that for randomly chosen weights the LLL-algorithm will find the correct solution in almost all problems of density $d < 1/n$ with high probability. In [12], Coster et al. improved the attack by replacing the last column by the vector $(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}, S)$. Using the same assumptions as Lagarias and Odlyzko, they showed the attack would work for problems with density $d < 0.9408$.

Both these attack methods require something to solve the SVP. Since the LLL-algorithm performs well in low dimensions, the systems can be broken for small n. The subset sum problem in high-density knapsacks remains hard in general, but for the practical dimensions it is not hard enough. Furthermore, the Merkle-Hellman system suffers from the fact that it requires iterations to defeat combinatorial attacks, but these iterations lower the density of the problem. The knapsack-based cryptosystems form a good example of the fact that it is difficult to turn a provably hard problem into a cryptosystem, and that not every instance of a hard problem is hard.

Chapter 2

Early Lattice-based Cryptography

2.1 Introduction

As discussed in Chapter 1, there are several hard lattice problems that appear suitable for public key cryptography. One of the reasons is Ajtai's discovery of a connection between the worst-case and average-case of these problems. This chapter contains a description of the first three cryptosystems that were associated with lattices. They were proposed in 1996 and 1997, and two of the systems were subsequently broken in 1998 and 1999.

In Section 2.2, the Ajtai-Dwork cryptosystem will be described, including a description of the attack that retrieves the secret key. Then, the Goldreich-Goldwasser-Halevi cryptosystem is explained in Section 2.3, as well as the attack that was used to break the encryption. Finally, Section 2.4 gives a description of the NTRU cryptosystem, including an explanation of the underlying algebraic concepts.

2.2 Ajtai-Dwork

The Ajtai-Dwork cryptosystem (AD) was introduced in 1997 in [4]. As opposed to the other two systems described in this chapter, the AD cryptosystem was accompanied by a security proof. Unfortunately, it is quite inefficient compared to other public-key cryptosystems. Although the system was not presented using lattices, the security proof showed that every instance of the *unique shortest vector problem* could be transformed into a random instance of their cryptosystem with high probability.

2.2.1 Parameters and setup

The AD system is defined in a Euclidean vector space setting, using the standard Euclidean norm. The security parameter n determines the dimension of the vector space. Given n, let $m = n^3$, and $r_n = n^n$. For these parameters, denote the n-dimensional cube with sides of length r_n by

$$B_n = \{\mathbf{x} \in \mathbb{R}^n : |x_i| \leq r_n/2, \, \forall i\} .$$

Furthermore, for some integer $c > 0$, denote the n-dimensional ball of radius n^{-c} by

$$S_n = \left\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq n^{-c}\right\} .$$

In the original proposal $c = 8$ was used. Here, c is chosen to be 9, in order to simplify some of the proofs. This does not change the essence of the security result. The precision of the binary expansion of the real numbers will be n bits, as in the original proposal.

The private key is a vector $\mathbf{u}$ that is chosen randomly from the n-dimensional unit ball. Then, given a private key $\mathbf{u}$, the distribution $\mathcal{H}_{\mathbf{u}}$ is defined on B_n using the following construction:

$\mathcal{H}_{\mathbf{u}}$:

1. Take $\mathbf{x}$ from $\{\mathbf{x} \in B_n : \langle \mathbf{x}, \mathbf{u} \rangle \in \mathbb{Z}\}$, uniformly at random.
2. Draw n error vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$ from S_n, independently and uniformly at random.
3. Output $\mathbf{v} = \mathbf{x} + \sum_{i=1}^{n} \mathbf{y}_i$.

To obtain the public key, $n + m$ vectors $\mathbf{w}_1, \ldots, \mathbf{w}_n, \mathbf{v}_1, \ldots, \mathbf{v}_m$ are taken randomly from $\mathcal{H}_{\mathbf{u}}$. The $\mathbf{w}_i$'s must satisfy that the parallelepiped that they span, denoted by $\mathcal{P}(\mathbf{w}_1, \ldots, \mathbf{w}_n)$, is not too flat, i.e., the minimum distance of $\mathbf{w}_i$ to the hyperplane spanned by the other $\mathbf{w}_j$'s, denoted by $H_{j \neq i}$, must be at least r_n/n^2, for all i. If this is not the case, a new key is generated. A vector $\mathbf{v}$ can be reduced modulo the parallelepiped by finding a vector $\mathbf{v}'$, such that the difference $\mathbf{v} - \mathbf{v}'$ is a vector in the lattice spanned by the $\mathbf{w}_i$'s.

Parameter	Description	Knowledge
n	Dimension	Public
$m = n^3$	Integer bound	Public
$r_n = n^n$	Integer bound	Public
$\mathbf{u}$	n-dimensional vector	Private
$\mathbf{w}_1, \ldots, \mathbf{w}_n, \mathbf{v}_1, \ldots, \mathbf{v}_m$	$n + m$ n-dimensional vectors	Public

Table 2.1: Parameters of AD.

2.2.2 Encryption and decryption

Let the parameters of the system be defined as in Table 2.1. Then, encryption and decryption are performed as follows.

Encryption

The encryption is performed on one bit at a time. To encrypt a zero bit, take $b_1, \ldots, b_m$ uniformly at random from $\{0, 1\}$ and reduce the vector $\sum_i b_i \mathbf{v}_i$ modulo the parallelepiped $\mathcal{P}(\mathbf{w}_1, \ldots, \mathbf{w}_n)$. The resulting n-dimensional vector will be the ciphertext. To encrypt a one bit, randomly choose an n-dimensional vector in the parallelepiped $\mathcal{P}(\mathbf{w}_1, \ldots, \mathbf{w}_n)$ and take this as the ciphertext.

Decryption

To decrypt a ciphertext $\mathbf{c}$, which is an n-dimensional vector corresponding to a single bit of plaintext, compute the inner product with the private key $\mathbf{u}$. If $\text{dist}(\langle \mathbf{c}, \mathbf{u} \rangle, \mathbb{Z}) \leq n^{-1}$, then $\mathbf{c}$ is decrypted as zero and it is decrypted as one otherwise.

2.2.3 Why it works

The idea behind $\mathcal{H}_{\mathbf{u}}$ is that the points in the resulting distribution are all 'close to' the secret or hidden hyperplanes $H_i = \{\mathbf{h} \in \mathbb{R}^n : \langle \mathbf{h}, \mathbf{u} \rangle = i\}$. Here, 'close to' is quantified by taking the inner product with $\mathbf{u}$, and seeing if the result is 'close to' $\mathbb{Z}$, in the sense that the inner product lies in the interval $[z - \varepsilon, z + \varepsilon]$ for some integer z and $\varepsilon > 0$.

Flatness of the parallelepiped

During the encryption of a zero bit, a vector is reduced modulo the parallelepiped spanned by the $\mathbf{w}_j$, which cannot be too flat. To see the reason of this restriction on the $\mathbf{w}_j$'s, consider what

happens when a sum of some of the $\mathbf{v}_i$'s is reduced modulo the parallelepiped $\mathcal{P}(\mathbf{w}_1, \ldots, \mathbf{w}_n)$. The result is a vector $\mathbf{v}'$ of the form

$$\mathbf{v}' = \sum_{i=1}^{m} b_i \mathbf{v}_i + \sum_{j=1}^{n} \lambda_j \mathbf{w}_j, \tag{2.1}$$

where $b_i \in \{0, 1\}$ for $1 \leq i \leq m$ and $\lambda_j \in \mathbb{Z}$ for $1 \leq j \leq n$. This vector lies in the parallelepiped $\mathcal{P}(\mathbf{w}_1, \ldots, \mathbf{w}_n)$, and the vectors $\mathbf{w}_j$ lie in the cube with side length r_n (disregarding the insignificant error vectors $\mathbf{y}_i$), which bounds their norm by $\sqrt{n} r_n/2$. Therefore, the norm of $\mathbf{v}'$ is bounded by

$$\|\mathbf{v}'\| \leq \sum_{j=1}^{n} \|\mathbf{w}_j\| \leq n\sqrt{n} r_n/2. \tag{2.2}$$

Now consider the vector $\mathbf{v}' - \sum_i b_i \mathbf{v}_i$. Since the norm of the $\mathbf{v}_i$'s satisfies the same bound as those of the $\mathbf{w}_j$'s, it follows from (2.2) that

$$\|\mathbf{v}' - \sum_i b_i \mathbf{v}_i\| \leq \|\mathbf{v}'\| + \sum_i \|\mathbf{v}_i\| \leq (n\sqrt{n} + n^3\sqrt{n}) r_n/2. \tag{2.3}$$

Take any $1 \leq k \leq n$, and then rewrite (2.1) as follows:

$$\begin{aligned} \lambda_k \mathbf{w}_k - (\mathbf{v}' - \sum_i b_i \mathbf{v}_i) &= -\sum_{j \neq k} \lambda_j \mathbf{w}_j, \\ \mathbf{w}_k - \lambda_k^{-1}(\mathbf{v}' - \sum_i b_i \mathbf{v}_i) &= -\lambda_k^{-1} \sum_{j \neq k} \lambda_j \mathbf{w}_j. \end{aligned} \tag{2.4}$$

Now, the right hand side of (2.4) is in the hyperplane $H_{j \neq k}$ spanned by the $\mathbf{w}_j$'s for $j \neq k$. By construction, the distance between $\mathbf{w}_k$ and this hyperplane is lower bounded by r_n/n^2. Combining this with (2.3) and (2.4) gives

$$\begin{aligned} r_n/n^2 \leq \operatorname{dist}(\mathbf{w}_k, H_{j \neq k}) &\leq \lambda_k^{-1} \|\mathbf{v}' - \sum_i b_i \mathbf{v}_i\| \\ &\leq \lambda_k^{-1}(n\sqrt{n} + n^3\sqrt{n}) r_n/2. \end{aligned}$$

This can then be rewritten as

$$\lambda_k \leq n^3\sqrt{n}(1 + n^2), \tag{2.5}$$

for $n > 2$. Note that (2.5) must hold for all $1 \leq k \leq n$. This result will be used to show that the encryption of a zero is always decrypted as a zero.

Encryption of zeroes

The encryption of a zero bit is a linear combination of vectors that are all close to the hidden hyperplanes. Therefore, this encryption lies reasonably close to the hidden hyperplanes as well. On the other hand, the encryption of a one bit is randomly distributed, and with a reasonably high probability (depending on n) it will not lie close to a hidden hyperplane. As mentioned, vectors close to a hyperplane will have an inner product that is close to $\mathbb{Z}$.

By design, an encrypted zero will always be decrypted as a zero. To see this, consider

$$\mathbf{c}_0 = \sum_{i=1}^{m} b_i \mathbf{v}_i + \sum_{j=1}^{n} \lambda_j \mathbf{w}_j, \tag{2.6}$$

where the b_i are bits and the λ_j are integers. Now consider the fact that the $\mathbf{v}$'s and the $\mathbf{w}$'s come from the distribution $\mathcal{H}_{\mathbf{u}}$:

$$\mathbf{v}_i = \mathbf{x}_i^{(1)} + \sum_{k=1}^{n} \mathbf{y}_{i,k}^{(1)} \tag{2.7}$$

$$\mathbf{w}_j = \mathbf{x}_j^{(2)} + \sum_{l=1}^{n} \mathbf{y}_{j,l}^{(2)}, \tag{2.8}$$

where $\langle \mathbf{x}_i^{(*)}, \mathbf{u} \rangle \in \mathbb{Z}$ for all i, and where $\|\mathbf{y}_{i,j}^{(*)}\| \leq n^{-9}$ for all i and j. Combining (2.6), (2.7) and (2.8) gives

$$\begin{aligned}\langle \mathbf{c}_0, \mathbf{u} \rangle &= \sum_{i=1}^{m} b_i \langle \mathbf{v}_i, \mathbf{u} \rangle + \sum_{j=1}^{n} \lambda_j \langle \mathbf{w}_j, \mathbf{u} \rangle \\ &= \sum_{i=1}^{m} b_i \left(\langle \mathbf{x}_i^{(1)}, \mathbf{u} \rangle + \sum_{k=1}^{n} \langle \mathbf{y}_{i,k}^{(1)}, \mathbf{u} \rangle \right) + \sum_{j=1}^{n} \lambda_j \left(\langle \mathbf{x}_j^{(2)}, \mathbf{u} \rangle + \sum_{l=1}^{n} \langle \mathbf{y}_{j,l}^{(2)}, \mathbf{u} \rangle \right).\end{aligned}$$

Furthermore, since $\langle \mathbf{x}_i^{(*)}, \mathbf{u} \rangle \in \mathbb{Z}$ for all i, and $\|\mathbf{y}_i^{(*)}\| \leq n^{-9}$ for all i, applying the triangle inequality gives

$$\begin{aligned}\operatorname{dist}(\langle \mathbf{c}_0, \mathbf{u} \rangle, \mathbb{Z}) &\leq \sum_{i=1}^{m} \sum_{k=1}^{n} \operatorname{dist}(\langle \mathbf{y}_{i,k}^{(*)}, \mathbf{u} \rangle, \mathbb{Z}) + \sum_{j=1}^{n} \lambda_j \sum_{l=1}^{n} \operatorname{dist}(\langle \mathbf{y}_{j,l}^{(*)}, \mathbf{u} \rangle, \mathbb{Z}) \\ &\leq n^4 n^{-9} + \lambda_{\max} n^2 n^{-9} < n^{-5} + n^{-5/2} + n^{-3/2} < n^{-1},\end{aligned} \tag{2.9}$$

where $\lambda_{\max} = \max_j \lambda_j \leq n^3 \sqrt{n}(1 + n^2)$, by (2.5).

Encryption of ones

The encryption of ones is taken randomly from the parallelepiped $\mathcal{P}(\mathbf{w}_1, \ldots, \mathbf{w}_n)$. Let $\mathbf{c}_1$ be such an encryption. There is a small probability that $\mathbf{c}_1$ accidentally ends up close to the hidden hyperplanes, causing a decryption error. Indeed, consider the fractional part $\langle \mathbf{c}_1, \mathbf{u} \rangle - \lfloor \langle \mathbf{c}_1, \mathbf{u} \rangle \rfloor$. As $\mathbf{c}_1$ was chosen uniformly random, this fractional part will also be roughly uniform on the interval $[0, 1]$. If this fractional part is smaller than n^{-1} or larger than $1 - n^{-1}$, $\mathbf{c}_1$ will be decrypted as a zero. Hence, the probability of a decryption error is approximately $2n^{-1}$. These decryption errors can be eliminated by changing the encryption procedure of ones, as shown by Goldreich, Goldwasser and Halevi [29].

Example 2.1. *The aim of this example is to show the system in action for dimension $n = 3$. Given this dimension, the other parameters become $m = 27$ and $r_3 = 27$. All values in this example were calculated using machine precision, but for the sake of simplifying the exposition they will not be given fully.*

First, a private key is randomly generated from the unit ball:

$$\mathbf{u} = (-0.524655, 0.606024, 0.36764).$$

This $\mathbf{u}$ determines the hyperplanes, and the public key $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{v}_1, \ldots, \mathbf{v}_{27})$ is derived by randomly taking vectors from within the cube with side length $r_3 = 27$ on these hyperplanes and adding 3 vectors from the 3-dimensional ball with radius $3^{-8} = 1/6561$, as described above. The resulting public key is shown in Table 2.2. Note that the inner products $\langle \mathbf{w}_j, \mathbf{u} \rangle$ and $\langle \mathbf{v}_i, \mathbf{u} \rangle$ lie within distance $1/3^7 = 2187 \approx 0.00046$ of $\mathbb{Z}$.

To check whether the parallelepiped spanned by $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ is not too flat, the least-squares problem was solved to find a vector closest to $\mathbf{w}_i$ in the span of the other two. From this vector,

$\mathbf{w}_1$	(7.60816, 1.84201, −8.49921)
$\mathbf{w}_2$	(−7.37496, −4.58596, −2.96519)
$\mathbf{w}_3$	(−10.8126, 7.34985, 13.2545)
$\mathbf{v}_1$	(−9.87709, −1.47613, 12.8186)
$\mathbf{v}_2$	(8.17359, −7.28432, 1.91144)
$\mathbf{v}_3$	(3.82588, 10.4402, 4.57008)
$\mathbf{v}_4$	(−7.44063, −2.91462, −11.2543)
$\mathbf{v}_5$	(−4.52958, 3.59412, 1.21152)
$\mathbf{v}_6$	(9.41711, −5.15847, 5.6217)
$\mathbf{v}_7$	(−10.7713, 13.1401, −7.11104)
$\mathbf{v}_8$	(−11.6452, −12.5039, −4.16762)
$\mathbf{v}_9$	(−4.45261, 11.9807, −12.5033)
$\mathbf{v}_{10}$	(3.98351, −6.5037, −10.7949)
$\mathbf{v}_{11}$	(−10.5945, 5.37026, −2.21125)
$\mathbf{v}_{12}$	(8.90485, 3.89708, 9.00435)
$\mathbf{v}_{13}$	(−0.186633, 9.97465, −11.2685)
$\mathbf{v}_{14}$	(10.1227, 10.7732, 10.2871)
$\mathbf{v}_{15}$	(4.89837, 9.93115, 9.65963)
$\mathbf{v}_{16}$	(−11.8204, −12.5187, 9.20751)
$\mathbf{v}_{17}$	(11.1477,−0.826234, 6.39036)
$\mathbf{v}_{18}$	(4.98783, −9.41796, 3.60244)
$\mathbf{v}_{19}$	(5.09802, −7.89569, −1.46938)
$\mathbf{v}_{20}$	(−6.02514, −8.08041,−0.718234)
$\mathbf{v}_{21}$	(11.9069, −10.9563, 13.2924)
$\mathbf{v}_{22}$	(−8.34388, 0.0449246, −11.9814)
$\mathbf{v}_{23}$	(−2.58845, 7.41754, 3.11883)
$\mathbf{v}_{24}$	(12.3989, −7.34945, −5.55127)
$\mathbf{v}_{25}$	(−8.40949, 8.06724, −6.25887)
$\mathbf{v}_{26}$	(−7.28547, 11.2038, −1.66535)
$\mathbf{v}_{27}$	(−6.41749, −7.77925, 3.66542)

Table 2.2: Public Key of AD in dimension 3

the distance was computed, resulting in the distances 4.83028, 7.12895 and 8.37488, for $\mathbf{w}_1, \mathbf{w}_2$ *and* $\mathbf{w}_3$*, respectively. As these distances are all at least* $r_3/3^2 = 3$*, the parallelepiped is not too flat.*

Now, the message $\mathbf{m} = (1, 0, 1, 1, 1, 1, 1, 0, 1)$ *is encrypted bit by bit as described, resulting in the ciphertext*

$$\mathbf{c} = ((9.986, 3.746, -2.791), (1.365, 1.417, -3.108), (-16.955, -1.992, 9.227), \tag{2.10}$$

$$(-5.223, -1.139, 1.278), (5.590, -3.151, -6.728), (-7.319, 9.134, 17.364), \tag{2.11}$$

$$(-3.014, 3.752, 2.509), (-9.874, 4.964, 7.645), (-9.039, 4.727, 5.035)), \tag{2.12}$$

where each triple corresponds to the encryption of one bit. Now, to decrypt, the inner product of each triple with $\mathbf{u}$ *is computed and it is then determined if the inner product lies close enough to* $\mathbb{Z}$*, i.e., if the distance to the nearest integer is less than* $n^{-1} = 1/3$*. These inner products are shown in Table 2.3.*

$\mathbf{c}_i$	$\langle \mathbf{c}_i, \mathbf{u} \rangle$
$\mathbf{c}_1$	3.994
$\mathbf{c}_2$	1.000
$\mathbf{c}_3$	-11.08
$\mathbf{c}_4$	2.519
$\mathbf{c}_5$	7.316
$\mathbf{c}_6$	-15.76
$\mathbf{c}_7$	-4.777
$\mathbf{c}_8$	-10.99
$\mathbf{c}_9$	-9.458

Table 2.3: Inner products of parts of the ciphertext with $\mathbf{u}$.

Those that are within the interval $(z - 1/3, z + 1/3)$ *for some integer* z *are decrypted as zeroes, and the others as ones, resulting in the decryption* $\mathbf{m}' = (0, 0, 0, 1, 0, 0, 0, 0, 1)$*. As expected, the zeroes were both successfully decrypted. However, note that out of 7 bits of one in* $\mathbf{m}$*, only 2 are decrypted correctly and 5 decryption errors have occurred. For* $n = 3$*, the decryption failure rate of ones is approximately* $2/3$*.*

2.2.4 Security proof

Before working on the cryptosystem, Ajtai showed a connection between the worst-case and average-case complexity of SVP [1]. He proved that SVP is NP-hard under randomized reductions [2]. The proof used a reduction to transform instances of the unique shortest vector problem to an approximation version of the SVP for random instances of a specific class of lattice. Inspired by this result, Ajtai and Dwork attempted to base a cryptosystem on the hardness of these problems.

For their cryptosystem, Ajtai and Dwork proved that being able to distinguish between the encryption of zeroes and ones implies being able to solve the unique shortest vector problem. The proof shows that if an encryption of a zero can be distinguished from an encryption of a one in polynomial time (without knowledge of the private key) with a probability of $n^{-c'}$ for some constant $c' > 0$, then the unique shortest vector problem has a probabilistic polynomial time solution in the worst-case. However, the hardness of the unique shortest vector problem was not as well understood as that of SVP. This gave rise to the question whether it was possible to base a cryptosystem on the worst-case hardness of SVP.

2.2.5 Attack on the encryption

Nguyen and Stern showed a converse to the security result of the AD cryptosystem in [78]. They reduced distinguishing between zero and one encryptions to approximating SVP within a factor

$n^{0.5-\varepsilon}$ or CVP within a factor $n^{1.33}$. Earlier, there had been a result by Goldreich and Goldwasser [28], that suggests that approximating CVP to within such a factor is unlikely to be NP-hard, unless the polynomial-time hierarchy collapses. Thus, it is likely that breaking AD is not NP-hard.

For decrypting with CVP, Nguyen and Stern construct an $(n+m)$-dimensional lattice in $\mathbb{R}^{2n+m}$ that incorporates the $\mathbf{w}_i$'s and $\mathbf{v}_i$'s, such that an encryption of zero will lie close to the lattice. A CVP-oracle can now be used to solve the decision variant of the CVP, which yields whether a zero or a one was encrypted. The SVP-reduction is more technical and will be omitted here. For more details on these constructions, see [78].

2.2.6 Attack on the private key

Nguyen and Stern also presented a heuristic attack to recover the secret vector $\mathbf{u}$, with a slightly different approach than the attack on the encryption. They construct a lattice in $\mathbb{R}^{n+m}$ depending only on the $\mathbf{v}_i$'s and then try to approximate the inner products $\langle \mathbf{v}_i, \mathbf{u} \rangle$ by making clever use of short vectors in this lattice. With enough of these approximations, the secret vector $\mathbf{u}$ can be sufficiently approximated to break the system.

The basis vectors of the lattice will be constructed by appending the m-dimensional unit vector $\mathbf{e}_i$ to $\beta \mathbf{v}_i$, where β is some real constant. This results in the m-dimensional lattice L_β spanned by the columns of the following $(n+m) \times m$ matrix:

$$M = \begin{pmatrix} \beta A \\ I_m \end{pmatrix},$$

where A is the $n \times m$ matrix that has the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$ as its columns and I_m is the $m \times m$ identity matrix.

Recall that by (2.7), the distance $\text{dist}(\langle \mathbf{v}_i, \mathbf{u} \rangle, \mathbb{Z}) \leq \sum_{k=1}^{n} \|\mathbf{y}_{i,k}\| \leq n^{-7}$. Define V_i as the closest integer to the inner product $\langle \mathbf{v}_i, \mathbf{u} \rangle$. Nguyen and Stern proved that short vectors in L_β give information on these V_i's. In the proof, they take some integral vector $\mathbf{x} = (x_1, \ldots, x_m)$ such that $M\mathbf{x}$ is short, and consider $\sum_{i=1}^{m} x_i V_i$. This sum is an integer, but Nguyen and Stern showed that whenever $\beta^2 \geq \frac{n^{14}}{2n^7 - 1} \geq \|M\mathbf{x}\|^2$, it must be small as well. They conclude that it must therefore be 0, which means that the vector $\mathbf{x}$ is orthogonal to the integral vector $V = (V_1, \ldots, V_m)$.

Consider the one-dimensional (integral) lattice spanned by the m-dimensional vector V, as well as its $(m-1)$-dimensional orthogonal lattice $V^\perp = \{\mathbf{a} \in \mathbb{Z}^m : \langle \mathbf{a}, V \rangle = 0\}$. Using this definition, $\mathbf{x} \in V^\perp$, since $\mathbf{x}$ is orthogonal to V.

Next, they prove that there are many such sufficiently short vectors in this lattice. They conjecture that there are at least $m-1$ short and linearly independent vectors in L_β. Note that, for all $\mathbf{x}$, the last m coordinates of $M\mathbf{x}$ are equal to $\mathbf{x}$. Thus, the $m-1$ short and linearly independent vectors $M\mathbf{x}_i \in L_\beta$ correspond to $m-1$ vectors $\mathbf{x}_i$, spanning the $(m-1)$-dimensional lattice $V^\perp$. Now, the orthogonal lattice $(V^\perp)^\perp$ of $V^\perp$ must be one-dimensional, since $m - (m-1) = 1$. It is possible to retrieve this lattice using the $\mathbf{x}_i$'s, by calculating the determinant of the $m \times m$ matrix

$$X = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_m \\ x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(m-1),1} & x_{(m-1),2} & \cdots & x_{(m-1),m} \end{pmatrix},$$

where $\mathbf{e}_i$ symbolizes the i'th unit vector, and $\mathbf{x}_i = (x_{i1}, \ldots, x_{im})$ for $1 \leq i \leq m-1$. This determinant can also be written as

$$\det(X) = \mathbf{e}_1 \begin{vmatrix} x_{12} & \cdots & x_{1m} \\ x_{22} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{(m-1),2} & \cdots & x_{(m-1),m} \end{vmatrix} + \ldots + \mathbf{e}_m \begin{vmatrix} x_{11} & \cdots & x_{1,(m-1)} \\ x_{21} & \cdots & x_{2,(m-1)} \\ \vdots & \ddots & \vdots \\ x_{(m-1),1} & \cdots & x_{(m-1),(m-1)} \end{vmatrix}. \quad (2.13)$$

The result will be an integral vector V' generating the lattice $(V^{\perp})^{\perp}$, such that $zV' = V$ for some integer z. If there is no denominator common to all V_i (which is true with high probability), then $z = \pm 1$, because there can be no smaller integral vector generating $(V^{\perp})^{\perp}$. Thus, each subdeterminant in (2.13) gives one $\pm V_i$, depending on z.

Now, once the vector $V' = \pm V$ is obtained, it is possible to find an approximation to $\mathbf{u}$ by solving the system of linear equations

$$\begin{pmatrix} - & \mathbf{v}_1 & - \\ & \vdots & \\ - & \mathbf{v}_m & - \end{pmatrix} \mathbf{u}' = V'. \tag{2.14}$$

Since $V' = \pm V$, it is possible that the solution $\mathbf{u}'$ of (2.14) approximates $-\mathbf{u}$ instead of $\mathbf{u}$, but both can be used for decryption. Furthermore, there are only n unknowns and m equations in (2.14), which means that n subdeterminants of (2.13) are sufficient to solve (2.14).

Example 2.2. *The example from Subsection 2.2.3 is now continued to describe the attack on this system. The first step of the attack is to construct the lattice L_β. As β must satisfy the restriction $\beta \geq \frac{2187}{\sqrt{4373}} \approx 33$, β is taken to be equal to $\frac{2187}{\sqrt{4373}}$. This gives rise to the lattice L_β as described previously. Now, this lattice was reduced using the LLL-algorithm, resulting in a reduced lattice, spanned by the columns $\mathbf{y}_i$ of the matrix that can be found in the appendix on page 99.*

$\mathbf{y}$	$\|\mathbf{y}\|$	$\mathbf{y}$	$\|\mathbf{y}\|$	$\mathbf{y}$	$\|\mathbf{y}\|$
$\mathbf{y}_1$	2.83325	$\mathbf{y}_{10}$	3.83746	$\mathbf{y}_{19}$	3.18018
$\mathbf{y}_2$	3.91709	$\mathbf{y}_{11}$	3.54524	$\mathbf{y}_{20}$	3.22969
$\mathbf{y}_3$	3.47864	$\mathbf{y}_{12}$	3.05059	$\mathbf{y}_{21}$	3.98926
$\mathbf{y}_4$	3.69904	$\mathbf{y}_{13}$	3.66608	$\mathbf{y}_{22}$	3.53119
$\mathbf{y}_5$	3.76439	$\mathbf{y}_{14}$	4.24623	$\mathbf{y}_{23}$	3.49574
$\mathbf{y}_6$	3.6791	$\mathbf{y}_{15}$	3.28848	$\mathbf{y}_{24}$	3.73659
$\mathbf{y}_7$	4.21238	$\mathbf{y}_{16}$	3.33429	$\mathbf{y}_{25}$	3.57329
$\mathbf{y}_8$	4.33437	$\mathbf{y}_{17}$	2.94213	$\mathbf{y}_{26}$	4.10063
$\mathbf{y}_9$	4.25502	$\mathbf{y}_{18}$	4.00259	$\mathbf{y}_{27}$	37.6377

Table 2.4: The norms of the vectors obtained from the reduced basis

Those vectors $\mathbf{y}_i$ that have norm $\|\mathbf{y}_i\| \leq \frac{2187}{\sqrt{4373}} \approx 33$ are usable in the attack. As can be seen in Table 2.4, only $\mathbf{y}_{27}$ is unsuitable and the rest is suitable. Now, by the result of Nguyen and Stern (see [78, Theorem 1]), the last 27 entries of each $\mathbf{y}_i$ correspond to vectors $\mathbf{x}_i = (x_1, \ldots, x_{27})$ such that $\sum_j x_j V_j = 0$. Therefore, the vector $V = (V_1, \ldots, V_{27})$ is equal to the orthogonal complement of the $m-1$ vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{26}$. As described, it is possible to use the $\mathbf{x}_i$'s to find $\pm V$ with high probability.

Because the orthogonal complement of the $\mathbf{y}_i$'s is 1-dimensional, it can be explicitly computed by computing the determinant of the following 27×27 matrix:

$$\begin{pmatrix} \mathbf{e}_1 & \cdots & \mathbf{e}_{27} \\ - & \mathbf{x}_1 & - \\ & \vdots & \\ - & \mathbf{x}_{m-1} & - \end{pmatrix}.$$

This results in the sum over the unit vectors $\mathbf{e}_j$ times some subdeterminant determined by the $\mathbf{x}_i$'s, and the result V' will be either $+V$ or $-V$. As mentioned, $n = 3$ such subdeterminants are sufficient to uncover $\mathbf{u}$. By solving the system $A\mathbf{u}' = V^*$, where A is the matrix consisting of the row vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ and $V^* = (V_1', V_2', V_3') = (-9, 8, -6)$. The solution to this system of equations is $\mathbf{u}' = (0.524644, -0.606028, -0.36764)$, which is a reasonable approximation of $-\mathbf{u} = (0.524655, -0.606024, -0.36764)$: $|-\mathbf{u} - \mathbf{u}'| = (1.09, 0.37, 0, 40) \times 10^{-5}$. Both $\mathbf{u}$ and $-\mathbf{u}$ can

be used to decrypt, and using $\mathbf{u}'$ *to decrypt the ciphertext* $\mathbf{c}$ *from* (2.10) *results in the inner products given in Table. This results in the decrypted message* $\mathbf{m}' = (0, 0, 0, 1, 0, 0, 0, 0, 1)$*, as desired. This completes the attack.*

$\mathbf{c}_i$	$\langle \mathbf{c}_i, \mathbf{u}' \rangle$
$\mathbf{c}_1$	-3.995
$\mathbf{c}_2$	-1.000
$\mathbf{c}_3$	11.081
$\mathbf{c}_4$	2.519
$\mathbf{c}_5$	-7.316
$\mathbf{c}_6$	15.760
$\mathbf{c}_7$	4.777
$\mathbf{c}_8$	10.999
$\mathbf{c}_9$	9.458

Table 2.5: Inner products of parts of the ciphertext with $\mathbf{u}'$.

2.2.7 Practicality of the system

Object	Size(bits)
Private key $\mathbf{u}$	n^2
Public key $\{\mathbf{w}_j\}, \{\mathbf{v}_i\}$	$n^5 \log_2(n)$
Ciphertext $\mathbf{c}$	$n^2(\log_2(n) + 1)$

Table 2.6: Sizes of keys and message expansion in AD.

The private key consists of a vector in the n-dimensional unit ball. Since the precision of the binary expansion of each entry is n, the private key can be represented using n^2 bits. The public key consists of $n+n^3$ n-dimensional vectors, with entries of size $\pm r_n/2 = \pm n^n/2$. Thus, it requires at least $n^3 \cdot n \log_2(n^n) = n^5 \log_2(n)$ bits in storage space.

Note that the encryption of a single bit consists of a vector inside the parallelepiped spanned by the $\mathbf{w}_j$'s. Thus, this encryption consists of an n-dimensional vector whose entries can range from $-r_n/2$ to $r_n/2$. Recall that the precision of the binary expansion of each entry is n. Therefore, the ciphertext for one bit is represented by $n(\log_2(n^n) + n) = n^2(\log_2(n) + 1)$ bits.

Experiments by Nguyen and Stern showed that they could successfully attack the system for low values of n, and they claim that a successful attack for $n = 32$ looks feasible. As shown in Table 2.6, for $n = 32$, storing the public key would require $32^5 \log_2(32)$ bits, or 20 Megabytes, and the ciphertext for each single bit would consist of $32^2(\log_2(32) + 1) = 6144$ bits. Thus, the system is not very practical, even for these parameters that provide little security.

2.2.8 Conclusion

The Ajtai-Dwork cryptosystem is theoretically interesting, because of the security proof incorporating hard lattice problems. However, it is not a very efficient system due to the message expansion and high storage space requirements for the public key. As Nguyen and Stern subsequently showed, for parameters where the system is still somewhat practical, it is not secure, due to a clever lattice construction combined with the good performance of lattice reduction algorithms in lower dimensions.

These results give rise to the interesting question whether it is possible to base the security of cryptosystems on the (worst-case) hardness of SVP, rather than uSVP.

2.3 Goldreich-Goldwasser-Halevi

The GGH cryptosystem, named for its introducers Goldreich, Goldwasser and Halevi, was published in 1997 in [30]. It introduced a trapdoor one-way function based on the difficulty of the closest vector problem. Using this trapdoor function, they introduced a public key cryptosystem and digital signature scheme. The system did not come with a proof of security, but it was very efficient in comparison to the Ajtai-Dwork system. Of the three lattice based systems discussed in this chapter, GGH was the only one explicitly defined using lattices in its original paper.

2.3.1 Parameters and setup

GGH relies on two parameters, the lattice dimension n and the security parameter σ. The security parameter determines the difficulty of the closest vector problem that arises from the encryption.

The private key consists of a secret matrix R, whose columns form a basis for the lattice L. This basis consists of reasonably short integral vectors. The creators proposed several methods to construct R. The public key consists of a public matrix B, representing a different basis for L. The public basis is a bad basis, in the sense that it is not as reduced as the secret basis. There are several methods to randomly generate the public basis B from the secret basis R. The authors themselves proposed two methods and later Micciancio proposed to use the more efficient Hermite Normal Form (HNF) of R as the public basis [65, 66].

Parameter	Description	Knowledge
n	Dimension	Public
σ	Security parameter	Public
R	$n \times n$ integral matrix	Private
B	$n \times n$ integral matrix	Public

Table 2.7: Parameters of GGH.

2.3.2 Encryption and decryption

Let the parameters of the system be defined as in Table 2.7. Then, encryption and decryption are performed as follows.

Encryption

To encrypt a message, encode it as an integral vector $\mathbf{m} \in \mathbb{Z}^n$, generate a random error vector $\mathbf{e}$ and compute the ciphertext

$$\mathbf{c} = B\mathbf{m} + \mathbf{e}. \tag{2.15}$$

In the original proposal, the error vector $\mathbf{e}$ is taken randomly from $\{-\sigma, \sigma\}^n$.

Decryption

To decrypt a ciphertext $\mathbf{c}$, compute

$$\mathbf{m} = B^{-1}R\lfloor R^{-1}\mathbf{c} \rceil \tag{2.16}$$

to retrieve the message $\mathbf{m}$.

2.3.3 Why it works

In the GGH system, each message $\mathbf{m} \in \mathbb{Z}^n$ corresponds to a lattice point $\mathbf{m}_L = B\mathbf{m}$. Conversely, each lattice point $\mathbf{m}_L$ corresponds to some message $\mathbf{m} = B^{-1}\mathbf{m}_L$. During the encryption, $\mathbf{m}$ is first transformed into its corresponding lattice point $\mathbf{m}_L$, and then the ciphertext $\mathbf{c}$ is obtained by adding an error vector $\mathbf{e}$. Now, σ and $\mathbf{e}$ are chosen such that $\mathbf{m}_L$ will be the lattice vector that is closest to the ciphertext $\mathbf{c}$.

To retrieve $\mathbf{m}_L$ (and therefore $\mathbf{m}$), this closest vector problem must be solved. Therefore, the decryption process consists of Babai's rounding method for approximating CVP, as explained in Chapter 1. The idea is that Babai's method will work sufficiently well when using the private basis R, but not well enough when using the public basis B. Decryption using the private basis works when Babai's method finds the correct lattice vector $R\lfloor R^{-1}\mathbf{c}\rceil = \mathbf{m}_L$. Because $R^{-1}\mathbf{m}_L$ is an integral vector, Babai's method will find

$$\begin{aligned} R\lfloor R^{-1}\mathbf{c}\rceil &= R\lfloor R^{-1}(\mathbf{m}_L + \mathbf{e})\rceil \\ &= R(R^{-1}\mathbf{m}_L + \lfloor R^{-1}\mathbf{e}\rceil) \\ &= \mathbf{m}_L + R\lfloor R^{-1}\mathbf{e}\rceil. \end{aligned} \tag{2.17}$$

If $\lfloor R^{-1}\mathbf{e}\rceil = \mathbf{b}$ is a non-zero vector, then $R\mathbf{b}$ will be a non-zero lattice vector. In this case, Babai's method will not return the desired lattice point, and hence the wrong message is retrieved. Thus, the decryption is correct whenever $\lfloor R^{-1}\mathbf{e}\rceil = \mathbf{0}$. This will happen when σ is small, because the error vector $\mathbf{e}$ is of the form $(\pm\sigma, \pm\sigma, \ldots, \pm\sigma)$. Increasing σ will increase the distance between the lattice vector $\mathbf{m}_L$ and the ciphertext $\mathbf{c}$. As this distance increases, the CVP will become harder. The probability that a decryption error occurs increases as well, as it will become more likely that $\lfloor R^{-1}\mathbf{e}\rceil \neq \mathbf{0}$. For a more detailed analysis of the probability of decryption errors, see [30].

Trying to use the public basis B for Babai's method will fail, since B is constructed to be a bad basis. Trying to reduce the basis B to some basis B' that can be used to solve the CVP is equivalent to solving the approximate shortest basis problem (see Chapter 1).

Example 2.3. *This example shows an instance of the GGH cryptosystem for dimension $n = 4$ and security parameter $\sigma = 1$. The private matrix R is generated by randomly taking its entries as integers in the interval $[-4, 4]$, which is one of the proposed methods in [30]. The public matrix B is the HNF of the matrix R (as implemented in Mathematica[1]). This results in the following matrices:*

$$R = \begin{pmatrix} 3 & 1 & -1 & 3 \\ 3 & -4 & 3 & 1 \\ 3 & -3 & -4 & -3 \\ -2 & -3 & -2 & 3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -349 & 311 & 321 & 851 \end{pmatrix}.$$

Next, a message $\mathbf{m} = (-2, 0, -4, -1)$ is encrypted using the error vector $\mathbf{e} = (-1, 1, 1, -1)$. The corresponding ciphertext is given by

$$\mathbf{c} = B\mathbf{m} + \mathbf{e} = (-3, 1, -3, -1438).$$

To decrypt this message using the private basis R, Babai's method is applied:

$$\begin{aligned} \mathbf{x} &= \lfloor R^{-1}\mathbf{c}\rceil = (143, 152, 112, -157), \\ \mathbf{m}_L &= R\mathbf{x} = (-2, 0, -4, -1437), \\ \mathbf{m} &= B^{-1}\mathbf{m}_L = (-2, 0, -4, -1). \end{aligned}$$

Indeed, m is correctly decrypted, as

$$\begin{aligned} R^{-1}\mathbf{e} &= (72/851, -(124/851), 136/851, -(269/851)), \\ \lfloor R^{-1}\mathbf{e}\rceil &= (0, 0, 0, 0). \end{aligned}$$

[1] Mathematica's implementation of the HNF is different from the standard implementation. This results in a matrix that is not quite suitable for encryption, although it does not harm the example.

Trying to decrypt using the public basis B results in

$$\lfloor B^{-1}\mathbf{c} \rceil = (-3, 1, -3, -2),$$

which is not equal to $\mathbf{m}$.

2.3.4 Attack on the encryption

The authors posted several instances of their system on the Internet to challenge people to break it. These were instances in the dimensions $n = 200$, 250, 300, 350 and 400, each instance using the security parameter $\sigma = 3$. For these challenges, plaintexts were chosen uniformly from $\{-128, \ldots, 127\}^n$. The challenge in dimension 200 was defeated using high-quality lattice reduction algorithms combined with the embedding technique that heuristically transforms a CVP in dimension n to an SVP in dimension $n+1$ (see Chapter 1). In 1999, Nguyen published an attack on the GGH system [74] and was able to beat the challenges for dimensions up to 350. Although the challenge for dimension 400 was not defeated, the running time and storage required for these dimensions was no longer practical.

Nguyen's attack is based on two flaws in the GGH system. The first flaw is that the error vectors are always very short when compared to the lattice vectors. As described in Chapter 1, this will create a gap between the successive minima $\lambda_1(L)$ and $\lambda_2(L)$ in the lattice L that is constructed using the embedding technique of Chapter 1, and hence this CVP-instance becomes easier to solve than a general CVP-instance.

The second flaw is based on the choice of the error vector. Because the absolute values of the entries in $\mathbf{e}$ are all σ, the following equality holds:

$$\begin{aligned} \mathbf{c} + (\sigma, \ldots, \sigma) &= B\mathbf{m} + \mathbf{e} + \sigma\mathbf{1} \\ &\equiv B\mathbf{m} \pmod{2\sigma}. \end{aligned} \tag{2.18}$$

Nguyen subsequently showed that there is a reasonable probability that B is invertible $\bmod 2\sigma$ for several values of σ. He also showed that even if B is not invertible $\bmod 2\sigma$, the kernel will likely be small, which means there will be a small number of possibilities of $\mathbf{m}_{2\sigma} = \mathbf{m} \bmod 2\sigma$.

Once $\mathbf{m}_{2\sigma}$ is known (or several possibilities), the decryption problem can be simplified:

$$\begin{aligned} \mathbf{c} &= B\mathbf{m} - B\mathbf{m}_{2\sigma} + B\mathbf{m}_{2\sigma} + \mathbf{e}, \\ \mathbf{c} - B\mathbf{m}_{2\sigma} &= B(\mathbf{m} - \mathbf{m}_{2\sigma}) + \mathbf{e}. \end{aligned}$$

The vector $\mathbf{m} - \mathbf{m}_{2\sigma}$ is the zero vector (mod 2σ), and hence of the form $2\sigma\mathbf{m}'$ for some vector $\mathbf{m}' \in \mathbb{Z}^n$. Therefore, the following equation holds:

$$\frac{\mathbf{c} - B\mathbf{m}_{2\sigma}}{2\sigma} = B\mathbf{m}' + \frac{\mathbf{e}}{2\sigma}. \tag{2.19}$$

As the left-hand side of (2.19) is known, this becomes an instance of CVP with an even smaller error vector of the form $\left\{-\frac{1}{2}, \frac{1}{2}\right\}^n$. The goal of this CVP instance is to find the closest lattice vector $B\mathbf{m}'$ to the target vector $\mathbf{c}'$ defined by:

$$\begin{aligned} \mathbf{c}' &= \frac{\mathbf{c} - B\mathbf{m}_{2\sigma}}{2\sigma}, \\ \mathbf{m}' &= \mathbf{m} - \mathbf{m}_{2\sigma}, \\ \mathbf{e}' &= \mathbf{c}' - \mathbf{m}' = \frac{\mathbf{e}}{2\sigma}. \end{aligned}$$

Solving this CVP-instance yields the error vector $\mathbf{e}'$, which in turn yields the solution for the original error vector $\mathbf{e}$. Applying the embedding technique of Chapter 1 will now be easier, because it is more likely to work with smaller error vectors. Once $\mathbf{e}$ is acquired, it is easy to retrieve the message $\mathbf{m} = B^{-1}(\mathbf{c} - \mathbf{e})$.

Nguyen proposed some solutions to repair the second flaw in the system, but noted that these would also increase the vulnerability to the first flaw. For example, rather than taking the error vector from $\{\pm\sigma\}^n$, he proposed to take the error vector uniformly from $\{-\sigma, -(\sigma-1), \ldots, (\sigma-1), \sigma\}^n$. He notes that, unfortunately, this would decrease the expected length of the error vector from $\sigma\sqrt{n}$ to $\sigma\sqrt{n/3}$. Increasing the security parameter σ in this case will increase security, but the probability that decryption errors occur will increase as well. Therefore, σ needs to be kept small, which leaves the first flaw unrepaired.

Example 2.4. *The example from Section 2.3.3 is continued here to show Nguyen's attack on the encryption. First, the vector* $\mathbf{m}_{2\sigma} = \mathbf{m} \bmod 2\sigma$ *is computed using* (2.18)*. Fortunately,* B *is invertible modulo 2, where the inverse is given by*

$$B^{-1} \pmod{2\sigma} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Now $\mathbf{m}_{2\sigma}$ *can be found by solving* (2.18)*:*

$$\begin{aligned} B\mathbf{m} &\equiv \mathbf{c} + \sigma\mathbf{1} \pmod{2\sigma} \\ &\equiv (-2, 2, -2, -1437) \pmod 2 \\ &\equiv (0, 0, 0, 1) \pmod 2, \\ \mathbf{m}_{2\sigma} &= B^{-1} \cdot (0, 0, 0, 1) = (0, 0, 0, 1). \end{aligned}$$

Next, (2.19) *is used to construct the easier CVP-instance*

$$\mathbf{c}' = B\mathbf{m}' + \mathbf{e}',$$

where

$$\begin{aligned} \mathbf{c}' &= \frac{\mathbf{c} - B\mathbf{m}_{2\sigma}}{2\sigma} = (-3/2, 1/2, -3/2, -2289/2), \\ \mathbf{m}' &= \frac{\mathbf{m} - \mathbf{m}_{2\sigma}}{2\sigma}, \\ \mathbf{e}' &= \frac{\mathbf{e}}{2\sigma}. \end{aligned}$$

Now, solving this CVP will give $\mathbf{e}'$*, which in turn gives* $\mathbf{e}$*. To solve this CVP, the embedding technique from Chapter 1 was used. The embedded lattice is spanned by the columns of the following matrix:*

$$\begin{pmatrix} 1 & 0 & 0 & 0 & -(3/2) \\ 0 & 1 & 0 & 0 & 1/2 \\ 0 & 0 & 1 & 0 & -(3/2) \\ -349 & 311 & 321 & 851 & -(2289/2) \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.20}$$

Recall from the description of the embedding technique in Chapter 1 that for any vector $\mathbf{x}$ *in the original lattice,* $(\mathbf{c}', 1) - (\mathbf{x}, 0)$ *is a vector in the embedded lattice. Thus, the embedded lattice will contain the short vector* $(\mathbf{c}' - B\mathbf{m}', 1)$*, where* $B\mathbf{m}'$ *is the closest vector to* $\mathbf{c}'$ *in the original lattice. The matrix in 2.20 is now reduced using LLL, resulting in the following matrix of the reduced basis (the last row has been removed for simplicity):*

$$\begin{pmatrix} -1/2 & -2 & 1 & -7/2 & -1/2 \\ 1/2 & -2 & -3 & -5/2 & 7/2 \\ 1/2 & 2 & 4 & -5/2 & 5/2 \\ -1/2 & -2 & 2 & 3/2 & 7/2 \end{pmatrix}.$$

Now, the first column of this matrix is equal to $\mathbf{e}' = \mathbf{e}/2$*, thus the attacker can now retrieve* $\mathbf{e} = (-1, 1, 1, -1)$*. Using* $\mathbf{e}$*, it is easy to retrieve the message* $\mathbf{m}$ *from the ciphertext* $\mathbf{c}$*, and thus the attack is complete.*

2.3.5 Practicality of the system

Object	Size(bits)
Private key R	$n^2 \log_2(k)$
Public key B	$n^2 \log_2(n)$
Operation	Time
Encryption	$n^2 \log_2(n)$

Table 2.8: Sizes of keys and encryption time in GGH.

Table 2.8 contains the asymptotic key sizes and encryption time. The private key R consists of a matrix of n^2 integers, which means it requires $n^2 \log_2(k)$ bits in storage space, where the elements of R are in the interval $[i, i+k]$ for some integer i. Micciancio showed in [66] that, if no care is taken in selecting the method to transform R to B, the public key B can become an $n \times n$ matrix with integral coefficients of that require $O(n \log_2(n))$ bits to be represented. Thus, this will result in a public key B requiring $O(n^3 \log_2(n))$ bits in storage. When using the HNF of R as the public basis, this can be reduced to $O(n^2 \log_2(n))$ bits. The encryption procedure consists of the multiplication of an $n \times n$ matrix and an n-dimensional vector, as well as the addition of one n-dimensional vector. As such, the encryption time is $O(n^2 \log(n))$ for dimension n.

The attack by Nguyen broke all systems except for that of dimension $n = 400$. For the $n = 400$ challenge, the public basis chosen by the authors of GGH required more than 2 Megabyte of storage space. As Micciancio observed in [66], the size of the public key of this challenge is smaller than it should have been. This is probably because the public bases were reduced using LLL to somehow reduced their size. Micciancio also estimates that the use of his HNF method would reduce the public key size to about 140 Kilobyte for dimension $n = 400$.

The attack would not perform well asymptotically in n, as the embedding technique is based on lattice reduction algorithms to solve a SVP instance. Nonetheless, it defeats the GGH system in the practical dimensions, and especially in the dimensions where GGH is competitive with other public key cryptosystems, such as RSA and ElGamal (see Chapter 1).

2.3.6 Conclusion

The GGH cryptosystem is interesting because it was explicitly based on lattices. However, it turned out that the trapdoor function that was used for the closest vector problem introduced structural weaknesses that could be exploited. Once again the attack uses lattice reduction algorithms that performed sufficiently to break the system in the low dimensions. Although more effective than AD, raising the parameters is not an option for GGH either, if the system needs to compete with current cryptosystems such as RSA and ElGamal (see Chapter 1).

2.4 NTRU

The first version of the NTRU cryptosystem was presented in 1996 by Hoffstein, Pipher and Silverman [41]. The name NTRU allegedly stands for "N-th degree *tru*ncated polynomial ring", but it has alternatively been explained as "Number Theorists "R" Us," a reference to the toy store chain "Toys "R" Us".

Although the NTRU system is regarded as lattice-based cryptography, it is typically described using the ring of convolution polynomials. Since these convolution products of polynomials can also be expressed as the multiplication with a circulant matrix, it is possible to describe NTRU using lattices. This fact was used to attack the system using lattice reduction methods. Like GGH, NTRU is aimed at practicality, and not supported by a proof of security.

NTRU consists of the cryptosystem NTRUEncrypt and the signature scheme NTRUSign. The aim of this section is to introduce the NTRUEncrypt cryptosystem using lattices and to show

an attack using lattice reduction. First a short introduction to truncated polynomial rings and convolution products will be given.

2.4.1 Truncated polynomial rings

Consider the polynomial ring $\mathbb{Z}[X]$ of polynomials in the variable X having integer coefficients. Truncated polynomial rings arise when taking the quotient ring $\mathbb{Z}[X]/\langle X^n - 1\rangle$ for some n. This results in a ring of polynomials a of the form

$$a(X) = a_0 + a_1X + a_2X^2 + \ldots + a_{n-1}X^{n-1},$$

where the a_i's are all integers for $0 \leq i \leq n-1$. Now, this ring has the usual addition for polynomials, and the sum of two truncated polynomials a and b is given by

$$(a+b)(X) = (a_0 + b_0) + (a_1 + b_1)X + (a_2 + b_2)X^2 + \ldots + (a_{n-1} + b_{n-1})X^{n-1}.$$

The multiplication in this ring is the same as the usual multiplication, with one exception. When multiplying two normal polynomials a' and b' of degree $n-1$, the result is

$$\begin{aligned}(a' \cdot b')(X) &= a_0b_0 + (a_0b_1 + a_1b_0)X + \ldots + (\sum_{i+j=k} a_ib_j)X^k + \ldots + (\sum_{i+j=2(n-1)} a_ib_j)X^{2(n-1)} \\ &= c'_0 + c'_1X + c'_2X^2 + \ldots + c'_{2(n-1)}X^{2(n-1)},\end{aligned}$$

where $c'_k = \sum_{i+j=k} a_ib_j$ for $0 \leq k \leq 2(n-1)$. However, truncated polynomials are in the quotient ring $\mathbb{Z}[X]/(X^n - 1)$, which has the equivalence classes

$$\begin{aligned}X^n &= 1 \pmod{X^n - 1}, \\ X^{n+1} &= X \pmod{X^n - 1}, \\ X^{n+2} &= X^2 \pmod{X^n - 1}, \\ &\vdots \\ X^{n+k} &= X^{k \bmod n} \pmod{X^n - 1},\end{aligned}$$

for any integer $k \geq 1$. Thus, when multiplying the truncated polynomials a and b, the result will be

$$(a * b)(X) \equiv c_0 + c_1X + c_2X^2 + \ldots + c_{n-1}X^{n-1} \pmod{X^n - 1},$$

where

$$c_k = c'_k + c'_{k+n} = \sum_{i+j \equiv k \bmod n} a_ib_j.$$

An equivalent approach is to consider c_k as the inner product of the coefficients of a and b, where b is cyclically rotated to the left over k positions. The multiplication is sometimes called a convolution product, and it satisfies the usual properties such as associativity, commutativity and distributivity, which confirms that the truncated polynomials form a ring. For the NTRU system, the coefficients will be integers modulo q for some q.

2.4.2 Parameters and setup

As mentioned, NTRU is typically described as a polynomial ring cryptosystem. However, the relation between the public and private key defines a certain lattice, which is called the NTRU lattice. A basis for this lattice can be derived from the public key. Furthermore, the secret key of the cryptosystem corresponds to certain short vectors in this lattice. Thus, a natural attack is to try and solve the approximate shortest vector problem in the NTRU lattice. Still, it is not necessary to use lattices when defining the encryption and decryption procedures.

The system relies on several parameters that provide a balance between security and practicality. These are

- the degree n,
- the large modulus q,
- the small modulus p, and
- the integer bounds d_f, d_g and d_r.

For security reasons, n must be prime, or the polynomial $X^n - 1$ can be factorized, which improves the efficiency of lattice attacks. Furthermore, p and q must be relatively prime. The parameters n and q determine the parameters of the associated NTRU lattice, whereas the integer bounds restrict the form of the lattice vectors, as will become apparent from the description of the setup.

Before describing the NTRU lattice, some extra notation is needed. Let C denote the cyclic rotation that sends a vector $(x_1, x_2, \ldots, x_n)^T$ to $(x_n, x_1, \ldots, x_{n-1})^T$. Additionally, define for some $\mathbf{x} \in \mathbb{R}^n$ the circulant matrix of $\mathbf{x}$ as

$$[C^*\mathbf{x}] = [\mathbf{x}, C\mathbf{x}, \ldots, C^{n-1}\mathbf{x}] = \begin{pmatrix} x_1 & x_n & \cdots & x_2 \\ x_2 & x_1 & \cdots & x_3 \\ \vdots & \vdots & \ddots & \vdots \\ x_n & x_{n-1} & \cdots & x_1 \end{pmatrix}.$$

Note that the convolution product (modulo $X^n - 1$) of the polynomials $f * g$ is equivalent to the matrix multiplication $[C^*\mathbf{f}]\mathbf{g}$, where $\mathbf{f}$ and $\mathbf{g}$ are the coefficient vectors of f and g, respectively. Furthermore, the following identities can be derived from the properties of the convolution product:

$$[C^*\mathbf{f}](\mathbf{g} + \mathbf{g}') = [C^*\mathbf{f}]\mathbf{g} + [C^*\mathbf{f}]\mathbf{g}', \tag{2.21}$$

$$[C^*\mathbf{f}][C^*\mathbf{g}] = [C^*([C^*\mathbf{f}]\mathbf{g})] \tag{2.22}$$

$$[C^*(C\mathbf{f})]\mathbf{g} = [C^*\mathbf{f}](C\mathbf{g}). \tag{2.23}$$

The setup of NTRUEncrypt starts with the choice of the private key $(\mathbf{f}, \mathbf{g})$, which is a short vector in $\mathbb{Z}^{2n}$. The vectors $\mathbf{f}$ and $\mathbf{g}$ are secret, and must satisfy the following properties:

- The matrix $[C^*\mathbf{f}]$ must be invertible mod q and mod p. Denote the inverses by $[C^*\mathbf{f}]_q^{-1}$ and $[C^*\mathbf{f}]_p^{-1}$, respectively.
- $\mathbf{f}$ and $\mathbf{g}$ must be small, i.e., $\mathbf{f}$ has d_f entries equal to 1 and $d_f - 1$ entries equal to -1, the rest being 0. Likewise, $\mathbf{g}$ has d_g entries equal to 1 and d_g entries equal to -1, the rest being 0.

The public key $\mathbf{h}$ can now be derived from the private key by the equation

$$[C^*\mathbf{f}]\mathbf{h} \equiv p\mathbf{g} \pmod{q}. \tag{2.24}$$

Since $[C^*\mathbf{f}]$ is invertible mod q, this becomes $\mathbf{h} = p[C^*\mathbf{f}]^{-1}\mathbf{g} \bmod q$.

Let $H = p_q^{-1}[C^*\mathbf{h}]$, where p_q^{-1} is the inverse of p mod q. Now, the NTRU lattice is defined by all integral vectors $(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^{2n}$ that satisfy $\mathbf{y} \equiv H\mathbf{x} \pmod{q}$, Note that for every vector $(\mathbf{x}, \mathbf{y})$ in the lattice, the pairwise rotation $(C\mathbf{x}, C\mathbf{y})$ will be in the lattice as well, because $C\mathbf{y} \equiv C(H\mathbf{x}) \equiv H(C\mathbf{x}) \pmod{q}$, by (2.23). Specifically, $(\mathbf{f}, \mathbf{g})$ and all its pairwise rotations $(C^k\mathbf{f}, C^k\mathbf{g})$ are vectors in this lattice for $0 \leq k \leq n - 1$. The lattice is spanned by the columns of the matrix

$$L = \begin{pmatrix} I_n & O_n \\ H & qI_n \end{pmatrix}, \tag{2.25}$$

where I_n is the $n \times n$ identity matrix and O_n is the $n \times n$ all-zero matrix. Although the NTRU lattice has dimension $2n$, its basis can be represented using only the n-dimensional vector $\mathbf{h}$.

Parameter	Description	Knowledge
n	Degree, prime	Public
q	Large modulus	Public
p	Small modulus	Public
d_f	Integer bound for f	Public
d_g	Integer bound for g	Public
d_r	Integer bound for r	Public
$\mathbf{f}$	Coefficient vector of f	Private
$\mathbf{g}$	Coefficient vector of g	Private
$\mathbf{h} = p[C^*\mathbf{f}]_q^{-1}\mathbf{g} \pmod q$	Coefficient vector of h	Public
$H = p_q^{-1}[C^*\mathbf{h}] \pmod q$	Circulant $n \times n$ matrix	Public

Table 2.9: Parameters of NTRU.

2.4.3 Encryption and decryption

Let the parameters of the system be defined as in Table 2.9. Then, encryption and decryption are performed as follows.

Encryption

To encrypt a message, encode it as a vector $\mathbf{m} \in \mathbb{Z}^n$ with coefficients modulo p. Then, randomly generate a blinding factor $\mathbf{r} \in \{-1, 0, 1\}^n$ (where $\mathbf{r}$ has d_r entries 1, d_r entries -1 and the rest 0). Finally, compute the ciphertext

$$\mathbf{c} = [C^*\mathbf{h}]\mathbf{r} + \mathbf{m} \pmod q.$$

Decryption

To decrypt a ciphertext $\mathbf{c}$, reduce

$$\mathbf{a} \equiv [C^*\mathbf{f}]\mathbf{c} \pmod q,$$

such that all coefficients of $\mathbf{a}$ lie in the interval $[-q/2, q/2)$. Then, retrieve the message by computing

$$\mathbf{m} = [C^*\mathbf{f}]_p^{-1}\mathbf{a} \pmod p.$$

2.4.4 Why it works

To decrypt a ciphertext $\mathbf{c}$, it is multiplied with the matrix $[C^*\mathbf{f}]$ to get

$$\begin{aligned}[C^*\mathbf{f}]\mathbf{c} &= [C^*\mathbf{f}]\mathbf{m} + [C^*\mathbf{f}][C^*\mathbf{h}]\mathbf{r} \\ &= [C^*\mathbf{f}]\mathbf{m} + [C^*([C^*\mathbf{f}]\mathbf{h})]\mathbf{r} \\ &\equiv [C^*\mathbf{f}]\mathbf{m} + p[C^*\mathbf{g}]\mathbf{r} \pmod q, \end{aligned} \tag{2.26}$$

where the identities (2.21) and (2.22) were used.

Now, $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{r}$ are by definition "small" polynomials with respect to q, depending on the integer bounds d_f, d_g and d_r. As a result, the entries of the resulting vector $[C^*\mathbf{f}]\mathbf{m} + p[C^*\mathbf{g}]\mathbf{r}$ in (2.26) are likely to be in the interval $[-q/2, q/2)$, without having to reduce mod q. Thus, reducing the coefficients of $[C^*\mathbf{f}]\mathbf{c} \bmod q$ such that they lie in the interval $[-q/2, q/2)$ will yield the exact vector $[C^*\mathbf{f}]\mathbf{m} + p[C^*\mathbf{g}]\mathbf{r}$ over $\mathbb{Z}$, rather than merely an equivalent vector mod q. The message $\mathbf{m}$ is subsequently retrieved by multiplying by $[C^*\mathbf{f}]_p^{-1}$ and reducing modulo p, resulting in

$$[C^*\mathbf{f}]_p^{-1}([C^*\mathbf{f}]\mathbf{m} + p[C^*\mathbf{g}]\mathbf{r}) \equiv [C^*\mathbf{f}]_p^{-1}[C^*\mathbf{f}]\mathbf{m} + 0 \equiv \mathbf{m} \pmod p.$$

The integer bound parameters d_f, d_g and d_r are chosen such that even the convolution products are likely to have small coefficients. The bigger d_f, d_g and d_r are, the bigger the probability that the entries of $\mathbf{x} = [C^*\mathbf{f}]\mathbf{m} + p[C^*\mathbf{g}]\mathbf{r}$ are not in the correct interval $[-q/2, q/2)$. If this happens, it might for example occur that, during the decryption, an entry $x_i + q$ is reduced modulo p rather than x_i, and these do not have the same outcome $\bmod p$ since p and q are co-prime. This will lead to a decryption error, and since these decryption errors are highly key-dependent, they can be used to extract the private key in some situations, as shown in [45, 84]. This prompted the creators of NTRU to change the recommended parameters of the system. For more information on these new parameters as well as a more detailed analysis on the probability bounds of decryption errors, see [39, 36].

Example 2.5. *The following example shows the system in action for the small parameters $n = 7$, $q = 32$, $p = 3$ $d_f = 3$, and $d_g = d_r = 2$. The randomly generated private key consists of $\mathbf{f} = (-1, 0, 0, 1, 1, -1, 1)$ and $\mathbf{g} = (1, -1, 0, 0, -1, 1, 0)$. By (2.24), the public key becomes $\mathbf{h} = (0, 24, 26, 27, 10, 12, 29)$. Furthermore, the inverses of $[C^*\mathbf{f}]$ modulo q and p become*

$$[C^*\mathbf{f}]_q^{-1} = \begin{pmatrix} 17 & 28 & 16 & 1 & 30 & 9 & 28 \\ 28 & 17 & 28 & 16 & 1 & 30 & 9 \\ 9 & 28 & 17 & 28 & 16 & 1 & 30 \\ 30 & 9 & 28 & 17 & 28 & 16 & 1 \\ 1 & 30 & 9 & 28 & 17 & 28 & 16 \\ 16 & 1 & 30 & 9 & 28 & 17 & 28 \\ 28 & 16 & 1 & 30 & 9 & 28 & 17 \end{pmatrix}, \quad [C^*\mathbf{f}]_p^{-1} = \begin{pmatrix} 2 & 1 & 2 & 2 & 0 & 1 & 2 \\ 2 & 2 & 1 & 2 & 2 & 0 & 1 \\ 1 & 2 & 2 & 1 & 2 & 2 & 0 \\ 0 & 1 & 2 & 2 & 1 & 2 & 2 \\ 2 & 0 & 1 & 2 & 2 & 1 & 2 \\ 2 & 2 & 0 & 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 0 & 1 & 2 & 2 \end{pmatrix}.$$

Now, a message $\mathbf{m} = (0, -1, 1, -1, 1, 0, -1)$ is encrypted using the randomly chosen blinding factor $\mathbf{r} = (1, 1, 0, 0, 0, -1, -1)$. This results in the ciphertext

$$\mathbf{c} = [C^*\mathbf{h}]\mathbf{r} + \mathbf{m} \equiv (11, 2, 14, 30, 29, 25, 16) \pmod q.$$

To decrypt this ciphertext, it is multiplied by $[C^*\mathbf{f}]$ and reduced $\bmod q$ such that its coefficients are in the interval $[-q/2, q/2)$:

$$[C^*\mathbf{f}]\mathbf{c} \equiv (4, 4, -4, 1, -7, -4, 5) \pmod q. \tag{2.27}$$

Note that the right-hand side of (2.27) is precisely equal to

$$[C^*\mathbf{f}]\mathbf{m} + p[C^*\mathbf{g}]\mathbf{r} = (4, 4, -4, 1, -7, -4, 5).$$

Thus, multiplying by $[C^*\mathbf{f}]_p^{-1}$ and reducing that modulo p gives

$$[C^*\mathbf{f}]_p^{-1} \cdot (4, 4, -4, 1, -7, -4, 5) \equiv (0, -1, 1, -1, 1, 0, -1) \pmod p = \mathbf{m}.$$

2.4.5 Attack on the private key

This attack on the private key was introduced by Coppersmith and Shamir in [11]. The goal is to retrieve the secret vectors $\mathbf{f}$ and $\mathbf{g}$, or at least vectors that are not too far removed from them, and can still be used to decrypt ciphertexts.

Now, the vector $(\mathbf{f}, \mathbf{g})$ and its pairwise rotations are vectors in the NTRU lattice. Using the currently chosen parameters, the Euclidean norm of $(\mathbf{f}, \mathbf{g})$ is given by $\|(\mathbf{f}, \mathbf{g})\| = \sqrt{2d_f - 1 + 2d_g}$. The lattice volume of the NTRU lattice is given by $\det(L) = q^n$ and the dimension of the lattice is $2n$. Now, the shortest vector is expected to have a length that is approximately $\det(L)^{\frac{1}{2n}} = \sqrt{q}$, as seen in Chapter 1.

Since $\sqrt{2d_f - 1 + 2d_g} \ll \sqrt{q}$ when using the recommended parameters for NTRUEncrypt, it becomes highly probable that $(\mathbf{f}, \mathbf{g})$ is a shortest vector of the lattice. Since all the pairwise rotations of $(\mathbf{f}, \mathbf{g})$ are in the lattice, they are candidates for the shortest vector as well. Furthermore,

they can be used to decrypt messages, since by (2.21), (2.22) and (2.23)

$$\begin{aligned}[C^*(C\mathbf{f})]\mathbf{c} &= [C^*(C\mathbf{f})]\mathbf{m} + [C^*(C\mathbf{f})][C^*\mathbf{h}]\mathbf{r} \\ &= [C^*(C\mathbf{f})]\mathbf{m} + [C^*([C^*\mathbf{f}]\mathbf{h})](C\mathbf{r}) \\ &\equiv [C^*(C\mathbf{f})]\mathbf{m} + p[C^*\mathbf{g}]\mathbf{r} \pmod{q}.\end{aligned}$$

The decryption can now be finished by using the inverse of $[C^*(C\mathbf{f})] \bmod p$ and reducing $\bmod p$.

To retrieve a short vector in the lattice, any lattice reduction algorithm can be applied. Coppersmith and Shamir showed that even when the resulting short vector is longer than $(\mathbf{f}, \mathbf{g})$, but still within some constant factor, several of these vectors can be combined to decrypt the ciphertexts. However, using the current lattice reduction algorithms, the approximation factor of the shortest vector problem is still exponential in n. As shown by Howgrave-Graham in [44] this method can be improved by combining it with a combinatorial attack. These attacks have increased the recommended parameters, but NTRUEncrypt remains practical for now.

Example 2.6. *The example from 2.4.4 is continued here, to describe the attack. For the public key* $\mathbf{h} = (0, 24, 26, 27, 10, 12, 29)$*, the NTRU lattice is spanned by the columns of the following matrix:*

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 319 & 132 & 110 & 297 & 286 & 264 & 32 & 0 & 0 & 0 & 0 & 0 & 0 \\
264 & 0 & 319 & 132 & 110 & 297 & 286 & 0 & 32 & 0 & 0 & 0 & 0 & 0 \\
286 & 264 & 0 & 319 & 132 & 110 & 297 & 0 & 0 & 32 & 0 & 0 & 0 & 0 \\
297 & 286 & 264 & 0 & 319 & 132 & 110 & 0 & 0 & 0 & 32 & 0 & 0 & 0 \\
110 & 297 & 286 & 264 & 0 & 319 & 132 & 0 & 0 & 0 & 0 & 32 & 0 & 0 \\
132 & 110 & 297 & 286 & 264 & 0 & 319 & 0 & 0 & 0 & 0 & 0 & 32 & 0 \\
319 & 132 & 110 & 297 & 286 & 264 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 32
\end{pmatrix}.$$

Reducing this matrix using LLL gives:

$$\begin{pmatrix}
1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 0 & -2 & -1 & 4 & 7 & 6 \\
-1 & 0 & 1 & 0 & -2 & 1 & 1 & -2 & 5 & -3 & -4 & 0 & -4 & 1 \\
1 & 0 & -1 & 1 & -1 & -1 & 1 & -6 & -6 & -7 & -2 & -5 & -1 & -2 \\
-1 & 1 & 0 & 2 & 0 & 1 & 1 & 4 & -1 & 6 & 5 & 6 & 3 & -6 \\
0 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 2 & 4 & 0 & 1 & -6 & 4 \\
0 & -1 & 1 & 0 & -1 & 0 & 1 & -5 & 6 & 3 & 0 & -2 & -1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 6 & -4 & 0 & 4 & -6 & 2 & -5 \\
-1 & 1 & 1 & 0 & 0 & 0 & 0 & -4 & -5 & -7 & 2 & -3 & 6 & -1 \\
1 & -1 & 0 & -1 & 0 & -1 & 0 & 5 & -4 & 2 & 6 & 5 & 4 & -4 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & -6 & 1 & 0 & 1 & 4 & -5 & 5 \\
1 & 0 & -1 & -1 & 1 & 0 & 0 & -3 & 4 & 4 & 7 & -1 & -3 & -6 \\
-1 & -1 & 0 & 0 & -1 & 1 & 0 & 5 & -5 & 4 & 10 & -4 & 1 & -3 \\
0 & 1 & 0 & 1 & -1 & -1 & 0 & 4 & 6 & 3 & 4 & 5 & 3 & 5 \\
0 & 0 & -1 & 1 & 1 & 0 & 0 & -1 & 3 & -6 & 2 & -6 & -6 & 4
\end{pmatrix}.$$

Now, columns 1, 2, 3 and 6 of this matrix are of the proper form: the entries are ± 1 *or* 0*, the first 7 entries of these columns contain* d_f *ones and* $d_f - 1$ *minus ones and the last 7 entries of these columns contain* d_g *ones and* d_g *minus ones. Thus, the vectors* $(1, -1, 1, -1, 0, 0, 1)$*,* $(-1, 0, 0, 1, 1, -1, 1)$*,* $(-1, 1, -1, 0, 0, 1, 1)$ *and* $(1, 1, -1, 1, -1, 0, 0)$ *are obtained. The second vector is equal to* $\mathbf{f} = (-1, 0, 0, 1, 1, -1, 1)$*, and the other three are pairwise rotations of* $\mathbf{f}$*. This completes the attack, because they can all be used to decrypt encrypted messages.*

2.4.6 Practicality of the system

Object	Size(bits)
Public key $\mathbf{h}$	$n \log_2(q)$

Table 2.10: Public key size in NTRU.

The public key of NTRU consists of a vector of length n with entries ranging from 0 to $q-1$. Therefore, it requires $O(n \log_2(q))$ bits of storage space. The time complexity of the different operations depend on both n and the different integer bounds d_f, d_g and d_r. The most recent recommended parameters have appeared in [36, Table 1]. The authors make a distinction between parameters that are recommended to minimize the cost of space, speed, or a trade-off between these two. Examples of such parameters for (n, d_f) are $(449, 134)$ and $(1087, 120)$ for minimizing the space cost, and $(761, 42)$ and $(1499, 79)$ for maximizing the speed.

Several improvements have been suggested to increase the efficiency. A private key of the form $\mathbf{f} = \mathbf{e}_1 + p\mathbf{f}'$ can be used (where $\mathbf{e}_1$ is the first unit vector and the entries of $\mathbf{f}'$ are randomly $1, 0, -1$ depending on d_f as before), in order to increase efficiency by removing a multiplication step later. This removes the need for $[C^*\mathbf{f}]$ to be invertible $\bmod p$. It is also possible to take $p = X + 2$ rather than $p = 3$.

2.4.7 Conclusion

Although the NTRUEncrypt cryptosystem is defined using truncated polynomials, its most effective attacks make use of the lattice structure of the keys. This gives rise to lattice-based attacks that use lattice reduction algorithms, which work better than theoretically expected in low dimensions. But unlike GGH and AD, NTRU is efficient enough to raise the security parameters while staying competitive. This gives rise to the question whether it is possible to add structure to the lattices used in lattice-based cryptography without decreasing the security, in order to increase the efficiency.

2.5 Conclusions

In this chapter, three different systems that are somehow related with lattices were described. In the case of GGH, the cryptosystem is explicitly based on and defined in terms of lattices. The description of the Ajtai-Dwork system barely mentions lattices at all, yet its security is provably related to hard lattice problems. The NTRU system is regarded as a lattice-based cryptosystem due to the lattice structure that arises from the relation between the public and private key. This structure gives rise to a natural attack using lattice reduction.

Not all of the early lattice-based cryptosystems were successful. The GGH and AD cryptosystems were not efficient enough to cope with the good performance of lattice reduction algorithms in lower dimensions. The NTRU system is efficient enough to hold out for now, but improvements in lattice reduction techniques could change this. It appears that lattice reduction is not yet very well understood. There does not yet seem to be a simple explanation for the fact that lattice reduction algorithms work better than theoretically expected in low dimensions.

These first systems give rise to some interesting questions about lattice-based cryptography. It is known that several lattice problems are hard to solve, but is it possible to base a cryptosystem on these problems without introducing weaknesses such as in GGH? Are there perhaps other (hard) lattice problems that can be used for cryptosystems? Can extra structure be added to the lattices without decreasing the security, in order to increase the efficiency of the system such as in NTRU? These questions will be examined in the following chapter.

Chapter 3

Current developments in Lattice-Based Cryptography

New developments in the area of lattice-based cryptography are fueled by the advantages of cryptosystems based on lattices, some of which have already been mentioned. The connection between worst-case and average-case hardness of certain lattice problems, as shown by Ajtai, remains a strong reason to create cryptosystems based on these problems. Ideally this would result in a cryptosystem such that breaking the system is provably as hard as solving the problem. Furthermore, in 1994, Shor invented an algorithm for quantum computers that is able to efficiently factorize numbers and to solve the discrete logarithm problem[98]. As the research into quantum computers continues, Shor's algorithm threatens the security of systems such as RSA, ElGamal and other cryptosystems based on the discrete logarithm problem. This has prompted a search for so-called "post-quantum" alternatives to cryptosystems based on these problems. Lattice-based cryptography is one of several candidates that are possibly resistant to attacks from quantum computers. Thus, lattice problems seem to be an attractive option as a basis for public-key cryptography.

However, as shown in Chapter 2, the first attempts at lattice-based cryptosystems demonstrated several drawbacks. The difficulty of some of the underlying lattice problems was not well understood and two of the three cryptosystems have been broken – one of those even despite a security proof. These failures were caused by inefficiency of the cryptosystems (due to relatively large key sizes), combined with the unexpectedly good performance of lattice reduction algorithms in practical dimensions. Thus, the following questions arose: Is it possible to provably base cryptosystems on harder lattice problems? Is it possible to increase the efficiency of the cryptosystems by adding structure to the used lattices, while retaining the provable difficulty of the lattice problems? And finally, how can the practical security of such cryptosystems be determined, while taking into account the power of lattice reduction?

The first two questions are examined in this chapter. First, several new (variations of) lattice problems will be considered in Section 3.1, in order to better understand the difficulty of lattice problems. In Section 3.2, ideal lattices will be introduced, as a possible solution to the efficiency problem. Finally, some applications of these new lattice problems and ideal lattices will be considered in Section 3.3. The final question on the practical security of lattice-based cryptosystems will be examined in Chapter 4.

3.1 Lattice problems

As mentioned in Section 2.2, the security of the Ajtai-Dwork cryptosystem is provably based on the n^c-unique shortest vector problem, where n is the dimension of the space and c is a constant. However, not much was known about the hardness of this problem. In 2004, Regev introduced a new cryptosystem based on the n^c-unique shortest vector problem (uSVP) with constant $c = 1.5$ [85], whereas the Ajtai-Dwork system is based on uSVP with constant $c = 8$. Later, Regev

improved his system and introduced the *Learning With Errors* problem (LWE) [86]. Based on Regev's work, Peikert introduced a system whose security is based on the decisional variant of SVP (*GapSVP*) rather than uSVP [80]. Lyubashevsky and Micciancio used Peikert's results to prove several reductions of uSVP to other lattice problems [62].

The additional problems that were used in these results will be defined in this section. These problems include the decisional variant GapSVP of the shortest vector problem, the bounded distance decoding problem, the small integer solutions problem, the shortest independent vector problem and the learning with errors problem. Often, these problems are variants or special cases of other lattice problems, but understanding the difficulty of special cases can be useful for cryptographic purposes. Afterwards, the relation between the different lattice problems will be discussed in order to give a clear picture of the different reductions between problems.

3.1.1 Decisional Shortest Vector Problem (GapSVP)

The goal of the shortest vector problem is to find an explicit shortest vector. In formal terms, this means that in Definition 1.23, the shortest vector problem has been described as a *search problem*, where the goal is to search for something. There also exist *decisional problems*, where the goal is to decide whether a statement is true for the given problem instance. The decisional variant of SVP (GapSVP) is to decide whether a short vector exists:

Definition 3.1 (GapSVP)**.** *Given a basis B of a d-rank lattice $L \subseteq \mathbb{Z}^n$, a real number r and an approximation factor γ, return YES if $\lambda_1(\mathcal{L}(B)) \leq r$ and return NO if $\lambda_1(\mathcal{L}(B)) > \gamma r$. If $r < \lambda_1(\mathcal{L}(B)) \leq \gamma r$, both YES and NO can be returned.*

Such a problem is called a *promise problem*, because it is promised that either $\lambda_1(\mathcal{L}(B)) \leq r$ or $\lambda_1(\mathcal{L}(B)) > \gamma r$ is the case. If the promise is not satisfied, the returned result will be useless, because both YES and NO are acceptable answers. Like the shortest vector problem, GapSVP_γ is NP-hard for any constant approximation factor γ (see [49]).

3.1.2 Bounded Distance Decoding (BDD)

Recall from Chapter 2 that the encryption and decryption of the GGH system is based on the closest vector problem. However, instances of the closest vector problem that arise from GGH are easier than general CVP-instances, because the distance of the target vector to the lattice is always bounded. The following problem formalizes this special case of the closest vector problem:

Definition 3.2 (Bounded Distance Decoding (BDD))**.** *Given a basis B of a d-rank lattice $L \subseteq \mathbb{Z}^n$, a distance parameter $\alpha > 0$ and a target vector $\mathbf{x} \in \text{span}(L)$ such that $\text{dist}(\mathbf{x}, L) < \alpha\lambda_1(L)$, find a lattice vector $\mathbf{u} \in L$ such that $\|\mathbf{x} - \mathbf{u}\| = \text{dist}(\mathbf{x}, L)$.*

The name *bounded distance decoding* stems from the similarities between the areas of lattices and codes. Note that the problem is basically the closest vector problem in the case that the target vector is not further than $\alpha\lambda_1(L)$ from the lattice. A slightly easier variation on this problem is to accept any lattice vector within a distance of $\alpha\lambda_1(L)$ from the target vector as a solution. It has been shown that the BDD_α problem is NP-hard for distance parameter $\alpha > 1/\sqrt{2}$ (see [58]). The BDD_α problem becomes harder as α becomes larger.

Recall from Section 2.3.4 that for the GGH cryptosystem, the distance between the target vector and the lattice is always equal to $\sigma\sqrt{n}$, where σ and n are system parameters. Thus, this distance is bounded, which suggests that it gives rise to an instance of BDD. However, instances of the closest vector problem that arise from GGH are not equivalent to instances of the bounded distance decoding problem. The error vector of GGH is the difference between the target vector and the closest lattice vector. In GGH, this error vector is not only bounded in length, but it also has a particular structure. This structure allowed Nguyen to reduce the problem to a much easier instance of BDD, as described in Section 2.3.4. In the general case of BDD, the difference vector between target and lattice does not necessarily have such a structure. Thus, while breaking the encryption of GGH can be reduced to BDD, the converse is not true.

3.1.3 Small Integer Solutions (SIS)

Recall the modular lattices from Section 1.6.5. Ajtai used the problem of finding short – not necessarily shortest – vectors in certain modular lattices as his average-case problem to show the connection between worst-case and average-case hardness for certain lattice problems. Micciancio and Regev [70] later introduced a problem similar to Ajtai's using the following definition:

Definition 3.3 (Small Integer Solution (SIS))**.** *Given a modulus q, a matrix $A \in \mathbb{Z}_q^{n \times m}$ where $m \geq n$ and a real constant ν, find a nonzero vector $\mathbf{u} \in \mathbb{Z}^m$ such that $A\mathbf{u} \equiv 0 \bmod q$ and $\|\mathbf{u}\| \leq \nu$.*

It is assumed that ν is chosen such that a solution exists. Furthermore, vectors of the form $\mathbf{x} = q\mathbf{e}_i$ always trivially satisfy $A\mathbf{x} \equiv 0 \bmod q$. Therefore, the constant ν is usually chosen to be smaller than q, such that these trivial vectors are not counted as solutions. In the notation of Section 1.6.5, the goal of this problem is to find a vector of length at most ν in the lattice

$$\Lambda_q^{\perp}(A) = \{\mathbf{x} \in \mathbb{Z}^m : A\mathbf{x} = 0 \pmod q\},$$

where A is an $n \times m$ matrix with coefficients in $\mathbb{Z}_q$. Note that this is not equivalent to the shortest vector problem in the lattice $\Lambda_q^{\perp}(A)$, although closely related. An average-case instance of the SIS problem arises when the matrix A is taken uniformly at random from all matrices in $\mathbb{Z}_q^{n \times m}$. In their paper [70], Micciancio and Regev note that solving SVP in a random lattice $\Lambda_q^{\perp}(A)$ is at least as hard as solving SIS in the average case, since ν is chosen such that a shortest vector in the lattice satisfies the length requirement of SIS. They also show that the worst-case of various lattice problems can be reduced to the average-case of the SIS problem, and hence, to the average-case of the shortest vector problem. This will be described in more detail in Section 3.1.6.

3.1.4 Shortest Independent Vector Problem (SIVP)

The next problem is similar to Ajtai's shortest basis problem (SBP), as described in Section 1.6.

Definition 3.4 (Shortest Independent Vector Problem (SIVP))**.** *Given a basis B of a d-rank lattice $L \subseteq \mathbb{Z}^n$ and an approximation factor γ, find a set of d linearly independent vectors $\mathbf{u}_1, \ldots, \mathbf{u}_d$ such that*

$$\max_{i=1}^{d} \|\mathbf{u}_i\| \leq \gamma \lambda_d(L). \tag{3.1}$$

The length restriction only holds for the longest of the vectors. Thus, the independent vectors do not need to be close to the successive minima $\lambda_1(L), \ldots, \lambda_{d-1}(L)$. Blömer and Seifert proved that SIVP is NP-hard for $\gamma = n^{1/\log\log n}$ [9]. Note that it is not required that the resulting independent vectors form a basis of the lattice. Conversely, a basis satisfying the length requirements in Equation (3.1) does not necessarily exist. Thus, there do not seem to be trivial reductions between SBP and SIVP. In Section 3.1.6 it will be shown that these problems can nonetheless be reduced to each other.

3.1.5 Learning With Errors (LWE)

Regev introduced the *Learning With Errors* problem in [86]. Before giving the problem description, some extra notation is needed. Let $q \geq 2$ be an integral modulus, let $\mathbf{s} \in \mathbb{Z}_q^n$ be an n-dimensional vector and let χ be a probability distribution on $\mathbb{Z}_q$. Now, define $A_{\mathbf{s},\chi}$ as the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$, where samples of this distribution are obtained by the following procedure:

$A_{\mathbf{s},\chi}$**:**

1. Take $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random.
2. Take $e \in \mathbb{Z}_q$ according to the distribution χ.
3. Return the tuple $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \bmod q$.

The learning with errors problem can now be formulated as follows:

Definition 3.5 (Learning With Errors (LWE)). *Given a size parameter $n \geq 1$, a modulus $q \geq 2$, a probability distribution χ on $\mathbb{Z}_q$ and an arbitrary number of independent samples from the distribution $A_{\mathbf{s},\chi}$, find $\mathbf{s}$.*

For most practical applications, the error distribution χ is taken to be a *discrete Gaussian distribution*. This is a normal distribution with mean zero and standard deviation αq that is rounded to the nearest integer. Here, $\alpha > 0$ is generally taken such that α^{-1} is a polynomial in n, i.e., $\alpha \approx 1/n^c$ for some constant c.

As with the SIS problem, LWE can be described in terms of lattices using the notation of Section 1.6.5. Consider m LWE samples $(\mathbf{a}_i, b_i) = (\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)$ from $A_{\mathbf{s},\chi}$ for $1 \leq i \leq m$. Let A be the $n \times m$ matrix that has the vectors $\mathbf{a}_i$ as its columns. Now, the matrix A has rank n with high probability. The rows of A give rise to the lattice

$$\Lambda_q(A) = \left\{ \mathbf{x} \in \mathbb{Z}^m : \mathbf{x} = A^T \mathbf{y} \pmod{q} \text{ for some } \mathbf{y} \in \mathbb{Z}^n \right\}.$$

A secret $\mathbf{s} \in \mathbb{Z}^n$ in the LWE problem now corresponds to the lattice vector $A^T\mathbf{s} \in \Lambda_q(A)$. The i'th entry of the vector $A^T\mathbf{s}$ consists of the inner product $\langle \mathbf{a}_i, \mathbf{s} \rangle$ for $1 \leq i \leq m$. Thus, writing $\mathbf{b} = (b_1, \ldots, b_m)$ and $\mathbf{e} = (e_1, \ldots, e_m)$, the LWE samples give rise to the equation

$$\mathbf{b} = A^T\mathbf{s} + \mathbf{e}.$$

The goal of the LWE problem is to find $\mathbf{s}$. This is equivalent to finding $A^T\mathbf{s}$, because the matrix A has rank n with high probability. Since $A^T\mathbf{s}$ is a lattice vector of $\Lambda_q(A)$, LWE can be described as a closest vector problem on this lattice. Depending on the choice of the error vector $\mathbf{e}$, $A^T\mathbf{s}$ will be the closest lattice vector to $\mathbf{b}$ in the lattice $\Lambda_q(A)$. For practical applications, the error distribution χ is chosen such that $\mathbf{e}$ is bounded with high probability. This means that LWE can be described as an instance of BDD, rather than the more general case of CVP. It should be noted that the error distribution gives some information on the structure of the error vector. Thus, the LWE problem is essentially a bounded distance decoding problem in the lattice $\Lambda_q(A)$, given a hint in the form of samples from the error distribution.

Using this description, the learning with errors problem can be defined using the parameters n (the length of the secret), m (the number of noisy inner products), q (the modulus) and α, where α indicates that the error is drawn from the discrete Gaussian distribution with standard deviation $r = \alpha q/\sqrt{2\pi}$. Another consequence of this description is that it shows a connection between LWE and SIS. Recall that the goal of SIS is to find a short vector in the lattice $\Lambda_q^{\perp}(A)$ where A is an $n \times m$ matrix. Now, the goal of LWE is to find a close vector in the lattice $\Lambda_q(A)$. As described in Section 1.6.5, these two lattices are dual to one another, in the sense that $\Lambda_q(A)^{\times} = q^{-1} \cdot \Lambda_q^{\perp}(A)$ and $\Lambda_q^{\perp}(A)^{\times} = q^{-1} \cdot \Lambda_q(A)$. Hence, LWE is sometimes described as the 'dual' problem of SIS.

Definition 3.5 gives the search variant of LWE. It is also possible to consider a decisional variant, where the goal is to decide whether a set of independent samples come from the distribution $A_{\mathbf{s},\chi}$ or from the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Definition 3.6 (Learning With Errors (decision version)). *Given a size parameter $n \geq 1$, a modulus $q \geq 2$, a probability distribution χ on $\mathbb{Z}_q$ and a set of independent samples from $\mathbb{Z}_q^n \times \mathbb{Z}_q$, return YES if the samples come from $A_{\mathbf{s},\chi}$ and NO if they come from the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*

Regev showed that the search version can be reduced to the decision version, if q is polynomial in n, i.e., $q \approx n^c$ for some constant c. Since the decision version can trivially be reduced to the search version, this means that the search and decision versions of LWE are computationally equivalent if q is polynomial in n.

It is worth noting that taking $q = 2$ reduces LWE to the *Learning Parity with Noise* problem (LPN), which is a well-known problem in the field of machine learning. However, the hardness results for LWE, which will be discussed in Section 3.1.6, require q to be at least some polynomial in n.

3.1.6 Reductions

Recall from Section 1.2.3 that the difficulty of problems can be compared by using *reductions* between these problems. A reduction from a problem A to a problem B (A is reduced to B) is a transformation from the instances of problem A to instances of problem B. This means that a method that solves all instances of problem B can be used to solve the instances of problem A. Intuitively, this means that problem B cannot be easier than problem A or equivalently, problem B is at least as hard as problem A. This follows from the fact that a solution to problem B immediately gives a solution to problem A, whereas the converse is not true: a solution to problem A does not necessarily provide a solution to problem B. A summary of several reductions between the different lattice problems will be given below, to give an indication of the relative difficulty of these problems. For a more extensive description of the relation of several classical lattice problems from a cryptographic perspective, see the book on the complexity of lattice problems by Micciancio and Goldwasser [69].

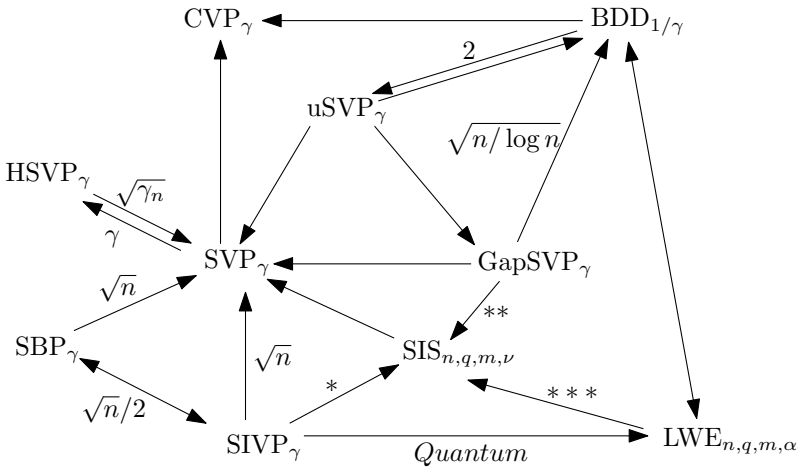

Figure 3.1: Relations among lattice problems.

Figure 3.1 shows the relations among lattice problems by depicting the reductions between lattice problems. For simplicity, each problem is assumed to be in a full-rank lattice in $\mathbb{Z}^n$ ($\mathbb{Z}^m$ for SIS and LWE). An arrow from a problem A to a problem B means that problem A can be reduced to problem B (in polynomial time). The subscript γ indicates the approximation factor of the problems, except for GapSVP, uSVP and BDD. For GapSVP, γ indicates the gap of the promise problem, for uSVP, γ indicates the gap λ_2/λ_1 and for BDD, $1/\gamma$ indicates the distance bound. If an arrow from problem A to problem B is labeled with a factor α, this means that approximating problem A with a factor $\alpha\gamma$ reduces to approximating problem B with a factor γ. Equivalently, this means that if it is possible to approximate B with a factor γ, then it is possible to approximate A with a factor $\alpha\gamma$. The SIS and LWE problems do not have approximation versions defined and depend on several parameters. Their reductions (labeled *, ** and ***) depend on the choice of these variables, and will be described in more detail. The reduction from SIVP to LWE is labeled *Quantum*, as it requires a quantum computer. Note that if there exists a directed path from a problem A to a problem B, then problem A can be reduced to problem B as well. However, these composite reductions were omitted to avoid an illegible figure.

Consider the Hermite shortest vector problem, as defined in Section 1.6. By definition of Hermite's constant γ_n, the length of a shortest vector can be bounded using $\lambda_1(L)^2/\operatorname{vol}(L)^{2/n} \leq \gamma_n$, which implies $\lambda_1(L) \leq \sqrt{\gamma_n}\operatorname{vol}(L)^{1/n}$. Thus, a solution to approximate SVP with approximation factor γ is also a solution to HSVP with approximation factor $\gamma\sqrt{\gamma_n}$, since $\gamma\lambda_1(L) \leq \gamma\sqrt{\gamma_n}\operatorname{vol}(L)^{1/n}$. Conversely, Lovász showed in [59] that any algorithm that solves HSVP with Hermite factor γ can be used to solve approximate SVP with approximation factor γ^2.

Next, recall the shortest independent vector problem and the shortest basis problem. As shown by Micciancio and Goldwasser in [69], the length of the longest vector in the shortest basis can be at most $\sqrt{n}/2\lambda_n(L)$. Thus, any solution to SBP with approximation factor γ is a solution to SIVP with approximation factor $\sqrt{n}/2\gamma$. Furthermore, there exists an algorithm that efficiently computes a basis from *any* set of n linearly independent lattice vectors, such that the length of each vector is increased by at most $\sqrt{n}/2$. Thus, any solution to SIVP with approximation factor γ can be transformed into a solution of SBP with approximation factor $\sqrt{n}2\gamma$ by using this algorithm. Finally, it is also shown by Micciancio and Goldwasser in [69] that both SBP and SIVP can be reduced to problems that can themselves be reduced to SVP. However, these reductions are not tight and a factor $\sqrt{n}$ is 'lost'.

The reduction from SVP to CVP was shown in Section 1.4.3. The reduction from BDD to CVP is trivial, because CVP is merely BDD without the distance bound. The reduction from GapSVP to SVP is trivial as well, because a solution to the search problem solves the decision problem. As for the reduction from uSVP to SVP, any algorithm for SVP that achieves an approximation factor $\leq \gamma$ can solve the unique shortest vector problem with gap γ. The algorithm will find a vector $\mathbf{v}$ of length at most $\|\mathbf{v}\| \leq \gamma\lambda_1(L)$, which, by definition of the unique shortest vector problem, is a multiple of $\mathbf{u}$. Consider the coordinates of $\mathbf{u}$ with respect to any basis of the lattice. If $\mathbf{u}$ is divided through by the greatest common divisor of these coordinates, the shortest vector is retrieved.

The reductions among uSVP, GapSVP and BDD were formalized by Lyubashevsky and Micciancio in [62]. Here, it is first shown that BDD with distance bound $\frac{1}{2\gamma}$ can be reduced to uSVP with gap γ, for any $\gamma \geq 1$. Then, it is shown that uSVP with gap γ can be reduced to BDD with distance bound $1/\gamma$, if γ is polynomially bounded in n, i.e., there exists a constant c such that $\gamma(n) \leq n^c$. For such polynomially bounded γ, it is shown that uSVP with gap γ reduces to GapSVP with approximation factor γ. Finally, it is shown that, for any $\gamma > 2\sqrt{n/\log n}$, GapSVP with approximation factor $\sqrt{n/\log n}\gamma$ can be reduced to BDD with distance bound $1/\gamma$.

When Micciancio and Regev introduced the SIS problem in [70], they also described the reductions from SIVP and GapSVP to SIS. As mentioned SIS can be trivially reduced to SVP, because a solution of SVP in the lattice $\Lambda_q^{\perp}(A)$ is a solution of SIS. As for the reduction from SIVP to SIS (labeled * in Figure 3.1), they showed that for all $m = \Theta(n \log n)$, there exists a modulus $q = O(n^2 \log n)$ such that for all $\gamma = \omega(n \log n)$, SIVP with approximation factor γ can be reduced to the *average case* of SIS with parameters (n, q, m, ν). Similarly, for the reduction from GapSVP to SIS (labeled ** in Figure 3.1), they showed that for all $m = \Theta(n \log n)$, there exists an odd modulus $q = O(n^{2.5} \log n)$ such that for all $\gamma = O(n\sqrt{\log n})$, GapSVP with approximation factor γ can be reduced to SIS with parameters (n, q, m, ν).

As mentioned in its description, the learning with errors problem is essentially equivalent to a bounded distance decoding problem in the lattice $\Lambda_q(A)$, with a hint in the form of samples from the error distribution. Here, the error distribution is the discrete Gaussian distribution. In [86], Regev gives a reduction from approximate versions of SVP and SIVP to LWE. However, this reduction uses a quantum computer, which means that solving LWE is at least as hard as solving SIVP *with a quantum computer*. This reduction bases the hardness of LWE on the quantum-hardness of SVP and SIVP and it is not known whether SVP and SIVP are hard to solve using a quantum computer. However, using the equivalence of LWE and BDD combined with the reductions from Lyubashevsky and Micciancio in [62], it is possible to reduce GapSVP to LWE (without a quantum computer). This implies that a solution to LWE provides a solution to GapSVP, making LWE at least as hard as GapSVP. Thus the hardness of LWE is based on that of GapSVP. However, there is one caveat: this requires the modulus q to be exponential in n. If q is polynomial in n, then LWE is as hard as GapSVP, given a 'hint' in the form of a 'short' basis.

Finally, the reduction from LWE to SIS is described by Micciancio and Regev in [71]. Here it is shown that a short vector in the dual lattice can be used to solve the decisional LWE problem. Recall from the description of LWE that finding a short vector in the dual lattice is exactly the SIS problem. If this vector has a length of at most $1.5\sqrt{2\pi}/(\alpha q)$, then it can be used to solve the decisional LWE problem. Thus, LWE with parameters (n, q, m, α) reduces to SIS with parameters $(n, q, m, \nu = 1.5\sqrt{2\pi}/\alpha)$.

3.2 Ideal lattices

In the description of the NTRU cryptosystem in Chapter 2, it was shown that adding extra structure to lattices can improve the efficiency of representing these lattices. To see whether such structured lattices could be used to improve the efficiency of other lattice-based cryptosystems, Micciancio examined *cyclic lattices* in [67]. He constructed a one-way function based on a combination of knapsacks and these cyclic lattices, leaving as an open problem whether the one-way function is also collision resistant. This open problem was independently answered by Lyubashevsky and Micciancio in [61] and by Peikert and Rosen in [81]. It was shown there that the one-way function is not collision resistant, but can be made collision resistant by some adaptations, which led to the closer examination of ideal lattices.

3.2.1 Definition

The name 'ideal lattice' stems from the fact that the set of all lattice vectors forms an ideal in a certain ring. An ideal is a special type of subset, defined as follows:

Definition 3.7. *Let $(R, +, \cdot)$ be any ring with 'addition' operation $+$ and 'multiplication' operation $\cdot$. Denote the inverse of the addition operation by $-$. Then, a non-empty subset $I \subseteq R$ is called an ideal of R if*

1. *for all $x, y \in I$, $x - y \in I$ and*
2. *for all $x \in I$ and $r \in R$, $r \cdot x \in R$.*

Note that the first requirement on I in Definition 3.7 is equivalent to I being a subgroup of R under the addition operation. For any subset $S \subset R$, the minimal ideal that contains S is denoted by $\langle S \rangle$. If S consists of one element $f \in R$, then the ideal $\langle f \rangle$ consists of all multiples of f, i.e., $\langle f \rangle = \{x \cdot f : x \in R\}$. Let 0 be the identity element with respect to the addition operation and let 1 be the identity element with respect to the multiplication operation. Then $\langle 0 \rangle = \{0\}$ and $\langle 1 \rangle = R$ are two trivial ideals of R.

As an ideal I is a subgroup of R, it is possible to consider the congruence classes of the quotient R/I. Here, R is divided into congruency classes modulo I, where two ring elements f and g are congruent modulo I if $f - g \in I$. As these quotient rings are themselves rings, they have ideals as well. Before formally defining ideal lattices, cyclic lattices will be considered first.

Cyclic lattices

Cyclic lattices are lattices $L \subseteq \mathbb{Z}^n$, where for every lattice vector $(u_1, u_2, \ldots, u_n) \in L$ the cyclically shifted vector $(u_n, u_1, \ldots, u_{n-1}) \in L$ as well. These cyclic lattices are closely related to ideals, as will be shown in the proposition below. Recall the convolution ring of truncated polynomials from Section 2.4.1. These are polynomials in the quotient ring $\mathbb{Z}[x]/\langle x^n - 1 \rangle$, where $\langle x^n - 1 \rangle$ is the ideal in the ring $\mathbb{Z}[x]$ consisting of all multiples of $x^n - 1$.

It is possible to see vectors $\mathbf{u} \in Z^n$ as a coefficient vector of a truncated polynomial in the quotient ring $\mathbb{Z}[x]/(x^n - 1)$. The vector $\mathbf{u} = (u_1, \ldots, u_n)$ corresponds to the polynomial

$$\sum_{i=1}^{n} u_i x^{i-1} = u_1 + u_2 x + \ldots + u_n x^{n-1}. \tag{3.2}$$

In the following, the notation $\mathbf{u}$ will be used interchangeably for both the vector and the corresponding polynomial as in (3.2). The fact that L is a cyclic lattice now translates to: for any lattice vector $\mathbf{u}$, the vector $x\mathbf{u}$ is in the lattice as well, since

$$x\mathbf{u} = x \sum_{i=1}^{n} u_i x^{i-1} = u_1 x + \ldots + u_n x^n \equiv u_n + u_1 x + \ldots + u_{n-1} x^{n-1} \pmod{x^n - 1}. \tag{3.3}$$

Note that the right-hand side corresponds to the cyclic rotation of $\mathbf{u}$: $(u_n, u_1, \ldots, u_{n-1})$. Thus, a cyclic rotation of the vector is equivalent to multiplying the corresponding polynomial by x. This leads to the following proposition:

Proposition 3.8. *Let L be a lattice with a basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$. Then L is a cyclic lattice if and only if it is isomorphic to the ideal $I = \langle \mathbf{b}_1, \ldots, \mathbf{b}_n \rangle \subseteq \mathbb{Z}^n/\langle x^n - 1 \rangle$.*

The following observation explains why a cyclic lattice L always corresponds to an ideal $I \subset \mathbb{Z}[x]/\langle x^n - 1 \rangle$. Because a cyclic rotation in L corresponds to multiplying the polynomial with x as shown in Equation (3.3), it follows that if $\mathbf{u} \in L$, then $x\mathbf{u} \in L$ as well. Inductively, this means that $x^i\mathbf{u}$ is also in L for any integer i, since this corresponds to cyclically rotating $\mathbf{u}$ i times. Let $\mathbf{u} \in L$ be a lattice vector corresponding to a polynomial and let $\mathbf{h} \in \mathbb{Z}[x]/\langle x^n - 1 \rangle$ be a polynomial with vector representation $(h_1, \ldots, h_n)$. Now consider the vector that corresponds to the polynomial $\mathbf{h} * \mathbf{u}$, where the $*$ is the convolution multiplication as defined in Section 2.4.1:

$$\begin{aligned} \mathbf{h} * \mathbf{u} &= \left(\sum_{i=1}^{n} h_i x^{i-1} \right) \mathbf{u} \pmod{x^n - 1} \\ &= \sum_{i=1}^{n} h_i (x^{i-1}\mathbf{u}) \pmod{x^n - 1}. \end{aligned}$$

Since $x^{i-1}\mathbf{u}$ are all lattice vectors and the h_i are all integers, the expression in this equation is merely a linear combination with integral coefficients of lattice vectors. Hence, the result $\mathbf{h} * \mathbf{u} \in L$ for any element in $\mathbb{Z}[x]/\langle x^n - 1 \rangle$. The other requirement of an ideal is that I is an additive subgroup, but this follows from the fact that L itself is an additive subgroup of $\mathbb{R}^n$. Thus, L corresponds to an ideal in $\mathbb{Z}[x]/\langle x^n - 1 \rangle$.

Why are cyclic lattices interesting? As shown for the NTRU cryptosystem, lattices with such extra structure can be represented more compactly. Lattices of rank n can be represented by only one vector of length n. Furthermore, the algebraic structure allows for fast arithmetic when using the *Discrete Fourier Transform*, or rather its fast implementation, the *Fast Fourier Transform* (FFT).

However, it turns out that the choice of the polynomial $x^n - 1$ is not optimal. This is because $x^n - 1$ is not *irreducible*, i.e., it can be reduced into factors of lower rank such as $x^n - 1 = (x - 1)(1 + x + \ldots + x^{n-1})$. Both Peikert and Rosen [81] and Lyubashevsky and Micciancio [61] used this reducibility to show that Micciancio's one-way function is not collision resistant. Peikert and Rosen resolved this weakness by adding constraints on which ring elements (from $\mathbb{Z}[x]/\langle x^n - 1 \rangle$) should be used in the function. Lyubashevsky and Micciancio [61] instead examined what would happen if $x^n - 1$ was replaced by a different polynomial, which led to the more general case of ideal lattices.

Ideal lattices

Ideal lattices are lattices that correspond to ideals in the ring $\mathbb{Z}[x]/\langle f \rangle$ for some *monic* polynomial $f \in \mathbb{Z}[x]$ of degree n. Here, monic means that the *leading coefficient* of f is equal to 1, i.e., f is of the form $f = x^n + \sum_{i=0}^{n-1} f_i x^i$. This ensures that reduction modulo f will result in a polynomial of degree $< n$.

Note that cyclic lattices are ideal lattices with $f = x^n - 1$. This choice of f did not appear to be optimal because it can be reduced into factors of lower degree. What would happen if f were replaced by an irreducible polynomial? The following proposition shows an important consequence of the irreducibility of f.

Proposition 3.9. *Let $f \in \mathbb{Z}[x]$ be a monic and irreducible polynomial of degree n and let $\mathbf{v} \in \mathbb{Z}[x]/\langle f \rangle$ be a nonzero polynomial corresponding to a vector in $\mathbb{Z}^n$. Then, the vectors corresponding to $\mathbf{v}, x\mathbf{v}, x^2\mathbf{v}, \ldots, x^{n-1}\mathbf{v}$ are linearly independent.*

Proof. Consider a linear combination of the vectors corresponding to $\mathbf{v}, x\mathbf{v}, \ldots x^{n-1}\mathbf{v}$ with integral coefficients:

$$\sum_{i=0}^{n-1} g_i(\mathbf{v}x^i) = \left(\sum_{i=0}^{n-1} g_i x^i\right)\mathbf{v} = \mathbf{g} * \mathbf{v},$$

where $\mathbf{g}$ is the polynomial with corresponding coefficient vector $(g_0, \ldots, g_{n-1})$. If this linear combination is equal to zero, then, equivalently, $\mathbf{g} * \mathbf{v} \equiv 0 \bmod f$. Now, by assumption f is irreducible, but because $\mathbb{Z}[x]$ is a unique factoring domain (which means that every element has a unique factoring), f is also a prime element of $\mathbb{Z}[x]$. This means that if f divides ab, then f divides a or f divides b (or both). Since f divides $\mathbf{g} * \mathbf{v}$ and f is prime, it follows that f must divide $\mathbf{g}$ or $\mathbf{v}$. However, the degree of both $\mathbf{g}$ and $\mathbf{v}$ is at most $n-1$, which means that $\mathbf{g}$ must be zero. As a consequence, the vectors corresponding to $\mathbf{v}, x\mathbf{v}, x^2\mathbf{v}, \ldots, x^{n-1}\mathbf{v}$ are linearly independent. □

Proposition 3.9 implies that nontrivial ideal lattices are always full-rank if f is irreducible. Note that, unlike in cyclic lattices, multiplying by x does not correspond to a cyclic rotation for a general polynomial f. It is not even guaranteed that the vector corresponding to $x\mathbf{v}$ has the same norm as $\mathbf{v}$, which was the case for cyclic lattices. This all depends on what happens when a polynomial $\mathbf{g}$ of degree $\geq n$ is reduced modulo f. Consider for example $f = x^n - 2x^{n-1}$, $\mathbf{v} = x^{n-1}$ and $\mathbf{g} = x^{n-1}\mathbf{v}$. Then, $\mathbf{g} = x^{2n-2} \equiv 2^{n-1}x^{n-1}$, which has a much higher norm than $\mathbf{v}$.

To solve this, Lyubashevsky and Micciancio consider an extra requirement on the polynomial f. Informally, this requirement says that for any polynomial $\mathbf{g}$, the ring norm $\|\mathbf{g}\|_f := \|\mathbf{g} \bmod f\|_\infty$ cannot be much bigger than $\|\mathbf{g}\|_\infty$. To measure the suitability of different polynomials f in this regard, they define the *expansion factor*. This gives an indication of how much the coefficients of a polynomial are expanded by reducing modulo f. By examining this expansion factor, they have found several appropriate choices for f. Two of these are worth noting here:

- $x^{n-1} + x^{n-2} + \ldots + x + 1$, where n is prime; and
- $x^n + 1$, where n is a power of two.

Note that the first is one of the factors of $x^n - 1$ and that it is irreducible for prime n. Likewise, the second is irreducible when n is a power of two. Of these two, $x^n + 1$ has half the expansion factor of $x^{n-1} + x^{n-2} + \ldots + x + 1$. In the following, several consequences of the choice $f = x^n + 1$ will be examined.

3.2.2 Lattice problems in ideal lattices

The choice of f strongly influences the hardness of the lattice problems, as will be shown here. Take for example $f = x^n + 1$ where n is a power of two. Whereas the polynomial $x^n - 1$ leads to cyclic lattices, $x^n + 1$ will lead to so-called *anti-cyclic* lattices. Recall that multiplying a polynomial in the ring $\mathbb{Z}[x]/\langle x^n - 1\rangle$ by x corresponds to a cyclic shift in the coefficient vector, as shown in Equation (3.3). A similar result is possible for the ring $\mathbb{Z}[x]/\langle x^n + 1\rangle$. For any polynomial $\mathbf{u} \in \mathbb{Z}[x]/\langle x^n + 1\rangle$, multiplying by x gives

$$x\mathbf{u} = x\sum_{i=1}^{n} u_i x^{i-1} = u_1 x + \ldots + u_n x^n \equiv -u_n + u_1 x + \ldots + u_{n-1}x^{n-1} \pmod{x^n + 1}.$$

Note that the right-hand side corresponds to a cyclic rotation of $\mathbf{u}$, except that the rotated element was multiplied by -1: $(-u_n, u_1, \ldots, u_{n-1})$. This is also called the anti-cyclic rotation of $\mathbf{u}$.

Consider a lattice L that corresponds to an ideal in the ring $\mathbb{Z}[x]/\langle x^n+1\rangle$ and let $\mathbf{u} \in L$ be a lattice vector. Because x^n+1 is irreducible when n is a power of two, the vectors $\mathbf{u}, x\mathbf{u}, x^2\mathbf{u}, \ldots, x^{n-1}\mathbf{u}$ are linearly independent by Proposition 3.9. These vectors all correspond to anti-cyclic rotations of $\mathbf{u}$ and hence have the same norm as $\mathbf{u}$. This observation has two important consequences in terms of the hardness of lattice problems.

The first consequence is that SVP is suddenly equivalent to SIVP. Take $\mathbf{u} \in L$ to be a shortest vector. Then $\mathbf{u}, x\mathbf{u}, x^2\mathbf{u}, \ldots, x^{n-1}\mathbf{u}$ are all linearly independent and have the same length as $\mathbf{u}$. Thus, finding one shortest vector immediately gives n linearly independent shortest vectors that form a solution for SIVP. Conversely, it follows that $\lambda_1(L) = \lambda_2(L) = \ldots = \lambda_n(L)$ and therefore solving SIVP requires n vectors that are also shortest vectors.

The second consequence is that $\text{GapSVP}_{\sqrt{n}}$ is suddenly easy, as shown by the following proposition:

Proposition 3.10. *Let L be an ideal lattice in $\mathbb{Z}[x]/\langle x^n + 1\rangle$. Then $\text{vol}(L)^{1/n} \leq \lambda_1(L) \leq \sqrt{n}\,\text{vol}(L)^{1/n}$.*

Proof. Recall from 1.4.4 that, by Equation (1.8), $\lambda_1(L) \leq \sqrt{\gamma_n}\,\text{vol}(L)^{1/n}$. Furthermore, as a consequence of Theorem 1.33, γ_n is essentially linear in n as shown in Equation (1.10). Thus, $\lambda_1(L) \leq \sqrt{n}\,\text{vol}(L)^{1/n}$. This holds for any lattice L, whereas the following lower bound is not true for lattices in general. Consider a shortest vector $\mathbf{u}$ of L. As shown, $\mathbf{u}, x\mathbf{u}, x^2\mathbf{u}, \ldots, x^{n-1}\mathbf{u}$ are all linearly independent and have the same length as $\mathbf{u}$, i.e., $\lambda_1(L)$. Thus, $\mathbf{u}, x\mathbf{u}, x^2\mathbf{u}, \ldots, x^{n-1}\mathbf{u}$ span a full-rank sublattice L' of L. By Lemma 1.14 and the fact that the group index is by definition ≥ 1, $\text{vol}(L) \leq \text{vol}(L')$. It follows that $\text{vol}(L) \leq \text{vol}(L') \leq \prod_{i=0}^{n-1} \|x^i\mathbf{u}\| = \lambda_1(L)^n$, which means that $\text{vol}(L)^{1/n} \leq \lambda_1(L)$, as required. □

Proposition 3.10 shows that $\text{vol}(L)^{1/n}$ is a $\sqrt{n}$-approximation of $\lambda_1(L)$ for ideal lattices L in $\mathbb{Z}[x]/\langle x^n + 1\rangle$. Recall from Definition 3.1 that for $\text{GapSVP}_{\sqrt{n}}$ a real r is given and the goal is to answer YES if $\lambda_1(L) \leq r$ and NO if $\lambda_1(L) > \sqrt{n}r$. Either answer is acceptable if $r < \lambda_1(L) \leq \sqrt{n}r$. Thus, a YES answer is acceptable if $0 < \lambda_1(L) \leq \sqrt{n}r$ and a NO answer is acceptable if $r < \lambda_1(L)$. Now, if $\text{vol}(L)^{1/n} \leq r$, then $\lambda_1(L) \leq \sqrt{n}\,\text{vol}(L)^{1/n} \leq \sqrt{n}r$, which means that YES is an acceptable answer. Conversely, if $\text{vol}(L)^{1/n} > r$, then $r < \text{vol}(L)^{1/n} \leq \lambda_1(L)$, which means that NO is an acceptable answer. Thus, for an ideal lattice L in $\mathbb{Z}[x]/\langle x^n + 1\rangle$, $\text{GapSVP}_{\sqrt{n}}$ can be solved by computing $\text{vol}(L)^{1/n}$ and answering YES if $\text{vol}(L)^{1/n} \leq r$ and NO if $\text{vol}(L)^{1/n} > r$.

This shows that the hardness of lattice problems can be affected by the choice of f. However, this hardness is still not understood very well for ideal lattices. This gives rise to several open problems. Is it possible to prove that certain lattice problems in ideal lattices are hard? And if not, is it possible to create new problems that are hard for ideal lattices? Is it possible to prove that approximating lattice problems on ideal lattices within certain approximation factors cannot be hard, akin to the results for CVP by Goldreich and Goldwasser[28]? Do there exist f (not necessarily irreducible) such that lattice problems in ideals of $\mathbb{Z}[x]/\langle f\rangle$ are easy to solve? Can quantum computers be used to solve lattice problems on ideal lattices? Current research aims to answer these questions and to construct more practical schemes and cryptographic primitives based on ideal lattices. In Section 3.3, some applications of ideal lattices will be described.

3.3 Applications

The LWE problem and its 'dual', the SIS problem, have been used to implement several cryptographic primitives using lattices. Additionally, ideal lattices have been used to implement cryptographic primitives based on lattice problems in ideal lattices. In this section, several of these applications will be described, as well as the cryptographic primitives involved. It should be noted that most of the applications require special trapdoor constructions. As these constructions can get quite technical, they will not be fully explained here. For each application, references will be given where these technical details can be found.

3.3.1 Public Key Encryption

The first primitive is one that has already appeared in Chapter 2: Public Key Encryption (PKE). In [86], Regev introduced a PKE cryptosystem based on the LWE problem. Its parameters consist of a security parameter n, the number of equations m, a modulus q and the error parameter α.

Depending on the security parameter n, which is an integer, choose q to be a prime between n^2 and $2n^2$, set $m = 1.1 \cdot n \log q$ and set $\alpha = (\sqrt{n} \log^2 n)^{-1}$.

Private key: Draw a secret $\mathbf{s}$ uniformly from $\mathbb{Z}_q^n$ as the private key.

Public key: Choose m samples from the LWE distribution with secret $\mathbf{s}$, modulus q and the discrete Gaussian distribution with standard deviation $\alpha q/\sqrt{2\pi}$ as error distribution. The public key consists of these m samples $(\mathbf{a}_i, b_i)_{i=1}^m$.

Encryption: Choose a random subset $S \subset \{1, \ldots, m\}$ uniformly. For a zero bit, the encryption is $(\sum_{i \in S} \mathbf{a}_i, \sum_{i \in S} b_i)$, while the encryption for a one bit is $(\sum_{i \in S} \mathbf{a}_i, \lfloor \frac{q}{2} \rfloor + \sum_{i \in S} b_i)$.

Decryption: For any ciphertext pair $(\mathbf{a}, b)$, compute $(b - \langle \mathbf{a}, \mathbf{s} \rangle) \bmod q$. The ciphertext is decrypted as a zero if this is closer to 0 or q than to $\frac{q}{2}$ and as a one otherwise.

To see that this system works correctly, first consider the public key. It consists of m LWE samples with secret $\mathbf{s}$. These samples are of the form $(\mathbf{a}_i, b_i)_{i=1}^m$, such that $\langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \equiv b_i \bmod q$. Here, e_i is taken from the discrete Gaussian distribution with standard deviation $\alpha q/\sqrt{2\pi}$. Now consider the encryption of a single bit k. This is equal to $(\sum_{i \in S} \mathbf{a}_i, k \lfloor \frac{q}{2} \rfloor + \sum_{i \in S} b_i)$ for some random subset $S \subset \{1, \ldots, m\}$. Let $\mathbf{a} = \sum_{i \in S} \mathbf{a}_i$ and let $b = k \lfloor \frac{q}{2} \rfloor + \sum_{i \in S} b_i$. Then, computing $b - \langle \mathbf{a}, \mathbf{s} \rangle$ gives

$$\begin{aligned} b - \langle \mathbf{a}, \mathbf{s} \rangle &= k \left\lfloor \frac{q}{2} \right\rfloor + \sum_{i \in S} b_i - \langle \sum_{i \in S} \mathbf{a}_i, \mathbf{s} \rangle \\ &= k \left\lfloor \frac{q}{2} \right\rfloor + \sum_{i \in S} b_i - \langle \mathbf{a}_i, \mathbf{s} \rangle \\ &\equiv k \left\lfloor \frac{q}{2} \right\rfloor - \sum_{i \in S} e_i \pmod q. \end{aligned}$$

Thus, without the errors e_i, $b - \langle \mathbf{a}, \mathbf{s} \rangle$ would be equivalent to $k \lfloor \frac{q}{2} \rfloor \bmod q$. This suggests that $b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod q$ is close to $\lfloor \frac{q}{2} \rfloor$ if $k = 1$ and close to 0 (or q) if $k = 0$. Indeed, for a decryption error to occur, $\sum_{i \in S} e_i$ must be greater than $q/4$. The sum has at most m terms and each of the e_i is roughly distributed as a normal random variable with mean 0 and standard deviation αq. Hence, the sum is again roughly normally distributed with mean 0 and a standard deviation of at most $\sqrt{m} \alpha q = q \frac{1.1 n \log q}{\sqrt{n} \log^2 n} < q/\log n$. The probability that a normally distributed random variable of mean 0 and standard deviation $q/\log n$ is greater than $q/4$ is negligible, as $q/\log n \ll q/4$ for large n.

It is easy to see that retrieving the secret key from the public key amounts to solving the LWE problem, as the public key consists of LWE samples that correspond to the secret key $\mathbf{s}$. Proving the security of the encryption is slightly less straightforward. Imagine that there exists an algorithm that can distinguish between encryptions of zeroes and encryptions of ones, given the public key. Now consider a similar encryption scheme where the only difference is that the public key $(\mathbf{a}_i, b_i)_{i=1}^m$ is chosen from the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$, rather than the LWE distribution. It can be shown that with very high probability, $(\mathbf{a}_i, b_i)_{i=1}^m$ will be chosen such that the distribution of the random sums $(\sum_{i \in S} \mathbf{a}_i, \sum_{i \in S} b_i)$ is 'extremely close' to the uniform distribution in statistical distance. As a result, encryptions of both zeroes and ones are distributed near to uniform in this scheme and the algorithm that breaks the LWE scheme should not be able to break this scheme. Now, this algorithm can correctly guess ciphertexts that come from the LWE distribution, but not ciphertexts that come from the uniform distribution. This allows the algorithm to distinguish between samples from the LWE distribution and samples from the uniform distribution, which is the decisional LWE problem. Thus, breaking the encryption implies a solution to the decisional LWE problem. The decisional LWE problem has been proven to be equivalent to the LWE problem, as mentioned in Section 3.1.

3.3.2 CCA-Secure PKE

An interesting type of attack against public key encryption schemes is the *Chosen Ciphertext Attack* (CCA). In this attack, an adversary attempts to retrieve the secret key by decrypting several ciphertexts of his own choosing and looking at the corresponding plaintext. This might seem counterintuitive, as being able to decrypt should only be possible when possessing the secret key. However, there are practical applications where an adversary has access to a so-called *decryption oracle* which allows him to decrypt (a limited number of) ciphertexts.

In [82], Peikert and Waters present a CCA-secure cryptosystem based on the LWE problem. To this end, they create a concept they call *lossy trapdoors*. These lossy trapdoors can behave in one of two different ways. They can serve as a trapdoor function, allowing someone with the trapdoor information to obtain pre-images, but they can also be 'lossy'. Here, being lossy means that the function loses a lot of information, which means that every output has many pre-images. These two behaviors are indistinguishable, in the sense that when given a description of the lossy trapdoor, an adversary cannot tell which behavior it will exhibit. For more details on these lossy trapdoors, see the article by Peikert and Waters [82].

In [80], the article where Peikert shows a connection between LWE and GapSVP, he also gives a new construction for a CCA-secure cryptographic scheme. Peikert's scheme relies on his earlier work with Waters [82], but uses the trapdoor construction from his work with Gentry and Vaikuntanathan [26]. Informally, these trapdoors consist of a 'good' basis of the lattice created by the LWE problem, similar to the GGH cryptosystem [30]. For more information on the CCA-secure cryptosystem, see the article by Peikert [80].

3.3.3 Identity-Based Encryption

Identity-Based Encryption (IBE) is a type of encryption scheme, proposed by Shamir in [97], where any string can serve as a public key. Specifically, one's name or identity can serve as a public key. Secret keys are managed by some authority who has a 'master secret key' of the system. Such a system would remove the need of a central public key repository, since each identity serves as the public key for that identity. This would reduce the possibility of a man-in-the-middle attack.

In [26], Gentry, Peikert and Vaikuntanathan construct trapdoors for lattices which they use for several new cryptographic constructions. One of these constructions is an identity-based encryption scheme based on the LWE problem. To this end, they define a 'dual' version of the LWE encryption scheme as described above, where the roles of the keys and the ciphertext are essentially swapped.

Private key: Draw an error vector $\mathbf{e}$ from the discrete Gaussian distribution $D_{\mathbb{Z}^m,r}$ as the private key.

Public key: The public key consists of $\mathbf{u} = A\mathbf{e} \bmod q$, also called the *syndrome* of $\mathbf{e}$.

Encryption: Choose a random secret $\mathbf{s} \in \mathbb{Z}_q^n$ uniformly, draw a single error x from the error distribution χ and create a vector $\mathbf{x} = (x_1, \ldots, x_m)$ where each x_i comes from χ. Then, compute $\mathbf{p} = A^T\mathbf{s} + \mathbf{x} \in \mathbb{Z}_q^m$ and output the ciphertext $(\mathbf{p}, c = \langle \mathbf{u}, \mathbf{s} \rangle + x)$ for a zero bit, and $(\mathbf{p}, c = \langle \mathbf{u}, \mathbf{s} \rangle + x + \lfloor \frac{q}{2} \rfloor)$ for a one bit.

Decryption: For any ciphertext pair $(\mathbf{p}, c)$, compute $(c - \langle \mathbf{e}, \mathbf{p} \rangle) \bmod q$. The ciphertext is decrypted as a zero if this is closer to 0 or q than to $\frac{q}{2}$ and as a one otherwise.

One advantage of this construction is that the ciphertexts now consist of LWE samples as well and being able to distinguish them from uniform is equivalent to the decisional LWE problem. Gentry, Peikert and Vaikuntanathan prove that this system is secure against *Chosen-Plaintext Attacks* (CPA) and that it is *anonymous*. Here, anonymous means that it is not possible to see which public key was used to encrypt the message, which is a desirable property for identity-based encryption schemes.

To turn this system into an IBE scheme, the authors use their trapdoor constructions. Specifically, they bind identities to public keys $\mathbf{u}$ using a random oracle. Recall that in the dual system,

$\mathbf{u} = A\mathbf{e} \bmod q$. To compute the corresponding secret key $\mathbf{e} = f_A^{-1}(\mathbf{u})$, they use a pre-image sampler with the trapdoor information (a 'good' basis for the corresponding LWE lattice). The encryption and decryption is done using the dual encryption scheme as described above. For more details, see their paper [26].

3.3.4 Hierarchical Identity-Based Encryption

In [43], Horwitz and Lynn introduced *Hierarchical Identity-Based Encryption* (HIBE) as an extension of the IBE concept. The idea is that a hierarchy is created to distribute keys. In an IBE scheme, every user obtains their key from one root that has the master key. In a HIBE scheme, there is a hierarchy of several levels, each getting their keys from a higher level and distributing keys to a lower level. For example, in a HIBE scheme of two levels, there is one root who distributes the keys to 'domain managers', who in turn can distribute keys to their users. This leads to a tree, where each node distributes the keys to its children. To this end, each node can use its own key to derive the secret key of its children. As a consequence, all ancestors of a node can derive the secret key of that particular node.

In [10], Cash et al. propose a HIBE scheme based on hard lattice problems. To this end, they present a new cryptographic notion called *bonsai tree* and realize the concept using lattices. The name bonsai tree stems from the art form with the same name, where miniature trees are grown in containers. In their article, the authors compare the bonsai trees to the tree structure that appear in HIBE schemes and note several other similarities. They distinguish between *natural* and *controlled* growth and describe cryptographic techniques with similar properties.

As with the system of Gentry, Peikert and Vaikuntanathan [26], the HIBE scheme is based on the LWE problem. The connection between the hierarchy and lattices can be described using the tree. Each node in the tree corresponds to some lattice. The corresponding lattice of a node is a sublattice for each of the lattices that corresponds to a child of the node. The lattice of a 'parent' node (in $\mathbb{R}^m$) can be retrieved from the lattice of any 'child' node (in $\mathbb{R}^{m+k}$) by restricting this lattice to the first m dimensions. Each node also has a 'short' basis for the corresponding lattice, which can easily be extended to a similar basis for the lattices of its children. To achieve this, Cash et al. adapt a method for generating lattices together with a short basis, which was first described by Ajtai in [3] and later improved by Alwen and Peikert in [5]. See the article by Cash et al. [10] for more information on the specific techniques.

3.3.5 Digital Signatures

The concept of digital signatures was first described by Diffie and Hellman as one of the possible applications of asymmetric cryptography [13]. It can be described as the 'dual' of public-key encryption. In a PKE scheme messages are encrypted using a public key and decrypted using the corresponding private key. In a digital signature scheme this is reversed: a message is *signed* using a private key and this *signature* can be verified using the corresponding public key. Digital signatures can be used to *authenticate*, i.e., prove that a message was signed by the holder of the private key. They can also be used to check the message *integrity*, i.e., see if it was not altered after being sent by the signer. Finally, they are sometimes used for *non-repudiation*, i.e., prevent the signer from denying that he signed it.

The first two digital signature schemes based on lattices were proposed by the authors of GGH [30] and those of NTRU [42]. The authors of GGH mentioned how to construct a signature scheme from their encryption scheme, but did not analyze this construction further. As a result, the signature scheme did not generate much interest until later. The authors of NTRU, Hoffstein, Pipher and Silverman, proposed the *NTRU Signature Scheme* (NSS) as a complementary scheme to the NTRU cryptosystem. However, the preliminary version of this scheme was broken by Gentry et al. [25], and its revision that the authors of NTRU sketched during the presentation of [42] was broken by Gentry and Szydlo [27]. The next attempt to create a signature scheme based on NTRU came from Hoffstein et al. in [37]. Here, they proposed a scheme called NTRUSign, which used the exact same approach as was sketched by the authors of GGH. However, Nguyen and Regev

broke these schemes in [76], by relating them to a new learning problem, which they subsequently solved.

Recently, Lyubashevsky proposed new digital signature schemes based on ideal lattices in [60]. His idea is to use the Fiat-Shamir transform [16] that turns identification schemes into digital signature schemes. However, Lyubashevsky notes that trying to apply this transformation to existing identification schemes based on lattices does not lead to an efficient scheme. This is a result of the fact that for all of these identification schemes each challenge bit requires a response of $O\left(n(\log n)^k\right)$ bits, for some $k \geq 0$ and where n is the security parameter. Lyubashevsky also notes that number-theoretic identification schemes are efficient in comparison, because they interpret the challenge string as an integer in some domain, as opposed to a string of separate 0's and 1's. This requires some underlying algebraic structure and Lyubashevsky's main goal is to use the algebraic structure of ideal lattices to represent the challenge bits collectively as opposed to individually.

He manages to construct an identification scheme based on the SIS problem for ideal lattices, but it has one minor defect. The prover cannot respond correctly to all challenges of the verifier, but fails some constant fraction ($\approx 2/3$) of the time. In these cases, the prover needs to abort the protocol and start over. Therefore, the 'commit' and 'challenge' procedures need to be repeated a few times, such that the probability that a valid prover is accepted is high enough. However, the effect of this defect can be minimized using standard techniques that reduce the length of the 'commit' part, while the 'challenge' part can remain the same. Thus, the length of the 'response' dominates the number of transmitted bits. Furthermore, the digital signature scheme that is obtained from the Fiat-Shamir transform suffers less from the defect in the identification scheme. The reason is that the signer does not need to add the 'failed' attempts to the signature, keeping its length as short as if no failures had occurred. Thus, the only efficiency that is lost to the defect is time. For more technical details on both the identification and signature schemes, see Lybashevsky's article [60].

3.3.6 Fully Homomorphic Encryption

Homomorphic encryption formalizes the notion of being able to perform computations on encrypted data, without having to decrypt the data first. This notion was introduced by Rivest, Adleman and Dertouzos in [87] under the name *privacy homomorphism*, not long after Rivest, Shamir and Adleman invented RSA [88]. Recall the description of RSA from Section 1.2.2. A message m is encrypted into a ciphertext c by computing $c = m^e \bmod N$, where e is the public exponent. Consider two messages m_1 and m_2, as well as their respective ciphertexts $c_1 = m_1^e \bmod N$, $c_2 = m_2^e \bmod N$. Multiplying these ciphertexts gives

$$c_1 \cdot c_2 \equiv m_1^e \cdot m_2^e \equiv (m_1 \cdot m_2)^e \pmod{N}.$$

Note that the right-hand side is equal to the ciphertext corresponding to the message $m_1 \cdot m_2$. Thus, using RSA it is possible to compute the multiplication of two encrypted numbers, without decrypting them in between. This means that RSA is homomorphic *with respect to multiplication.*

There also exist cryptosystems that are homomorphic with respect to addition. Encryption schemes that are homomorphic with respect to one operation are also called *partial* homomorphic encryption schemes. A scheme is a *Fully Homomorphic Encryption* (FHE) scheme if it is homomorphic with respect to both multiplication and addition. Such a scheme could be used to perform complex operations on encrypted data, as multiplication and addition can be combined to create more complex computable functions. The existence of a fully homomorphic encryption scheme has been an open question in cryptology for over thirty years, until Gentry showed one in [23] using ideal lattices.

Gentry's result can roughly be divided into three steps. First, there is an initial construction using ideal lattices, which is homomorphic for "shallow circuits". Here, a circuit stands for a theoretical circuit of electrical gates. The circuits that can be computed using Gentry's initial construction are "shallow", in the sense that this construction can (efficiently) do many additions,

but only a few multiplications. The reason for this is that each operation introduces a certain cumulative "error" and multiplications increase this error much faster than additions. Next, Gentry introduces a technique to "squash the decryption circuit", which allows him to express the decryption operation as a circuit that it is supported by his scheme. Lastly, he uses this to introduce a concept called "bootstrapping", which is a way to homomorphically decrypt the ciphertext, thus supposedly "refreshing" it, which decreases the error. As this refreshing requires two layers of encryption, the error introduced by the additional layer should be smaller than the one introduced by the original layer. Otherwise, the error will not decrease and no refreshing occurs.

While the system by Gentry is theoretically a fully homomorphic encryption scheme, it is not yet very practical. After Gentry's initial publication, Van Dijk et al. presented a second fully homomorphic scheme in [104] based on ideals over the integers as opposed to lattices. They still use many of the other tools that Gentry proposed, such as "squashing the decryption circuit" and the bootstrapping technique. Next, Stehl and Steinfeld proposed two optimizations to Gentry's scheme in [103]. The first implementation of Gentry's scheme is due to Smart and Vercauteren in [100]. Their scheme uses *principal ideal lattices* of prime determinant, which are lattices that can be represented using only two integers regardless of their dimension. Furthermore, their decryption method required only one integer to represent the secret key. Using this, they implemented the initial "somewhat homomorphic scheme", but were not able to implement the "squashing" technique, because their parameters would become too large. Gentry and Halevi continued in this direction in [24] and implemented Gentry's scheme with additional optimizations, which allows them to implement all aspects of the scheme. However, the parameters are still not fully practical, as the public-key sizes range from 70 Megabytes to 2.3 Gigabytes, depending on the dimension. For more technical details, see the respective papers.

3.4 Conclusions

In this chapter, several new lattice problems were introduced. These mostly consist of variants of classical lattice problems that better fit the standard cryptographic applications. Furthermore, the relative difficulty of these problems was described, by considering the reductions between these problems. Additionally, ideal lattices were discussed, as result of an attempt to improve the efficiency of lattice-based cryptosystems by using lattices with additional algebraic structure. Both the new lattice problems and ideal lattices gave rise to new applications, some of which were described in this chapter.

However, this chapter has mostly dealt with the theoretical side of lattice-based cryptography and its background in lattice theory. This still leaves the final question at the introduction of this chapter unanswered: how can the practical security of lattice-based cryptosystems be determined, while taking into account the power of lattice reduction? In the next chapter, lattice reduction and its practical impact on lattice-based cryptography will be examined in more detail, including several new developments in the area of lattice reduction.

Chapter 4

Lattice basis reduction

4.1 Introduction

As shown in Section 1.7, lattice basis reduction algorithms have been used to break cryptosystems long before lattices were used to build cryptosystems. Although these reduction algorithms only give approximations to the shortest vector, these approximations were good enough to break the system for practical dimensions – in special cases, the actual shortest vector would even be found. Then, in Chapter 2, it was shown that lattice-based cryptosystems are susceptible to attacks by lattice reduction as well. Again, the lattice reduction algorithms proved too strong for dimensions where cryptographic schemes such as GGH and Ajtai-Dwork are still practical.

In this chapter, lattice reduction will be examined more closely. As it is the most prominent tool to attack lattice-based cryptosystems, its performance will affect the security of such cryptosystems. Thus, examining this performance in practice not only shows what lattice reduction algorithms can do, but also gives a measure of security for lattice-based cryptosystems. Furthermore, new algorithms have been devised after LLL to improve upon its reduction. Some of these use other techniques for solving the shortest vector problem, such as enumeration, as subroutines. These will be discussed in this chapter as well.

In Section 4.2, new developments in the area of lattice reduction will be considered and some methods to solve SVP exactly are described in Section 4.3. Then, the performance of lattice reduction algorithms in practice will be discussed in Section 4.4. Finally, Section 4.5 describes a framework to measure the security using similar results on the performance of lattice reduction algorithms.

4.2 Developments in lattice reduction

Recently an excellent survey of LLL and its applications was published on the occasion of LLL's 25th birthday [79]. Specifically, it contains two chapters (one by Schnorr [94] and one by Stehlé [102]) on improvements to the original algorithm. The improvements that are described in these chapters will be discussed here first. Then, the method of enumeration that is used to solve SVP exactly is introduced, including several improvements.

4.2.1 Floating-point LLL

The original LLL-algorithm was created with rational arithmetic in mind. As it proposed a breakthrough in lattice reduction with several promising applications, it was implemented soon after its publication in 1982. Odlyzko tried to use one of these early implementations, which stayed close to the original description of LLL, to break knapsack-based cryptosystems, as described in Chapter 1. However, the proposed rational arithmetic turned out to be costly and since computing power was limited at the time, Odlyzko could only work with lattices in small dimensions. He

was one of the first who replaced the rational arithmetic by floating-point arithmetic to solve this problem.

While using floating-point arithmetic does speed up the lattice reduction process, it might also introduce floating-point errors. These errors can lead to incorrect results, and can even prevent lattice reduction algorithms from terminating. Therefore, it is interesting to study the behavior of the floating-point operations used in lattice reduction. The first implementations that used floating-point approximations as opposed to rational arithmetic were not yet theoretically sound. Thus, the correctness of the computations could not be proven by theoretical methods. The first provable variant of LLL with floating-point arithmetic was introduced by Schnorr [92] in 1988. Recently, a more efficient provable variant, dubbed L^2 was described by Nguyen and Stehlé in [77].

While provable results are useful, sometimes heuristic methods suffice. If a method works efficiently and with a high probability in practice, it is not important whether this stems from a theoretic proof or a heuristic method. One important heuristic method is due to Schnorr and Euchner [95], as it describes an actual implementation. As such, it has been the basis for all fast implementations of floating-point LLL until recently, when new algorithms such as L^2 were introduced. It closely follows the original LLL, with some additional checks to prevent problems caused by floating-point errors.

4.2.2 LLL with deep insertions

In addition to their floating-point variant on LLL, Schnorr and Euchner introduced another interesting variant on LLL in [95]. They describe a new step that replaces the swapping step of the original LLL algorithm (recall Algorithm 3 from Chapter 1) with a "deep insertion" step, which improves the quality of the resulting basis. The resulting algorithm is called LLL with deep insertions, or DEEP.

Rather than swapping the vectors $\mathbf{b}_{k-1}$ and $\mathbf{b}_k$ when the Lovász condition is violated (see Equation (1.16)), the new step "inserts" the vector $\mathbf{b}_k$ somewhere in place i for $1 \leq i \leq k-1$. Starting at $i = 1$ and moving upwards, the length of $\pi_i(\mathbf{b}_k)$ is compared to $\pi_i(\mathbf{b}_i) = \mathbf{b}_i^*$. For the first i that $\delta\mathbf{b}_i^* > \pi_i(\mathbf{b}_k)$ (where δ is as in Equation (1.16)), the basis is rearranged from $(\mathbf{b}_1, \ldots, \mathbf{b}_k, \ldots, \mathbf{b}_d)$ to $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_k, \mathbf{b}_i, \ldots, \mathbf{b}_{k-1}, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_d)$. This step is equivalent to replacing the Lovász condition (Equation (1.16)) by the following condition:

$$\begin{aligned} \delta\|\mathbf{b}_i^*\|^2 &\leq \|\pi_i(\mathbf{b}_k)\|^2 \\ &= \left\| \mathbf{b}_k^* + \sum_{j=i}^{k-1} \mu_{k,j}\mathbf{b}_j^* \right\|^2, \end{aligned}$$

for all $1 \leq i < k \leq d$.

This replacement seems to improve the quality of the resulting basis considerably, as will be shown in Section 4.4. However, no provable theoretical bounds on the length of the resulting shortest vector are known, except for the bounds that follow from the underlying LLL. Furthermore, it is unknown whether the running time is polynomially bounded in the lattice rank and the length of the input basis in bits. The time complexity is possibly superexponential. However, as will be shown in Section 4.4, DEEP appears to run reasonably fast in practice.

4.2.3 Block-Korkine-Zolotarev reduction

In 1987, Schnorr introduced a hierarchy of new reduction algorithms generalizing LLL [91]. These algorithms trade running time for the quality of the solution by compromising between HKZ-reduction and LLL-reduction(recall Definitions 1.41 and 1.44 from Chapter 1). To this end, Schnorr defines a new reduction notion, which he calls *block Korkine-Zolotarev reduction*.

Definition 4.1 (Block Korkine-Zolotarev reduction)**.** *Let $L \subset \mathbb{R}^n$ be a lattice of rank d and let β be an integer such that $2 \leq \beta \leq d$. A basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ of L is called block Korkine-Zolotarev-reduced*

(BKZ-reduced) with blocksize β and factor $\delta \in (\frac{1}{4}, 1)$ if it is size-reduced and if

$$\delta \|\mathbf{b}_i^*\|^2 \leq \lambda_1 \left(\pi_i \left(\mathcal{L}\left(\mathbf{b}_1, \ldots, \mathbf{b}_{\min(i+\beta-1,d)}\right)\right)\right)^2, \tag{4.1}$$

for $1 \leq i \leq d-1$.

Equation (4.1) is equivalent to the statement that the i'th GSO vector $\mathbf{b}_i^*$ is at most δ^{-1} times as long as a shortest vector in the lattice that is spanned by the first β vectors of $\pi_i(L)$. This lattice can also be expressed as $\mathcal{L}\left(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_{\min(i+\beta-1,d)})\right)$.

To see that this reduction notion is a compromise between the LLL and HKZ-reduction notions, consider the following facts. For $\beta = d$ and $\delta = 1$, BKZ-reduction is equivalent to HKZ-reduction. Furthermore, for $\beta = 2$, BKZ-reduction with factor δ is equivalent to LLL-reduction with factor δ. A proof of this last fact is given by Schnorr and Euchner in [95]. Here, they also provide a practical algorithm that produces a basis satisfying this reduction notion. This algorithm is called BKZ and it is currently one of the most (if not the most) practical lattice reduction algorithms.

Recall that the LLL algorithm considers two vectors at a time and swaps them if they violate the Lovász condition. This condition ensures that the swap results in a sufficient "gain" in the norms of the Gram Schmidt vectors. The BKZ algorithm considers a block of β vectors at a time and uses a subroutine to find a shortest vector in the lattice spanned by those vectors. If this new vector, whose norm is equal to the right hand side of Equation (4.1), causes the condition of this equation to be violated, it is inserted at place i. This is repeated while the index i is cycled from 1 to $d-1$, until it is satisfied for all $1 \leq i \leq d-1$ (as it is trivially satisfied for $i = d$).

The main difference with LLL is that the condition of BKZ is not as easily checked as the Lovász condition. Checking the BKZ condition requires the knowledge of a shortest vector in the projected lattice $\mathcal{L}\left(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_{\min(i+\beta-1,d)})\right)$, whereas checking the Lovász condition can be done by a simple computation with the norms and coefficients of Gram Schmidt vectors. Therefore, the BKZ algorithm cannot check all conditions simultaneously, which makes it harder to see when the algorithm is done. To solve this problem, BKZ cyclically shifts the index i and inserts short vectors where the BKZ condition is violated. It continues this process until no new vectors have been inserted $d-1$ times, in which case the lattice has not changed and the BKZ condition was not violated for each index i. But even in the steps where the shortest vector is not inserted, it has been computed using the subroutine, which is costly. The subroutine is an enumeration method that will be examined more closely in Section 4.3.

In a simple form, the BKZ algorithm looks like the following:

Algorithm 4 BKZ lattice reduction algorithm

```
Apply LLL to the basis {b_1, ..., b_d}
i ← 0
repeat
  i ← (i (mod d − 1)) + 1
  Find b_i^new such that ‖b_i^new‖ = λ_1 (L (π_i(b_i), ..., π_i(b_{min(i+β−1,d)})))
  if δ‖b_i^*‖^2 > ‖b_i^new‖^2 then
    {b_1, ..., b_d} ← {b_1, ..., b_{i−1}, b_i^new, b_i, ..., b_d}
  end if
  Apply LLL to the basis {b_1, ..., b_d}
until No new vector was inserted for d − 1 times in a row
```

As mentioned, the index i is cyclically shifted from through $1, \ldots, d-1$. For each index i, the vector $\mathbf{b}_i^{\text{new}}$ is the shortest vector in the lattice $\mathcal{L}\left(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_{\min(i+\beta-1,d)})\right)$. Hence, if $\delta\|\mathbf{b}_i^*\|^2 > \|\mathbf{b}_i^{\text{new}}\|^2$, this means the BKZ condition of Equation (4.1) is violated. In this case, $\mathbf{b}_i^{\text{new}}$ is inserted into the basis. The LLL algorithm ensures that the basis is size-reduced at all times and removes dependent vectors from the basis. In the practical algorithm due to Schnorr and Euchner, the LLL algorithm is not applied to the full basis, but only to a smaller subset of relevant basis vectors. If no new vector was inserted for $d-1$ times in a row, this means that the BKZ-condition was not

violated for $d-1$ times in a row, which means that it must hold for all indices $1 \leq i \leq d-1$. As the basis is also size-reduced by the LLL-algorithm, this proves that a basis is BKZ-reduced with blocksize β and factor δ after performing Algorithm 4.

While BKZ is a blockwise generalization of the polynomial-time LLL algorithm, no good upper bound is known on its complexity. As with the DEEP algorithm, the best upper bound known is super-exponential in the lattice rank. However, as will be shown in Section 4.4, this bound seems overly pessimistic. In fact, both the BKZ and DEEP algorithms outperform other blockwise generalizations of LLL (that are polynomial-time) in practice, despite their possibly super-exponential complexity.

4.3 Enumeration

The idea of enumeration dates back to Pohst [83], Kannan [47] and Fincke-Pohst [17]. The concept of enumeration consists of trying all possible combinations of the basis vectors and noting which vector is the shortest. Since "all possible combinations" means an infinite number of vectors, this number needs to be bounded somehow. As with lattice reduction, the concept of enumeration in lattices relies on the Gram Schmidt orthogonalization of the lattice basis. Consider a basis $\mathbf{b}_1, \ldots, \mathbf{b}_d$ of a lattice L and its GSO $\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*$. Now let $\mathbf{u} \in L$ be a shortest vector of L and let $R > 0$ be a bound such that $\lambda_1(L) = \|\mathbf{u}\| \leq R$, e.g. $R = \|\mathbf{b}_1\|$. Recall that every basis vector $\mathbf{b}_i$ can be written as a sum of Gram Schmidt vectors:

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

Now, using this and the fact that $\mathbf{u}$ is a lattice vector, it is possible to write

$$\begin{aligned} \mathbf{u} &= \sum_{i=1}^{d} \lambda_i \mathbf{b}_i = \sum_{i=1}^{d} \lambda_i \left(\mathbf{b}_i^* + \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b}_j^* \right) \\ &= \sum_{j=1}^{d} \left(\lambda_j + \sum_{i=j+1}^{d} \lambda_i \mu_{ij} \right) \mathbf{b}_j^*. \end{aligned}$$

By representing the shortest vector $\mathbf{u}$ as a sum of Gram Schmidt vectors, projections of $\mathbf{u}$ can suddenly be represented easily as well:

$$\begin{aligned} \pi_k(\mathbf{u}) &= \pi_k \left(\sum_{j=1}^{d} \left(\lambda_j + \sum_{i=j+1}^{d} \lambda_i \mu_{ij} \right) \mathbf{b}_j^* \right) \\ &= \sum_{j=k}^{d} \left(\lambda_j + \sum_{i=j+1}^{d} \lambda_i \mu_{ij} \right) \mathbf{b}_j^*. \end{aligned}$$

Furthermore, since the Gram Schmidt vectors are by construction orthogonal, the squared norms of $\mathbf{u}$ and its projections are given by

$$\begin{aligned} \|\pi_k(\mathbf{u})\|^2 &= \left\| \sum_{j=k}^{d} \left(\lambda_j + \sum_{i=j+1}^{d} \lambda_i \mu_{ij} \right) \mathbf{b}_j^* \right\|^2 \\ &= \sum_{j=k}^{d} \left(\lambda_j + \sum_{i=j+1}^{d} \lambda_i \mu_{ij} \right)^2 \|\mathbf{b}_j^*\|^2 \end{aligned}$$

Using this observation, it is possible to bound the number of vectors that needs to be enumerated until the shortest vector is found. Recall that the bound R was chosen such that $\|\mathbf{u}\| \leq R$. Since the projection of a vector cannot be longer than the vector itself, it follows that

$$\|\pi_d(\mathbf{u})\|^2 \leq \|\pi_{d-1}(\mathbf{u})\|^2 \leq \ldots \leq \|\pi_1(\mathbf{u})\|^2 = \|\mathbf{u}\|^2 \leq R^2.$$

Combining the previous two equations gives d inequalities of the form

$$\sum_{j=k}^{d} \left(\lambda_j + \sum_{i=j+1}^{d} \lambda_i \mu_{ij} \right)^2 \|\mathbf{b}_j^*\|^2 \leq R^2, \tag{4.2}$$

for $k = 1, \ldots, d$. Equation (4.2) can be used to give bounds for the unknowns $\lambda_d, \ldots, \lambda_1$, in that order. The first inequality is given by

$$\lambda_d^2 \|\mathbf{b}_d^*\|^2 = \|\pi_d(\mathbf{u})\|^2 \leq R^2.$$

Thus, it follows that $-R/\|\mathbf{b}_d^*\| \leq \lambda_d \leq R/\|\mathbf{b}_d^*\|$. Now, for any fixed $\lambda_d = \lambda_d'$ in this interval, the next inequality becomes

$$(\lambda_{d-1} + \lambda_d' \mu_{d,d-1})^2 \|\mathbf{b}_{d-1}^*\|^2 + \lambda_d'^2 \|\mathbf{b}_d^*\|^2 = \|\pi_{d-1}(\mathbf{u})\|^2 \leq R^2.$$

This inequality can be rewritten as

$$(\lambda_{d-1} + \lambda_d' \mu_{d,d-1})^2 \leq \frac{R^2 - \lambda_d'^2 \|\mathbf{b}_d^*\|^2}{\|\mathbf{b}_{d-1}\|^2}.$$

Taking the square root on both sides shows that λ_{d-1} must lie in the interval

$$-\lambda_d' \mu_{d,d-1} - \frac{\sqrt{R^2 - \lambda_d'^2 \|\mathbf{b}_d^*\|^2}}{\|\mathbf{b}_{d-1}\|} \leq \lambda_{d-1} \leq -\lambda_d' \mu_{d,d-1} + \frac{\sqrt{R^2 - \lambda_d'^2 \|\mathbf{b}_d^*\|^2}}{\|\mathbf{b}_{d-1}\|}.$$

Repeating this process leads to an iterative method to derive the interval of λ_k once $\lambda_{k+1}, \ldots, \lambda_d$ are fixed. To see this, rewrite Equation (4.2) as

$$\left(\lambda_k + \sum_{i=k+1}^{d} \lambda_i \mu_{ik} \right)^2 \leq \frac{R^2 - \sum_{j=k+1}^{d} \left(\lambda_j + \sum_{i=j+1}^{d} \mu_{ij} \lambda_i \right)^2 \|\mathbf{b}_j^*\|^2}{\|\mathbf{b}_k^*\|^2}.$$

Thus, for fixed $\lambda_{k+1} = \lambda_{k+1}', \ldots, \lambda_d = \lambda_d'$, λ_k must be in the interval

$$-\sum_{i=k+1}^{d} \lambda_i' \mu_{ik} - K \leq \lambda_k \leq -\sum_{i=k+1}^{d} \lambda_i' \mu_{ik} + K,$$

where

$$K = \frac{\sqrt{R^2 - \sum_{j=k+1}^{d} \left(\lambda_j' + \sum_{i=j+1}^{d} \mu_{ij} \lambda_i' \right)^2 \|\mathbf{b}_j^*\|^2}}{\|\mathbf{b}_k^*\|}.$$

Note that it is possible that the interval for λ_k is empty (or does not contain integers) for fixed $\lambda_{k+1}', \ldots, \lambda_d$. By trying all possible combinations of $\lambda_1, \ldots, \lambda_d$ that satisfy these inequalities, all lattice vectors of norm smaller than R are obtained. Thus, by keeping track of the shortest vector so far, the result will be one of the shortest vectors in the lattice.

It is perhaps simpler to view the enumeration vectors as a search through a tree where each node corresponds to some vector. The i'th level of the tree (where the 0th level is the root) consists of all vectors of $\pi_{d-i+1}(L)$, for $0 \leq i \leq d$. Let $\mathbf{v}$ be a node on the i'th level of the tree, i.e.,

$\mathbf{v} \in \pi_{d-i+1}(L)$. Then, its children consist of all vectors $\mathbf{u} \in \pi_{d-i+2}(L)$ that get projected onto $\mathbf{v}$ when applying π_{d-i+1}, i.e., $\mathbf{u} = \pi_{d-i+1}(L)$. Thus, the root of the tree consists of $\pi_{d+1}(L) = \{\mathbf{0}\}$, the zero vector. The first level of the tree consists of all vectors in $\pi_d(L) = \mathcal{L}(\mathbf{b}_d^*)$ of norm at most R, i.e., all multiples of $\mathbf{b}_d^*$ of norm at most R. The second level of the tree consists of the children of nodes on the first level. This continues until level d, which contains all vectors of $\pi_1(L) = L$ of norm at most R.

Figure 4.1 depicts a part of the first two levels of such an enumeration tree. Each block consists of a node containing a vector. The first level contains all integer multiples $\lambda_d \mathbf{b}_d^*$ such that $-R/\|\mathbf{b}_d^*\| \leq \lambda_d \leq R/\|\mathbf{b}_d^*\|$. On the second level, three children of $\mathbf{b}_d^*$ are drawn. These correspond to taking $\lambda_d = 1$ in the enumeration and then taking λ_{d-1} in the appropriate interval. If a vector $\mathbf{v} = \lambda_1 \mathbf{b}_1 + \ldots + \lambda_{d-1}\mathbf{b}_{d-1} + \lambda_d \mathbf{b}_d$, then

$$\pi_{d-1}(\mathbf{v}) = \pi_{d-1}(\lambda_d \mathbf{b}_d) + \pi_{d-1}(\lambda_{d-1}\mathbf{b}_{d-1}) = \lambda_d \mathbf{b}_d^* + (\lambda_d \mu_{d,d-1} - \lambda_{d-1})\mathbf{b}_{d-1}^*.$$

Note that this is exactly the form of the children of $\mathbf{b}_d^*$ in the figure. The other children of the node corresponding to $\mathbf{b}_d^*$ are omitted, as well as the children of the other nodes. Note that the tree is symmetric, as for each vector $\mathbf{v}$ in the tree, $-\mathbf{v}$ is in the tree as well. During the enumeration, only one side of the tree needs to be explored.

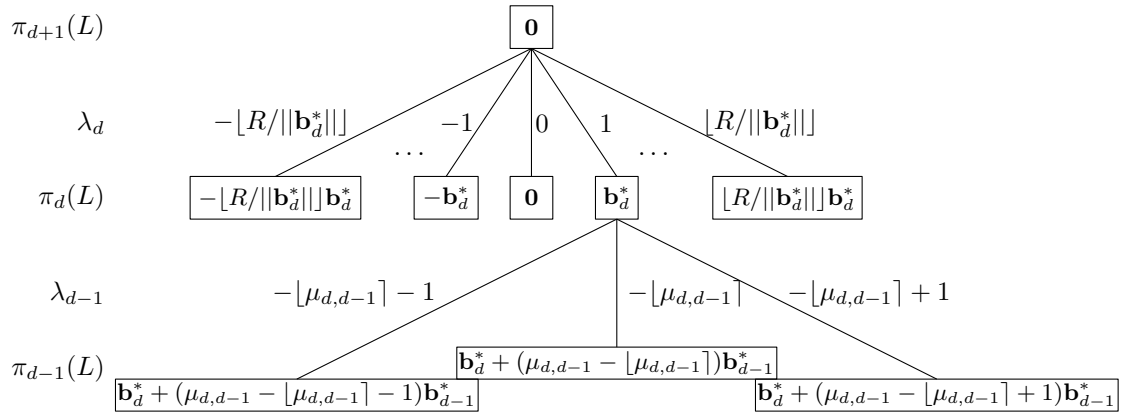

Figure 4.1: First two levels of the enumeration tree.

Such enumeration trees grow quite large. In fact, they become exponentially large, dependent on the precision of the bound R. The lower this bound, the smaller the corresponding enumeration tree. Thus, while such methods give an exact solution to the shortest vector problem, their running time is not polynomially bounded. In order to optimize the running time, lattice reduction algorithms are often used before enumerating. This improves the GSO of the basis, reduces the numbers μ_{ij} by size-reduction and additionally gives an exponential approximation to the shortest vector (which in turn gives exponential bounds for the running time of the enumeration).

4.3.1 Pruned enumeration

As mentioned, Schnorr and Euchner incorporated an enumeration method in BKZ as a subroutine. A later improvement by Schnorr and Hörner [96] uses *pruned enumeration*. Essentially, the basic enumeration algorithm by Schnorr and Euchner can be seen as a depth-first search in the aforementioned enumeration tree. In pruned enumeration, so-called branches or subtrees are excluded from the search. These subtrees are chosen in such a way that the probability that the shortest vector is inside that subtree is too small compared to some threshold. This means that with some small probability, the shortest vector may not be found. However, the vector that is found instead is still reasonably short and depending on the thresholds, the running time should decrease enough to justify this error probability.

Recently, Gama, Nguyen and Regev introduced a new concept called *extreme pruning* [20]. Their main idea is to prune a large number of branches, thus significantly reducing the search tree. As a result, the probability that the shortest vector is found becomes very low. However, the running time of the enumeration is reduced by a much bigger factor. Therefore, the pruned enumeration can be executed several times until the shortest vector is found. As the running time of a single enumeration was reduced significantly, this results in a net decrease in running time when performing several enumerations. Their method requires a very sharp (if not exact) bound on the length of the shortest vector to be known beforehand, however. Using this bound, whenever a vector on the highest level of the tree is found (i.e., in the lattice itself) it must be a shortest vector. This allows the algorithm to terminate whenever such a vector is found, as opposed to Schnorr-Euchner enumeration, where the whole tree must be searched.

Gama, Nguyen and Regev also introduce so-called *bounding functions* to improve the analysis of pruned enumeration methods, and show that the analysis of the original method by Schnorr and Hörner was not optimal. Additionally, Schnorr recently introduced new enumeration methods [93], using results of Gama, Nguyen and Regev from [20]. It would be interesting to study the possible improvements of the performance of lattice reduction algorithms, such as BKZ, when combining them with these new methods.

4.4 Performance of lattice reduction algorithms

As mentioned in Chapter 2, the behavior of lattice reduction algorithms when applied to lattices of different ranks is not well understood. Therefore, Gama and Nguyen [19] examined the performance of several reduction algorithms on a variety of lattices. Their goal was to show the possibilities of current lattice reduction algorithms, and to use these results to compare the practical difficulty of the HSVP, uSVP and approximate SVP problems, which were described in Chapter 1. This section gives a description of their methods and results.

First, their method of lattice generation is described. Then, the lattice reduction algorithms that they use in their experiments will be discussed. The quality of the resulting bases in the experiments will be considered afterwards, as well as the running time of the experiments.

4.4.1 Methodology

Generating lattices

To determine the performance of lattice reduction algorithms, lattices are required. Thus, the first issue that arises is the method of lattice generation. For examining the average case performance of the algorithms on HSVP and approximate SVP, Gama and Nguyen took a large sample of lattices from a distribution due to Goldstein and Mayer [32]. These lattices have the property that the successive minima of the lattice satisfy

$$\lambda_i(L) \approx \sigma(L) = \sqrt{\frac{n}{2\pi \mathrm{e}}}(\mathrm{vol}(L))^{1/n},$$

for all $1 \leq i \leq n$. This means that all minima are close to the expected shortest length $\sigma(L)$ as defined in Chapter 1.

One problem with this approach is that the unique shortest vector problem requires extra structure in the lattice. The lattice gap $\lambda_2(L)/\lambda_1(L)$ needs to exceed a value γ, but there is no 'standard' way to construct lattices with a prescribed gap. To solve this, Gama and Nguyen considered two classes of lattices where they could choose the approximate lengths $\lambda_1(L)$ and $\lambda_2(L)$ and then used orthogonality to ensure that the appropriate vectors have these lengths. It is not entirely clear how these choices affect the performance of lattice reduction. The lattice reduction algorithms may be able to exploit the orthogonality in order to perform better. Therefore, aside from the two aforementioned classes, they also perform experiments on lattices that arise from knapsack problems (see Section 1.7). However, there is no known formula for the second minimum

λ_2 of such lattices. As a result, the second minimum needs to be approximated heuristically in order to prescribe the gap.

Once these random lattices are generated, they need to be represented by means of a basis. A given basis might have special properties that influence the performance of a lattice reduction algorithm. To remove such influences, Gama and Nguyen want to work with random bases. However, there is no standard notion of random bases for a given lattice. Gama and Nguyen define a random basis as a basis consisting of relatively large lattice vectors that are chosen using a randomized heuristic. They do not explicitly give their methods to randomly generate lattice bases, but they refer to the description of the GGH-cryptosystem [30], which details heuristics to randomize a lattice basis. In the experiments, they performed the lattice reduction on at least twenty randomized bases for each lattice. This was done to prevent the reduction algorithms from taking advantage of special properties of the bases.

Algorithms

Lattice reduction algorithms give an approximate solution to the different variants of the shortest vector problem. They produce bases $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ that contain a short vector $\mathbf{b}_1$. To measure the quality of this short vector, the *approximation factor* is defined as $\|\mathbf{b}_1\|/\lambda_1(L)$ and the *Hermite factor* is defined as $\|\mathbf{b}_1\|/\operatorname{vol}(L)^{1/d}$, where d is the lattice rank. These factors try to capture how "close" $\mathbf{b}_1$ is to the shortest vector.

As mentioned in Section 4.3, there also exist algorithms that solve SVP exactly. The algorithms mentioned in that section perform an exhaustive search based on enumeration methods, which has a complexity that is at least exponential in the lattice rank. Gama and Nguyen pose that these algorithms cannot be run on lattices where the rank is greater than 100. For such ranks, only approximation algorithms such as basis reduction algorithms are practical. However, this observation was made before the introduction of extreme pruning by Gama, Nguyen and Regev in [20]. Extreme pruning has been used to find the shortest vector in lattices of rank 110 (with a running time of 62.12 CPU days).

For their experiments, Gama and Nguyen use three different reduction algorithms: LLL [54], DEEP [95] and BKZ [95]. The LLL-algorithm has been described in Chapter 1, the DEEP-algorithm was described in Section 4.2.2 and the BKZ-algorithm was described in Section 4.2.3.

Gama and Nguyen used the NTL [99] (version 5.4.1) implementations of BKZ and DEEP in their experiments. In NTL, both BKZ and DEEP use a 'blocksize' parameter β. For higher values of β, the quality of the reduction increases, but the running time increases as well. In addition to the quality of the reduced bases, Gama and Nguyen examined the running times of BKZ and DEEP in their experiments.

4.4.2 Results

The results of the experiments by Gama and Nguyen will be discussed here. In the cases of HSVP and approximate SVP, the performance of the lattice reduction algorithms is respectively measured by the Hermite and approximation factors of the resulting vector. In the case of uSVP, the only measure of performance is whether the shortest vector is retrieved or not.

For each of the three lattice problems, the theoretical expectations of the performance of the lattice reduction algorithms will be examined first. Then, these expectations will be compared to the experimental results and an attempt will be made to explain the difference between theory and practice. Finally, the experimental running time of BKZ and DEEP will be considered, as well as the running time of an exhaustive enumeration method.

HSVP

Recall the definition of Hermite SVP from Section 1.6. The goal is to find a vector of norm at most $\gamma \operatorname{vol}(L)^{1/d}$ in a d-rank lattice L for some approximation factor γ. Theoretically, lattice reduction algorithms solve HSVP with a Hermite factor $\|\mathbf{b}_1\|/\operatorname{vol}^{1/d} = (1+\varepsilon)^d$, where ε depends

on the algorithm and its parameters. For LLL with appropriate reduction parameters, the Hermite factor is theoretically provable to be $\lessapprox (\gamma_2)^{(n-1)/2} = (4/3)^{(n-1)/4} \approx 1.0746$, where γ_2 is Hermite's constant for rank 2 lattices. Note that this bound corresponds to Theorem 1.45(i).

The DEEP algorithm is based on the LLL algorithm and theoretically it is not known to perform better than LLL. In other words, there is no upper bound known for the Hermite factor of DEEP, except the upper bound of LLL. However, DEEP is expected to perform better than LLL in practice. BKZ does have theoretical upper bounds. By using arguments similar to those in [91], it can be proven that the Hermite factor of BKZ is $\sqrt{\gamma_\beta}^{1+(d-1)/(\beta-1)}$, where β is the blocksize parameter. For $\beta = 20$ the Hermite factor is approximately 1.0337^d and for $\beta = 28$ the Hermite factor is approximately 1.0282^d. Are these theoretical bounds tight in practice?

The first observation that Gama and Nguyen make from their experiments is that the Hermite factor of the result of lattice reduction does not seem to depend on the lattice. Only when the lattice has exceptional structure will the Hermite factor be relatively small compared to the general case. Here, exceptional structure means that either $\lambda_1(L)$ is very small, or the lattice contains a sublattice (of lower rank) of very small volume, i.e., a sublattice spanned by a few relatively short vectors.

The next observation is that, when there is no such exceptional structure in the lattice, the Hermite factor appears to be exponential in the lattice rank. This agrees with the theoretical predictions. However, the specific constants that are involved appear to differ in practice. The experiments show that the Hermite factor is approximately of the form e^{ad+b}, where d is the lattice rank and the constants a and b depend only on the reduction algorithm. Gama and Nguyen are only interested in rough estimations and they simplify e^{ad+b} to δ^d. Table 4.1 shows the base δ of the average Hermite factors that were derived from the experiments.

Algorithm-β	LLL	DEEP-50	BKZ-20	BKZ-28
Experimental δ = Hermite factor$^{1/d}$	1.0219	1.011	1.0128	1.0109
Theoretical proven upper bound	1.0746	1.0746	1.0337	1.0282

Table 4.1: Experimental Hermite factor compared to theoretical upper bounds.

Table 4.1 shows that DEEP and BKZ perform roughly the same as LLL, except that their constants are smaller. The constants for BKZ and DEEP are roughly the square root of the constants for LLL. To give a concrete example what these constants mean in practice, consider lattices of rank $d = 300$. According to these experimental constants, LLL will obtain a vector with Hermite factor of approximately $1.0219^{300} \approx 665$, while the theoretical upper bound is $1.0746^{300} \approx 2958078142$. Furthermore, BKZ-20 will obtain a vector with a Hermite factor of approximately $1.0128^{300} \approx 45$, while the theoretical upper bound is $1.0337^{300} \approx 20814$.

The results of LLL are quite interesting. It is known that in the worst-case, the Hermite factor of LLL is equal to the theoretical upper bound $\gamma_2^{(n-1)/2} = (4/3)^{(n-1)/4}$. This occurs when the input is a lattice basis such that all its 2-rank projected lattices are critical, i.e., they reach equality in Equation (1.8). However, this behavior is caused by a worst-case basis and not by a worst-case lattice. The experiments showed that when a random basis of these lattices was reduced instead of this worst-case basis, the resulting Hermite factor was again 1.0219^d.

It is harder to explain the gap between theory and practice for the BKZ algorithm. The BKZ algorithm uses projected lattices of rank β, where β is the blocksize. Although it is known that these projected lattices do not have the same distribution as random lattices of rank β, no good model for their distribution is known. This makes it difficult to analyze the performance of BKZ theoretically.

Based on their results, Gama and Nguyen conclude that the best algorithms can reach a Hermite factor of roughly 1.01^d. They also conclude that solving HSVP for Hermite factor d using BKZ is currently 'easy' for $d \leq 450$, as δ^d is approximately linear in this case, e.g., $1.013^{450} \approx 334 \leq 450$. However, they note that a Hermite factor of 1.005^d cannot be reached for rank $d = 500$ in practice, unless the lattice has an exceptional structure as discussed before.

Approximate SVP

Recall from 3.1.6 that any algorithm that solves HSVP with Hermite factor γ can be used to solve approximate SVP with approximation factor γ^2. This leads to the expectation that the reduction algorithms can solve approximate SVP with an approximation factor that is equal to the square of the Hermite factor. Since the experimental results for HSVP show that the Hermite factor is approximately δ^d with δ as in Table 4.1, it is to be expected that the approximation factor is roughly $(\delta^d)^2 = (\delta^2)^d$. Assuming that the best reduction algorithms can reach a Hermite factor of roughly 1.01^d, it is expected that they will reach an approximation factor of roughly $1.01^{2d} \approx 1.02^d$.

The experiments showed that this expectation was true in the worst-case. Gama and Nguyen constructed lattices where the approximation factor was the square of the Hermite factor. However, on the average it appeared that the approximation factor was in fact roughly the same as the Hermite factor 1.01^d, rather than its square $1.01^{2d} \approx 1.02^d$. A possible explanation is that in lattices where $\lambda_1(L) \geq \text{vol}(L)^{1/d}$, any algorithm that reaches a Hermite factor $\|\mathbf{b}_1\|/\text{vol}(L)^{1/d}$ will reach an approximation factor $\|\mathbf{b}_1(L)\|/\lambda_1(L) \leq \|\mathbf{b}_1(L)\|/\text{vol}(L)^{1/d}$. Thus, approximate SVP can only be harder than HSVP in lattices where $\lambda_1(L) \leq \text{vol}(L)^{1/d}$. However, if $\lambda_1(L)$ becomes too small compared to $\text{vol}(L)^{1/d}$, the lattice will have an exceptional structure. This structure can then be exploited by lattice reduction algorithms in order to improve their results.

The worst-case results are based on experiments with LLL, but Gama and Nguyen note that BKZ and DEEP perform essentially the same except for the better constants. They conclude that current algorithms can reach approximation factors of 1.01^d on the average and 1.02^d in the worst case. This suggests that solving approximate SVP with approximation factor d is easy on the average for $d \leq 500$, because 1.01^d is approximately linear in d for these values of d.

uSVP

Recall the definition of uSVP from Section 1.6. The goal is to find the shortest vector in a lattice L where the shortest vector $\mathbf{u}$ is unique. Here, unique means that all vectors of length $\leq \gamma\|\mathbf{u}\|$ are a multiple of $\mathbf{u}$ for some gap constant γ. Recall from Section 3.1.6 that any algorithm that achieves an approximation factor $\leq \gamma$ can solve the unique shortest vector problem with gap γ. Thus, using the results of the approximate shortest vector problem, the expectation is that uSVP can be solved for gap roughly $\geq 1.02^d$, the square of the Hermite factor.

As mentioned in the part about lattice generation, Gama and Nguyen constructed two classes of lattices where they could choose the lattice gap. For these lattices they found that LLL would retrieve the unique shortest vector whenever the gap was exponential in the lattice rank, as predicted. However, the gap did not need to be the square of the Hermite factor, which is approximately 1.042^d for LLL. Instead, LLL obtains the unique shortest vector with high probability as soon as the gap becomes a fraction of the Hermite factor (and not its square). For the first class this happened whenever $\lambda_2/\lambda_1 \geq 0.26 \cdot 1.021^d$ and for the second class whenever $\lambda_2/\lambda_1 \geq 0.45 \cdot 1.021^d$. For BKZ, the constants were so close to 1 that lattices of rank < 200 did not provide sufficient accuracy on the constants. The results from higher ranks seemed to indicate that BKZ could find the unique shortest vector whenever the gap was greater than $0.18 \cdot 1.012^d$ in the first class.

However, these constructions had an exceptional structure when compared to general uSVP-instances, which could affect the performance of reduction algorithms. Thus, Gama and Nguyen repeated their experiments for so-called Lagarias-Odlyzko lattices [51], which are lattices that arise from knapsack problems. Using heuristical estimates to determine the gap, they found that LLL could retrieve the unique shortest vector whenever the gap was greater than $0.25 \cdot 1.021^d$ and BKZ achieved this whenever the gap was greater than $0.48 \cdot 1.012^d$. This confirmed their earlier result that uSVP is easy to solve whenever the gap is a fraction of the Hermite factor, rather than its square.

Experimental running time

As mentioned in their description, no tight bounds are known for the BKZ and DEEP algorithms. Furthermore, while exhaustive enumeration methods are not practical in higher ranks, they are used as a subroutine in BKZ to search for short vectors in blocks. Therefore, Gama and Nguyen examined the experimental running time of such methods as well.

For their experiments on exhaustive enumeration, Gama and Nguyen used a method due to Schnorr and Euchner [95]. This method is used as a subroutine in BKZ and it seems to outperform other theoretical algorithms in practice. On input of a lattice basis, the algorithm finds a shortest vector. The enumeration becomes faster as the input basis is more reduced. From their experiments, Gama and Nguyen note that SVP can be solved within an hour for rank $d = 60$, whereas the curve of their results shows that solving it for rank $d = 100$ would take at least 35000 years. This can be improved by better preprocessing such as basis reduction, but still Gama and Nguyen think that enumeration is not possible for lattices of rank $d \geq 100$.

The best known upper bound for BKZ is superexponential, while BKZ with blocksize $\beta = 20$ can reduce the basis of a lattice of rank $d = 100$ in a few seconds. This suggests that the superexponential bound is not tight. Thus, Gama and Nguyen measured the running time of BKZ in their experiments for several blocksizes and on lattices of varying rank. They observed that the running time increased with the blocksize exponentially, as expected. However, for blocksizes $20 \leq \beta \leq 25$, the running time started to increase disproportionately. The slope of the running time suddenly increased as seen on a logarithmic scale. This effect increased further for lattices of higher rank. It follows that BKZ with blocksize $\beta > 25$ is infeasible for lattices with high rank.

The running time of the DEEP algorithm seems to increase more regularly for increasing blocksize. It increases exponentially in the blocksize, just like the running time of BKZ. As opposed to BKZ, there is no sharp increase for higher blocksizes. This allows for DEEP to be run on high-ranked lattices with relatively high blocksizes. However, the experimental results on the quality of the bases showed that even with higher blocksizes, the Hermite factor achieved by DEEP is not expected to improve significantly beyond 1.01^d.

Results for cryptography

What do these different results mean for cryptography? The experiments show that the Hermite and approximation factor that can be reached by lattice reduction algorithms are exponential in the dimension, as was expected from the theory. However, the base of this exponential is much lower in practice than the theory would predict. This means that, although approximating these problems is still hard asymptotically, i.e., for sufficiently large lattice rank, the lattice rank needs to be at least 500 before this hardness emerges. For instance, from the results of these experiments it follows that approximating either of these problems within a factor $\gamma = d$ is easy for $d \leq 450$, because the Hermite factor δ^d reached in practice is approximately linear in d. Furthermore, the experiments show that uSVP is easy to solve whenever the gap λ_2/λ_1 is a fraction of the Hermite factor. However, by focusing on the base δ of the Hermite factor δ^d and considering what is feasible and what is not, some information is lost. The parameter δ^d predicts the quality of the resulting vectors in terms of the lattice rank d, but it does not say anything about the effort in relation to the rank of the lattice.

It should be noted that these results apply to the specific distribution of lattices described in [32]. Unless the lattices that arise from cryptographic situations come from this distribution, some additional experiments are required to determine the actual performance of lattice reduction algorithms on these 'cryptographic' lattices. Rückert and Schneider performed similar experiments for lattices that come from cryptographic examples in [89]. These results will be discussed in Section 4.5.

Another point of interest is that these conclusions change as time goes on. As computing power increases, it becomes practical to break lattices of higher rank. This should be taken into account when determining the performance of lattice reduction algorithms, and especially in the context of lattice-based cryptography. Finally, improvements in the area of lattice reduction will

improve the performance of lattice reduction algorithms. As with computing power, this affects the security of all lattice-based cryptosystems.

While Gama and Nguyen have given much insight into the practical behavior of lattice reduction algorithms, there is still work to be done. The biggest downside to the method of Gama and Nguyen is that they only distinguish between lattice problems that are 'within reach' and those that are 'not within reach' given the current lattice reduction algorithms and computing power. They do not provide a method to measure the actual cost or required effort of these algorithms, nor a way to predict how hard the problems will be in the future. In the next section, a framework that attempts to solve these problems for lattice-based cryptosystems will be discussed.

4.5 Bit-security for lattice-based cryptosystems

Most of the results in the area of lattice-based cryptography are of a theoretical nature. These results do not always give an accurate picture of the practical aspects of the cryptosystems. As seen in Chapter 2, there are several issues with (purely) theoretical results. First of all, the security proofs that accompany cryptosystems are asymptotic statements, i.e., they describe the behavior of the cryptosystem for large values of the involved parameters. Furthermore, the hardness of breaking the cryptosystems can be controlled by several interdependent variables, which increases the difficulty of deriving the practical security from the parameters. Thus, it is interesting to consider how to determine the practical security of lattice-based cryptosystems.

Rückert and Schneider created a framework to determine the practical security of cryptosystems based on the SIS and LWE problems from their parameters in [89]. The framework allows them to relate the security to bit security, just like the standard expression of the security of symmetric algorithms. They also use the framework to derive recommended parameters for these systems, using results on the performance of lattice-reduction algorithms to gauge the possibilities of attackers. Their framework will be discussed here and then the recommended parameters will be considered.

4.5.1 Framework

The idea of a unified framework to consider the security of all lattice-based cryptosystems is not entirely new. It was inspired by the works of Lenstra and Verheul [55] and the subsequent update by Lenstra [53]. The framework of Rückert and Schneider is explained in three steps. First, they show how to represent the hardness of the SIS and LWE problems with a single parameter. Then, they perform experiments to relate this parameter to the attack effort. Finally, they apply their framework to several cryptographic schemes to measure and compare their security.

Before the framework is explained, the notion of *attack effort* will be examined. Afterwards, the three steps of the framework will be explained in more detail.

Measuring security

When measuring practical security, it is useful to take the capabilities of the attacker into account. Furthermore, advances in both computing power and cryptanalytic methods will increase these capabilities in the future. Therefore, Rückert and Schneider model the attacker in their framework as well. In order to measure the attack effort, they use the notion of dollar-days, which was introduced by Lenstra in [53]. Dollar days are the cost of equipment in dollars multiplied by the time spent on the attack measured in days. Thus, using a $1000 computer and spending 4 days to break a system costs as much in terms of dollar days as using a $2000 computer and spending 2 days.

Consider a cryptographic system with security parameter k. Assume that the best known attack against this system requires $t(k)$ seconds on a computer that costs d dollars. The cost of this attack, as represented in dollar days, is given by

$$T(k) = d \cdot t(k)/(3600 \cdot 24).$$

If (an estimation of) the function $T(k)$ is known, recommended parameters can be chosen as follows. Assume an attacker has T_{y_0} dollar days at his disposal, where y_0 stands for a certain year. To be secure against this particular attacker, the security parameter of the system must exceed a value k^* such that $T(k^*) \geq T_{y_0}$.

It is also possible to consider future developments and estimate what the security of the system will be in the future. To this end, Rückert and Schneider consider a rule that Lenstra calls the "double Moore law". Moore's law states that the computing power doubles every 18 months. The double Moore law also takes advances in the field of cryptanalysis into account. Each year, the security is expected to degrade by a factor of $2^{-12/9}$. However, this function is based on algorithmic progress in the area of integer factorization. Rückert and Schneider adopt it because they find the algorithmic progress of lattice basis reduction hard to judge. To be secure up until year y against an attacker that has T_{y_0} dollar days at his disposal in year y_0, the security parameter must satisfy $T(k) \geq T_{y_0} \cdot 2^{(y-y_0)\cdot 12/9}$.

Some cryptographic schemes also use symmetric cryptographic primitives. Rückert and Schneider assume that these primitives are always available and that they are only affected by Moore's law. Their reason is that symmetric primitives can be replaced more easily when attacks are found than asymmetric ones.

Measuring the hardness of SIS and LWE

Both the SIS and LWE problems have several parameters that can influence their hardness. However, it is desirable to represent the security of cryptographic systems with a single parameter. First, Rückert and Schneider analyze the hardness of the SIS problem, introducing a parameter that corresponds to the best known attack. Then, they provide some experimental data to show that this parameter influences the result the most. Finally, they show how to reduce LWE to SIS, which allows them to reuse the hardness results of SIS for LWE.

Recall from the description of SIS in Section 3.1 that it has the four parameters n, m, q and ν and that the corresponding lattice is of the form

$$\Lambda_q^\perp(A) = \{\mathbf{x} \in \mathbb{Z}^m : A\mathbf{x} = 0 \pmod q\},$$

where A is an $n \times m$ matrix with entries in Z_q. If the rows of A are linearly independent, then $\Lambda_q^\perp(A)$ is a full-rank lattice in $\mathbb{Z}^m$. Furthermore, it contains q^{m-n} lattice vectors that are in $\mathbb{Z}_q^m$ and hence its volume is q^n.

Practical hardness is measured by the resistance against attacks. Thus, in order to measure the practical hardness of SIS, the best way to attack it must be determined. At this time, no methods are known to perform better than lattice reduction algorithms. Recall that lattice reduction algorithms are δ-HSVP solvers that find a vector of norm at most $\delta^d \operatorname{vol}(L)^{1/d}$ in a lattice L, where d is the lattice rank.

The system $A\mathbf{x} = 0$ is underdetermined. Therefore, it is possible to fix k coordinates of $\mathbf{x}$ to zero, and attempt to solve the problem using the other $m - k$ coordinates. This results in a sublattice of $\Lambda_q^\perp(A)$ of lower rank, which still has volume q^n with high probability. Using this observation, Micciancio and Regev [71] introduced a sub-lattice attack on the SIS problem. Instead of trying to solve SIS in the lattice $\Lambda_q^\perp(A)$, they propose to solve it in the lattice $\Lambda_q^\perp(A')$, where A' is obtained by removing some columns from A. As the resulting lattice is contained in a space of lower dimension, the obtained vectors should be padded with zeroes where the columns of A were removed. Let A' have $m - k = d$ columns, which means the lattice $\Lambda_q^\perp(A')$ has rank d. Applying a lattice reduction algorithm to the sublattice $\Lambda_q^\perp(A')$ gives a vector of norm at most $\delta^d \operatorname{vol}(\Lambda_q^\perp(A')) = \delta^d q^{n/d}$. Micciancio and Regev showed that the minimum of this function is obtained for $d = \sqrt{n \log_2 q / \log_2 \delta}$. For this d, δ satisfies the equation

$$\delta = 2^{n \log_2 q / d^2}$$

Using a sufficiently strong HSVP solver will result in a vector that has norm at most $\delta^d q^{n/d} = q^{2n/d}$.

Rückert and Schneider note that the above analysis does not include the parameter ν of the SIS problem. Micciancio and Regev assume that the δ of the HSVP solver is fixed, but an attacker might employ lattice reduction algorithms of varying strength, until a suitably short vector is found. Therefore, they propose to take ν into account when determining the strongest attack. This results in the following proposition:

Proposition 4.2. *Let $n \geq 128$, $q \geq n^2$ and $\nu < q$. The optimal lattice rank for solving SIS(n, m, q, ν) with a δ-HSVP solver for variable δ is $d = \min\left\{x \in \mathbb{N} : q^{2n/x} \leq \nu\right\}$.*

In their proof, Rückert and Schneider show that the solver must be able to solve δ-HSVP for $\delta \leq \sqrt[d]{\nu/q^{n/d}}$. They also prove that the minimum attack rank d must be at least $2n = 256$, since if $d \leq 2n$ then $q \leq q^{2n/d} \leq \nu < q$, which gives a contradiction. They sum up the basis of their analysis in the following conjecture:

Conjecture 4.3. *Let $n > 128$, a constant $c \geq 2$, a prime $q \geq n^c$, $m = \Omega(n \log_2(q))$ and $\nu < q$ be given. Then, the best known approach to solve SIS(n, m, q, ν) is to solve δ-HSVP in rank $d = \min\left\{x \in \mathbb{N} : q^{2n/x} \leq \nu\right\}$ with $\delta = \sqrt[d]{\nu/q^{n/d}}$.*

This conjecture assumes that the best possible attack on the SIS-problem consists of solving the Hermite shortest vector problem in a suitable lattice. Now, Rückert and Schneider posit that the most natural approach to solve the decision version of the LWE-problem is by solving an instance of the SIS problem. By reducing LWE to SIS, they can use hardness estimates for SIS to provide hardness estimates for LWE. This reduction was mentioned in Section 3.1.6 and is formalized in the following proposition:

Proposition 4.4. *Any algorithm that solves SIS with the parameters $(n, q, m, \nu = 1.5\sqrt{2\pi}/\alpha)$ can be used to solve LWE with parameters (n, q, m, α).*

Next, Rückert and Schneider performed experiments to see the influence of the different parameters on the hardness of the SIS problem.

Experimental data

In the experiments, Rückert and Schneider apply BKZ (as implemented in NTL [99]) to sublattices of the optimal rank d (as defined by Proposition 4.2). They gradually increase the blocksize β of BKZ – thus decreasing δ – until a vector of the desired length is found. Then, they measure the running time of the reduction algorithm and compare these for varying n, m and q.

Their first observation is that for $\delta \in (1, 1.02]$, q has but a relatively minor impact on the running time of the reduction algorithm. Secondly, they note that the lattice rank m influences the running time more noticeably. However, they claim that the most influential parameter is δ. For $\delta < 1.015$, the running time increases rapidly as δ decreases. Thus, they conclude that δ should be considered the main security parameter.

Finally, in order to obtain the security estimates, Rückert and Schneider fix $m = 175$, $n > 128$ and $q \approx n^3$. With δ as the main security parameter, they consider the cost function to be $T(\delta) = a2^{1/(\log_2(\delta)^b)} + c$ dollar days, where a, b and c are constants. Next, they use the experimental data to approximate the constants a, b and c, resulting in the values $a \approx 10^{-15}$, $b \approx 1.001$ and $c = 0.005$. They consider c to be negligible for small δ, which leads to the following conjecture:

Conjecture 4.5. *Let $n > 128$, a constant $c \geq 2$, a prime $q \geq n^c$, $m = \Omega(n \log_2(q))$ and $\nu < q$ be given. Then, for any $\delta \in (1, 1.015]$, solving δ-HSVP in (normalized) q-ary lattices of rank d costs at least $T(\delta) = 10^{-15}2^{1/(\log_2(\delta)^{1.001})}$ dollar days.*

Using this cost function, Rückert and Schneider predict parameters δ for HSVP that are infeasible for certain attackers. They distinguish between the three classes of attackers, inspired by works of Blaze et al. ([8]) and Lenstra ([53]). These classes are Hacker, Lenstra and Intelligence agency, each having a different budget. The Hacker has access to 400 dollar days, the attacker Lenstra possesses 40 million dollar days and the Intelligence agency has 108 billion dollar days at its disposal. The infeasible values for δ that they computed using the cost function are shown

in Table 4.2. The table also includes values for the corresponding bit security. The derivation of the bit security follows from the work of Lenstra and Verheul. It is computed by the formula $\lceil 56 + 12(y - 1982)/18 \rceil$, where y is the year. The significance of this formula is that 56 is the bit security of DES, which was considered to be secure until the year 1982, "even against the strongest attacker". The factor 12/18 follows from the simple Moore law, and thus the formula is based on the assumption that DES was secure in 1982 and that since then, attackers have become able to break $12(y - 1982)/18$ more bits of security.

Year	2010	2020	2030	2040	2050	2060	2070	2080	2090	2100
Bit security	75	82	88	95	102	108	115	122	128	135
Hacker	1.01177	1.00965	1.00808	1.00702	1.00621	1.00552	1.00501	1.00458	1.00419	1.00389
Lenstra	1.00919	1.00785	1.00678	1.00602	1.00541	1.00488	1.00447	1.00413	1.00381	1.00356
Int. Agency	1.00799	1.00695	1.00610	1.00548	1.00497	1.00452	1.00417	1.00387	1.00359	1.00336

Table 4.2: Values of δ predicted to be infeasible to break for the attackers.

Sadly, this does not give a direct *relation* between bit security and the feasibility of lattice problems. It merely lists values of δ such that breaking some cryptosystem is infeasible for a given attacker in a given year next to the number of bits such that breaking a symmetric algorithm with this key length is infeasible for all attackers in a given year. It would be interesting to consider a more direct relation between the effort required to achieve such δ and the effort required to break a system with such a key length.

Furthermore, the decision to consider only δ for the effort function T, while ignoring parameters such as the dimension n and the lattice rank m seems questionable. Even if the effect of δ on the effort is much more noticeable than the effect of other parameters such as m and n, it is still interesting to consider the effects of the parameters δ, m and n on the efficiency of the cryptosystem. It might be that changing the δ is much more costly than changing the m or n parameters. In any case, it seems prudent to keep the effort function more general, and include other parameters as well. However, it should be noted that this will increase the complexity of the model.

4.5.2 Recommended parameters

Here, the framework will be applied to several cryptosystems that were described in Section 3.3. For each application, Rückert and Schneider choose different parameters and apply their framework to see whether the resulting cryptosystems are resilient against the attacker 'Lenstra' (40 million dollar days).

Encryption with LWE

Recall the encryption scheme based on LWE as described in Section 3.3.1. Rather than applying their framework to this system, Rückert and Schneider consider a multi-bit variant that encrypts κ bits at once. As with the single-bit variant, the other parameters consist of the dimensions n and m, modulus q and error parameter α.

Year	2010	2020	2030	2040	2050	2060
Bit security	150	164	176	190	204	216
n	191	221	253	283	314	346
q	72973	97687	128021	160183	197203	239441
α	$5.51 \cdot 10^{-4}$	$5.12 \cdot 10^{-4}$	$4.80 \cdot 10^{-4}$	$4.54 \cdot 10^{-4}$	$4.30 \cdot 10^{-4}$	$4.10 \cdot 10^{-4}$
m	3665	4234	4815	5400	6006	6609

Table 4.3: Recommended parameters for multi-bit LWE against attacker 'Lenstra'.

Rückert and Schneider choose $\alpha = (30\sqrt{m})^{-1}$ to restrict the probability of decryption errors to negligible levels. Furthermore, they set the modulus $q = q(n)$ as the smallest prime between $2n^2$ and $4n^2$. Finally, they choose $m = \lceil (n + \kappa) \log_2(q) + 2\kappa \rceil$, to ensure that ciphertexts are close to uniform. They note that there exist some trade-offs, such as choosing a different alphabet,

or increasing the randomness of the blinding factor of the ciphertexts. However, they choose to consider only the basic examples for simplicity. The resulting recommended parameters are shown in Table 4.3.

Encryption with Dual LWE

Consider the 'dual' encryption scheme based on LWE from Section 3.3.3, as used by Gentry, Peikert and Vaikuntanathan [26] for the creation of an identity-based encryption scheme. Rückert and Schneider apply their framework to a variant of the scheme from another article by Peikert [80]. This variant encrypts κ bits at once. As with the original system, it is parameterized by dimensions n and m, modulus q and error parameter α.

Year	2010	2020	2030	2040	2050	2060
Bit security	150	164	176	190	204	216
n	190	220	253	284	314	347
q	72211	96821	128021	161323	197203	240829
α	$5.65 \cdot 10^{-4}$	$5.21 \cdot 10^{-4}$	$4.82 \cdot 10^{-4}$	$4.52 \cdot 10^{-4}$	$4.27 \cdot 10^{-4}$	$4.04 \cdot 10^{-4}$
m	3367	3972	4645	5294	5932	6636

Table 4.4: Recommended parameters for multi-bit Dual LWE against attacker 'Lenstra'.

It should be noted that Rückert and Schneider choose not to use exponential moduli q, as used in the hardness reduction without a quantum computer (see Section 3.1.6). They justify this choice by noting that the key size would grow too much, as well as lead to a slightly more complicated encryption process. Instead, they choose $q = q(n)$ as the smallest prime between $2n^2$ and $4n^2$. They also observe that it is possible to choose the secret key from a somewhat larger set, which would increase its size, but decrease the size of the ciphertexts. Furthermore, they choose $m = \lceil (n \log_2(q) + 2\kappa \rceil$, to ensure that the public key is close to uniform and they choose $\alpha = (30(\sqrt{m} + 1))^{-1}$ to ensure that the errors do not become too big. This leads to several sets of recommended parameters, which are shown in Table 4.4.

Digital Signatures

Recall the digital signature scheme from Section 3.3.5 by Lyubashevsky [60]. As it is based on SIS, it is not necessary to convert the problem to SIS before applying the framework. Furthermore, Lyubashevsky proposed his own recommended parameters, so Rückert and Schneider can apply their framework to analyze these.

n	512	512	512	1024
q	$2^{31.727}$	$2^{59.748}$	$2^{95.747}$	$2^{95.872}$
ν	$5.6 \cdot 10^8$	$1.3 \cdot 10^{10}$	$2.6 \cdot 10^{10}$	$6.4 \cdot 10^{10}$
d	1118	1823	2835	5471
δ	1.0091	1.0064	1.0042	1.0023
Year	2010	2035	2077	2180
Bit security	75	92	120	188

Table 4.5: Analysis of the recommended parameters for Lyubashevsky's digital signature scheme.

The system is parameterized by dimensions n and m and modulus q, as well as some other parameters that were omitted in the system description for simplicity. Rückert and Schneider compute the corresponding norm bound ν from the dimensions and the norm bounds for signing keys and hashed message lengths. For each parameter set, they give an optimal attack dimension d, as well as the corresponding δ. Finally, they use their earlier results on the relation between δ and bit-security to provide the bit-security, as well as the year when attacker 'Lenstra' is about to break the system. These parameters are shown in Table 4.5. Rückert and Schneider conclude that, although the underlying hash-functions may be broken by the year 2018, the lattice parameters are quite reasonable.

4.6 Conclusions

In this chapter, several aspects of lattice basis reduction were considered. Basis reduction is one of the most prominent tools when attacking lattice problems and, by extension, lattice-based cryptosystems. Therefore, new developments in the area of lattice reduction were considered. Furthermore, lattice reduction is often used in conjunction with enumeration methods, in order to improve the quality of the reduction at the cost of a larger running time. Conversely, these enumeration methods, which solve SVP exactly, often rely on lattice reduction algorithms to improve their running time. New developments in the area of enumeration were explored as well.

Additionally, the practical implications of lattice reduction were considered. The performance of lattice reduction algorithms is much better than previous theoretical results indicate. This gives a reasonable explanation for why early lattice-based cryptosystems were broken by attacks based on lattice reduction. Furthermore, this performance can be used to give a measure of security of lattice-based cryptosystems.

The parameter δ, which stands for the Hermite factor, has been used to measure the performance of lattice reduction algorithms. This δ gives a relation between the lattice rank and the length of the short vector that is found by lattice reduction. The expected amount of effort that must be invested to achieve such a δ is based upon the extrapolation of experiments in lattices of a rank where this is still feasible. It would be interesting to examine the relation between δ and this effort more closely, for varying lattice rank.

While much progress has been made on understanding the practical side of lattice reduction, there is still some work to be done. The work by Gama and Nguyen does not provide methods to estimate the cost of lattice reduction of varying strength in varying dimensions. Such methods could be used to predict the hardness of problems in the future, when more computing power becomes available. The work by Rückert and Schneider attempts to solve this by defining the 'effort' using a cost function depending on δ, but it disregards the relation to other parameters, such as the dimension and lattice rank. Some thoughts on these problems will be discussed in the next chapter.

Chapter 5

Measuring the work factor of SVP

5.1 Introduction

As mentioned in the previous chapters, it is important to understand the performance of lattice reduction algorithms. In Sections 4.4 and 4.5, two different approaches to this problem have been described. These approaches each have their own drawbacks. In this chapter, a different approach will be suggested and discussed.

Section 5.2 contains the motivation for this new approach. In Section 5.3, the measurements of the experiment will be considered. The setup of the experiments is described in Section 5.4. Finally, the results will be discussed in Section 5.5 and possibilities for the future will be discussed in Section 5.6.

5.2 Motivation

If lattice-based cryptography is to be used in practical applications, there needs to be a way to determine the security of lattice-based cryptosystems. In practice, the security of a cryptosystem is determined by the strength of the best attack that is known, i.e., the amount of 'effort' that is needed to break the system using this best attack. For lattice-based cryptography, the best attacks currently consist of lattice basis reduction (sometimes combined with enumeration). Thus, it is vital to understand the practical performance of basis reduction algorithms in order to understand the security of lattice-based cryptosystems.

Two approaches to this problem have been described in the previous chapter. The first approach, due to Gama and Nguyen in [19], considers basis reduction algorithms in a general, non-cryptographic sense. It focuses on the Hermite factor δ^d reached by algorithms such as LLL and BKZ. They observe that each algorithm corresponds to some base δ, such that the algorithm returns a vector of length approximately $\delta^d \operatorname{vol}(L)^{1/d}$ when applied to a lattice L of rank d. However, their results are aimed at which δ's are feasible and which are not, rather than the relation between δ and the effort or time spent. The second approach, due to Rückert and Schneider in [89], is aimed at basis reduction in the cryptographic setting, in order to determine the security of lattice-based cryptosystems. It builds on the ideas of Gama and Nguyen and focuses on the effort required to reach Hermite factors with base δ by lattice reduction algorithms. However, they do not incorporate the lattice rank d, nor the space dimension n in this relation, as they deem it to be less influential than the base δ.

Here, a different approach is proposed. The goal of this approach is to examine the relation between the rank of a lattice, the quality (or "shortness") of the shortest vector found when applying a basis reduction algorithm and the effort required to reach this solution. Hopefully, this leads to a better understanding of the interaction between the different parameters. Furthermore, once this relation is better understood, it becomes possible to fine-tune the parameters of a cryptosystem to reach certain security levels. For example, the creator of such a cryptosystem could first compute

the quality of vectors that can be used to break the system. Then, he chooses a threshold for the effort needed to break the system. Finally, he uses the relation to find a dimension n and possibly a rank d such that no adversary is able to break the system by putting in less effort than the chosen threshold using the best (known) attack.

5.3 Measurements

To reach a better level of understanding about this relation, experiments with lattice basis reduction algorithms will be performed. But what will be measured in these experiments? First of all, each experiment starts by creating a (basis of a) lattice L of a fixed rank d. These lattices are all full-rank, i.e., $n = d$. Then, this basis is reduced using some basis reduction algorithm, resulting in a short vector $\mathbf{b}_1$ as part of a reduced basis. The time it takes to perform this reduction is measured, and the Hermite factor $\|\mathbf{b}_1\| / \operatorname{vol}(L)^{1/d}$ is computed. Thus, the effort is measured in seconds, and the quality of the shortest vector is measured using the Hermite factor.

The lattice basis reduction algorithms that are used in the experiments are LLL and BKZ. Applying LLL will result in a vector of decent quality in a reasonable amount of time, whereas BKZ is expected to take more time but provide shorter vectors. As BKZ becomes more costly to run with higher block sizes, a range of block sizes will be used in the experiments. Hopefully, this gives insight into the relation between effort and quality.

5.4 Experiment setup

The experiments are performed on an Intel Pentium 4, with two cores of 2.80GHz. The implementations of LLL and BKZ come from the number-theoretic library NTL (version 5.5.2) by Shoup [99]. All experiments are compiled using Microsoft Visual C++ 2005 (options /O2 /GL). The lattices are generated using the generator used in the SVP challenge [21]. This generator uses a random seed to generate a random lattice in the sense of Goldstein and Mayer, as described in [32].

To measure time, the *Performance Counter* is used, which is accessible through functions in the Windows API. However, it should be noted that this measures time spent on the dedicated computer, and not necessarily on the reduction alone. In order to reduce the "noise" caused by the computer, the experiments are performed multiple times for each lattice. The minimum of the resulting time measurements is then used for the analysis.

5.5 Results

The data gathered in the experiments were transformed into triples of data points of the form $\{d, t, \gamma\}$, where d is the dimension of the lattice L, t is the time spent on the reduction and γ is the Hermite factor $\|\mathbf{b}_1\| / \operatorname{vol}(L)^{1/d}$ reached by the algorithm. For each dimension, 10 seeds were used to generate different lattices. These 10 results were combined by computing the mean of both the running time and Hermite factor. Thus, for each dimension and each algorithm, there is one triple that expresses how much time the algorithm used and what the quality of the found vector is.

Figure 5.1 contains the results of the experiments. Each color corresponds to a different algorithm. For lower dimensions, the points are quite close together, as the difference between the algorithms is still small here. In higher dimensions, different behaviors appear for different algorithms. These will now be examined more closely.

Figure 5.2 depicts the average running time for the lattice reduction algorithms as a function of the dimension. It shows that BKZ with lower block sizes behaves similar to LLL, at least in these low dimensions. Higher block sizes appear to grow significantly faster with the dimension than lower block sizes, as expected when taking the results of Gama and Nguyen into account.

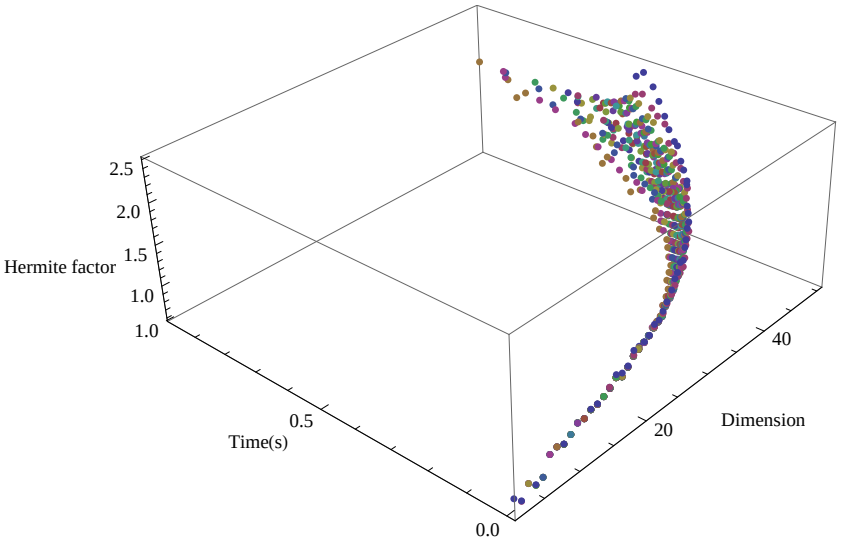

Figure 5.1: Average of the results of the experiments.

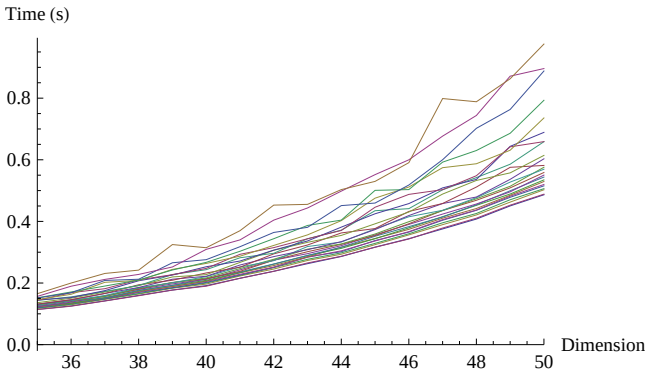

Figure 5.2: Average running time of lattice reduction algorithms in the dimension.

Figure 5.3 shows the Hermite factor of the reduced vectors that are found by the reduction algorithms as a function of the dimension. It appears to be linear, which can be explained by the fact that δ^d is approximately linear for δ close to 1, as Gama and Nguyen described.

The relation between the running time and the Hermite factor is shown in Figure 5.4. This figure shows the relation per algorithm and seems to suggest that the quality *decreases* as the running time increases. However, this is an illusion, caused by the fact that the dimension is not explicitly shown, as it is a projection of a 3-dimensional graph. To give a better view of the relation between running time and Hermite factor, Figure 5.5 shows the relation per dimension. For each dimension, the algorithms are joined into a line, with the left-hand endpoint corresponding to the LLL algorithm and the right-hand endpoint corresponding to the BKZ algorithm with block size 25. This picture suggests some sort of logarithmic relation between quality and running time, but it is unclear if this result remains true for higher dimensions. The results of Gama and Nguyen indicate that the Hermite factor is exponential in the dimension, while the running time of LLL is polynomial. As not much is known about the the running time of BKZ yet, it is unknown whether the Hermite factor will increase faster than the running time for higher dimensions.

To summarize, the results of these experiments appear to agree with the earlier results of Gama and Nguyen. However, as only small dimensions were included, the picture is still not yet

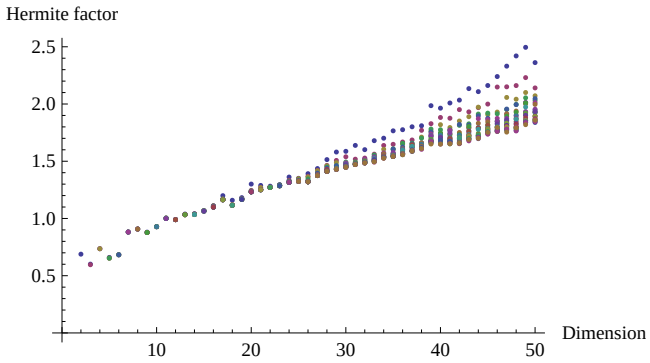

Figure 5.3: Average Hermite factor of reduced vectors in the dimension.

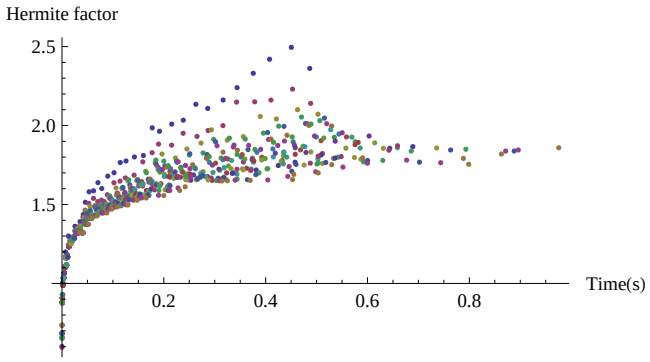

Figure 5.4: Relation between average running time and average Hermite factor per algorithm.

complete. How can it be completed? In the work of Gama and Nguyen as well as that of Rückert and Schneider, extrapolation is used to acquire information on higher dimensions. The problem with extrapolation is that it requires some knowledge of the underlying model, in order to 'fit' the experiment data to some function. But since the original problem is that the behavior of lattice reduction algorithms was not understood, how can extrapolation be done in a meaningful way?

This is a major roadblock for the assessment of the practical performance of basis reduction algorithms. As it stands now, these results are not enough to predict what happens on higher dimensions. Thus, they are not enough to determine the security of lattice-based cryptography either. But without extrapolation, it is very hard to get an idea about dimensions that will be used in practice. So, what does the way forward look like?

5.6 The way forward

5.6.1 Choosing parameters for NTRU

Before discussing a general way to move forward, it is a good idea to look at work that has been done in this area. Recall the NTRU system from Chapter 2. In an article by Hirshhorn et al. [36], the authors note the following: *"The paper [19] sounds promising but, despite the title, it does not allow us to predict lattice reduction times for a given quality of reduced basis if we, say, had computing power equivalent to* 2^{80} *operations."*

Rather than the δ used by Gama and Nguyen, Hirschhorn et al. use a slightly different measure of quality. This measure is called α and a higher α stands for a better quality. It is only based

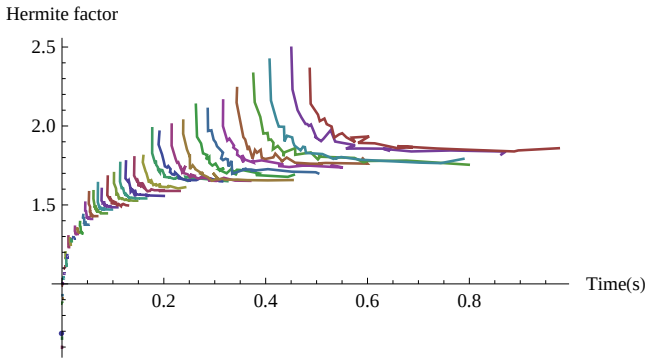

Figure 5.5: Relation between average running time and average quality per dimension.

on the reduction of the first y_2 rows, because this notion fits better in the setting of the Meet-in-the-middle attack, where an attacker combines reduction with combinatorial methods. They continue by assuming several things, most important of which is that the running time of lattice reduction depends polynomially on the dimension (i.e. n^3) and singly exponential on $1/\delta'$ (i.e. $e^{1/\delta'}$). Here, the parameter δ' is the average of $\delta_i = \log|\mathbf{b}_i^*| - \log|\mathbf{b}_{i+1}^*|$, where i lies in a specific interval $(y_1, y_2]$. Based on the Geometric Series Assumption (GSA), which was introduced by Schnorr [90], $\delta' = (1-\alpha)^2/(2(y_2 - N))$.

Finally, they suggest two levels by extrapolating the best known running times in low dimension. They note that the veracity of their model needs further examination, especially when $\alpha > 0$.

5.6.2 Other possibilities

Can anything else be done in the future to better understand the behavior of lattice reduction in practice? The experiments and subsequent results in this work are quite limited (only lattices of dimension up to 50 are used). One possible way forward is to perform more extensive experiments on lattices of higher rank in higher dimensions. However, this does not solve the problem of extrapolation.

So how to move forward? In order to make extrapolation more meaningful, there needs to be a better understanding of the theory and specifically of the running time of algorithms such as BKZ. This better understanding can then be used to model the running time, which allows for better extrapolation of experimental results. One of the problems that needs solving here is that there are no 'standard operations' of basis reduction algorithms to express their complexity. Expressing the complexity in actual running time gives different results for different implementations and different platforms.

Another interesting question is whether basis reduction algorithms perform differently on lattices with extra structure, such as ideal lattices, than they do on other lattices. This would give more insight into the practicality of the more efficient lattice-based cryptosystems. In conclusion, there is still much to be done in the area of lattice-based cryptography, and especially in understanding the practical security of lattice-based cryptosystems.

Appendix A

Ajtai-Dwork Example

$$\left(\begin{array}{cccccccccccccccccccccccc}
0.129 & 1.444 & -0.456 & -0.682 & 1.767 & -0.557 & 1.200 & -0.243 & -1.160 & 0.280 & 1.623 & -0.413 & 1.361 & 0.020 & 0.192 & 1.157 & -0.285 & 0.984 & -1.164 & -1.256 & 1.739 & 0.489 & 0.216 & -0.892 & 0.489 & 1.928 & 23.586 \\
0.079 & 1.091 & 0.173 & -0.943 & 1.425 & -0.473 & 0.358 & -0.584 & -1.265 & -0.453 & 0.672 & -0.153 & 1.257 & 0.106 & -0.327 & 0.846 & -0.541 & 1.261 & -0.672 & -0.762 & 1.875 & 0.475 & 0.329 & -1.249 & 0.633 & 1.317 & -25.074 \\
0.066 & 0.262 & -0.929 & 0.573 & 0.137 & -0.040 & 1.085 & 0.622 & 0.398 & 1.201 & 1.218 & -0.335 & -0.092 & -0.137 & 0.819 & 0.252 & 0.532 & -0.680 & -0.554 & -0.522 & -0.611 & -0.070 & -0.256 & 0.779 & -0.358 & 0.603 & -14.919 \\
0 & -1 & 0 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 1 & -1 & -1 & 0 \\
-1 & 2 & -1 & -1 & 0 & 0 & 2 & 0 & 1 & 1 & -1 & 1 & 0 & -3 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 & 1 & 0 & -1 & -1 & -1 \\
1 & 0 & -2 & 1 & 1 & -1 & 2 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & -1 \\
0 & -1 & 0 & 1 & 1 & 0 & 0 & -2 & -1 & -2 & 0 & -1 & -1 & 1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\
-1 & 1 & -1 & -1 & 1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & -1 & 0 & -1 & 1 & 1 & 1 & -1 & -1 & 0 \\
0 & -1 & 1 & 0 & 1 & 1 & 0 & -1 & 1 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & -1 & -1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & -2 & 1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & -1 & 1 & -1 & 1 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \\
0 & 0 & -1 & -1 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & -1 & 1 & -1 & 0 & 0 \\
0 & 1 & -1 & 1 & 0 & 2 & 0 & 2 & 0 & 1 & 1 & 1 & 0 & -1 & 1 & 1 & 0 & -1 & 0 & 0 & 1 & -1 & -1 & -1 & -1 & 1 & 0 \\
0 & -1 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 1 & -2 & 1 & -1 & 0 & 1 \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & -1 & -1 & -1 & -1 & 1 & 1 & 0 & 0 & -1 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & -1 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & -1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & -1 & 0 & 0 & -1 & -1 & 0 \\
-1 & 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 & -1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
1 & 0 & -1 & 1 & 0 & -1 & 1 & 1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0
\end{array}\right)$$

Bibliography

[1] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108. ACM, 1996.

[2] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19. ACM, 1998.

[3] M. Ajtai. Generating hard instances of the short basis problem. *Automata, Languages and Programming*, pages 706–706, 1999.

[4] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 284–293. ACM, 1997.

[5] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, pages 1–19, 2009.

[6] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In *Proceedings of the 1993 IEEE 34th Annual Foundations of Computer Science*, pages 724–733. IEEE Computer Society, 1993.

[7] L. Babai. On Lovászlattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[8] M. Blaze, W. Diffie, R.L. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener. Minimal key lengths for symmetric ciphers to provide adequate commercial security. A report by an Ad Hoc Group of Cryptographers and Computer Scientists, 1996.

[9] J. Blömer and J.P. Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 711–720. ACM, 1999.

[10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In H. Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer Berlin / Heidelberg, 2010.

[11] D. Coppersmith and A. Shamir. Lattice attacks on NTRU. In *Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques*, pages 52–61. Springer-Verlag, 1997.

[12] M.J. Coster, A. Joux, B.A. LaMacchia, A.M. Odlyzko, C.P. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2(2):111–128, 1992.

[13] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6), 1976.

[14] I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating CVP to within almost-polynomial factors is NP-hard. *Combinatorica*, 23(2):205–243, 2003.

[15] P. van Emde-Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. 1981.

[16] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. Odlyzko, editor, *Advances in Cryptology - CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer Berlin / Heidelberg, 1987.

[17] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of computation*, 44(170):463–471, 1985.

[18] S. Galbraith. Cryptosystems Based on Lattices. Available at `http://isg.rhul.ac.uk/~sdg/crypto-book/crypto-book.html`. Unpublished chapter from "Mathematics of Public Key Cryptography".

[19] N. Gama and P.Q. Nguyen. Predicting lattice reduction. In N. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer Berlin / Heidelberg, 2008.

[20] N. Gama, P.Q. Nguyen, and O. Regev. Lattice enumeration using extreme pruning. In H. Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 257–278. Springer Berlin / Heidelberg, 2010.

[21] N. Gama and M. Schneider. SVP Challenge. `http://www.latticechallenge.org/svp-challenge/index.php`.

[22] C.F. Gauss. Disquisitiones arithmeticae. *Gerh. Fleischer Iun*, 1801.

[23] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178. ACM, 2009.

[24] C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. In K. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer Berlin / Heidelberg, 2011.

[25] C. Gentry, J. Jonsson, J. Stern, and M. Szydlo. Cryptanalysis of the ntru signature scheme (nss) from eurocrypt 2001. In C. Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 1–20. Springer Berlin / Heidelberg, 2001.

[26] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.

[27] C. Gentry and M. Szydlo. Cryptanalysis of the revised NTRU Signature Scheme. In L. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 299–320. Springer Berlin / Heidelberg, 2002.

[28] O. Goldreich and S. Goldwasser. On the limits of non-approximability of lattice problems. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 1–9. ACM, 1998.

[29] O. Goldreich, S. Goldwasser, and S. Halevi. Eliminating decryption errors in the Ajtai-Dwork cryptosystem. In Burton Kaliski, editor, *Advances in Cryptology - CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 105–111. Springer Berlin / Heidelberg, 1997.

[30] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In B. Kaliski, editor, *Advances in Cryptology - CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer Berlin / Heidelberg, 1997.

[31] O. Goldreich, D. Micciancio, S. Safra, and J.P. Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999.

[32] D. Goldstein and A. Mayer. On the equidistribution of Hecke points. In *Forum Mathematicum*, volume 15, pages 165–189. Walter de Gruyter GmbH & Co. KG Berlin, Germany, 2003.

[33] M. Hellman and R. Merkle. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, 1978.

[34] M. Henk. Note on shortest and nearest lattice vectors. *Information Processing Letters*, 61(4):183–188, 1997.

[35] C. Hermite. Extraits de lettres de M. Ch. Hermite a M. Jacobi sur differents objets de la théorie des nombres, Deuxième lettre du 6 août 1845. *J. Reine Angew. Math*, 40:279–290, 1850.

[36] P. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte. Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In *Applied cryptography and network security*, pages 437–455. Springer, 2009.

[37] J. Hoffstein, N. Howgrave-graham, J. Pipher, J.H. Silverman, and W. Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. Full version of [38]. Draft of April 2, 2002, available at `http://www.securityinnovation.com/cryptolab/articles.shtml#2003_4`.

[38] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J.H. Silverman, and W. Whyte. NTRUSign: digital signatures using the NTRU lattice. In M. Joye, editor, *Proc. of CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 122–140. Springer Berlin Heidelberg, 2003.

[39] J. Hoffstein, N. Howgrave-Graham, J. Pipher, JH Silverman, and W. Whyte. Hybrid lattice reduction and meet in the middle resistant parameter selection for NTRU-Encrypt. *Submission/contribution to IEEE P1363*, 2007.

[40] J. Hoffstein, N. Howgrave-Graham, J. Pipher, and W. Whyte. Practical Lattice-Based Cryptography: NTRUEncrypt and NTRUSign. *The LLL Algorithm*, pages 349–390, 2010.

[41] J. Hoffstein, J. Pipher, and J. Silverman. NTRU: A new high speed public key cryptosystem, Algorithmic Number Theory (ANTS III), Portland, OR, June 1998. *Lecture Notes in Computer Science*, 1423:267–288, 1998.

[42] J. Hoffstein, J. Pipher, and J. Silverman. NSS: An NTRU Lattice-Based Signature Scheme. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 211–228. Springer Berlin / Heidelberg, 2001.

[43] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In L. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer Berlin / Heidelberg, 2002.

[44] N. Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *Proceedings of the 27th annual international cryptology conference on Advances in cryptology*, pages 150–169. Springer-Verlag, 2007.

[45] N. Howgrave-Graham, P.Q. Nguyen, D. Pointcheval, J. Proos, J. Silverman, A. Singer, and W. Whyte. The impact of decryption failures on the security of NTRU encryption. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 226–246. Springer Berlin / Heidelberg, 2003.

[46] D.S. Johnson. A Catalog of Complexity Classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A (Algorithms and Complexity), chapter 2, pages 67–161. Elsevier, 1990.

[47] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 193–206. ACM, 1983.

[48] R.M. Karp. Reducibility among combinatorial problems. *Complexity of computer computations: proceedings*, pages 85–104, 1972.

[49] S. Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM (JACM)*, 52(5):789–808, 2005.

[50] J.C. Lagarias, H.W. Lenstra, and C.P. Schnorr. Korkin-zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4):333–348, 1990.

[51] J.C. Lagarias and A.M. Odlyzko. Solving low-density subset sum problems. *J. ACM*, 32(1):229–246, 1985.

[52] J.L. Lagrange. Recherches d'arithmetique, Nouv. Mem. Acad. Roy. Sci. *Belles Lettres Berlin*, pages 265–312, 1773.

[53] A.K. Lenstra. Key lengths. In Bidgoli, H., editor, *The Handbook of Information Security*, chapter 114. Wiley, 2005.

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

[55] A.K. Lenstra and E.R. Verheul. Selecting cryptographic key sizes. *Journal of cryptology*, 14(4):255–293, 2001.

[56] H.W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

[57] H.W. Lenstra. Lattices. *Algorithmic Number Theory*, 44, 2008.

[58] Y.K. Liu, V. Lyubashevsky, and D. Micciancio. On bounded distance decoding for general lattices. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 450–461, 2006.

[59] L. Lovász. An algorithmic theory of numbers, graphs and convexity. In *CBMS-NSF Reg. Conf. Ser. Appl. Math.*, volume 50, page 91. SIAM publications, 1986.

[60] V. Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer Berlin / Heidelberg, 2009.

[61] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. *Automata, Languages and Programming*, pages 144–155, 2006.

[62] V. Lyubashevsky and D. Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 577–594. Springer Berlin / Heidelberg, 2009.

[63] K. Mahler. A theorem on inhomogeneous diophantine inequalities. In *Nederl. Akad. Wetensch., Proc*, volume 41, pages 634–637, 1938.

[64] D. Micciancio. *On the Hardness of the Shortest Vector Problem*. PhD thesis, Massachusetts Institute of Technology, 1998.

[65] D. Micciancio. Lattice based cryptography: A global improvement. *Technical report, Theory of Cryptography Library*, pages 99–05, 1999.

[66] D. Micciancio. Improving lattice based cryptosystems using the Hermite normal form. *Cryptography and Lattices*, pages 126–145, 2001.

[67] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions:(extended abstract). In *Annual IEEE symposium on Foundations Of Computer Science*, pages 356–365, 2002. Preliminary version of [68].

[68] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007.

[69] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, 2002.

[70] D. Micciancio and O. Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM Journal on Computing*, 37:267, 2007.

[71] D. Micciancio and O. Regev. Lattice-based Cryptography. 2008. Author's copy of [72].

[72] D. Micciancio and O. Regev. Lattice-based cryptography. In D. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer, 2009.

[73] H. Minkowski. *Geometrie der zahlen*. Teubner Leigzig, 1910.

[74] P. Nguyen. Cryptanalysis of the goldreich-goldwasser-halevi cryptosystem from crypto'97. In M. Wiener, editor, *Advances in Cryptology - CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 790–790. Springer Berlin / Heidelberg, 1999.

[75] P.Q. Nguyen. Hermites Constant and Lattice Algorithms. In P.Q. Nguyen and B. Vallée, editors, *The LLL Algorithm: Survey and Applications*, pages 19–69. Springer Berlin Heidelberg, 2010.

[76] P.Q. Nguyen and O. Regev. Learning a Parallelepiped: Cryptanalysis of GGH andNTRU Signatures. *Journal of Cryptology*, 22:139–160, 2009.

[77] P.Q. Nguyen and D. Stehlé. Floating-point lll revisited. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 567–567. Springer Berlin / Heidelberg, 2005.

[78] P.Q. Nguyen and J. Stern. Cryptanalysis of the Ajtai-Dwork cryptosystem. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 223–242. Springer Berlin / Heidelberg, 1998.

[79] P.Q. Nguyen and B. Vallée, editors. *The LLL Algorithm: Survey and Applications*. Springer Berlin Heidelberg, 2010.

[80] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 333–342. ACM, 2009.

[81] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. *Theory of Cryptography*, pages 145–166, 2006.

[82] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 187–196. ACM, 2008.

[83] M. Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *ACM SIGSAM Bulletin*, 15(1):37–44, 1981.

[84] J.A. Proos. Imperfect Decryption and an Attack on the NTRU Encryption Scheme. 2003.

[85] O. Regev. New lattice based cryptographic constructions. In *In Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 407–416. ACM, 2003.

[86] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *In STOC*, pages 84–93. ACM Press, 2005.

[87] R.L. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, pages 169–178, 1978.

[88] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[89] M. Rückert and M. Schneider. Estimating the security of lattice-based cryptosystems. Available at `http://eprint.iacr.org/2010/137`, 2010.

[90] C. Schnorr. Lattice reduction by random sampling and birthday methods. *STACS 2003*, pages 145–156, 2003.

[91] C.P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical computer science*, 53(2-3):201–224, 1987.

[92] C.P. Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of Algorithms*, 9(1):47–62, 1988.

[93] C.P. Schnorr. Average time fast SVP and CVP algorithms for low density lattices and the factorization of integers. 2010.

[94] C.P. Schnorr. Progress on lll and lattice reduction. In P.Q. Nguyen and B. Vallée, editors, *The LLL Algorithm: Survey and Applications*, pages 145–178. Springer Berlin Heidelberg, 2010.

[95] C.P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1):181–199, 1994.

[96] C.P. Schnorr and H.H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Proceedings of the 14th annual international conference on Theory and application of cryptographic techniques*, pages 1–12. Springer-Verlag, 1995.

[97] A. Shamir. Identity-based cryptosystems and signature schemes. In G. Blakley and D. Chaum, editors, *Advances in Cryptology – Crypto'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin / Heidelberg, 1985.

[98] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society, 1994.

[99] V. Shoup. Number Theory C++ Library (NTL). Available at `http://www.shoup.net/ntl/`.

[100] N. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In P.Q. Nguyen and D. Pointcheval, editors, *Public Key Cryptography - PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer Berlin / Heidelberg, 2010.

[101] I. Smeets. The history of the lll-algorithm. In P.Q. Nguyen and B. Vallée, editors, *The LLL Algorithm: Survey and Applications*, pages 145–178. Springer Berlin Heidelberg, 2010.

[102] D. Stehlé. Floating-point LLL: Theoretical and practical aspects. In P.Q. Nguyen and B. Vallée, editors, *The LLL Algorithm: Survey and Applications*, pages 179–213. Springer Berlin Heidelberg, 2010.

[103] D. Stehlé and R. Steinfeld. Faster fully homomorphic encryption. In M. Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 377–394. Springer Berlin / Heidelberg, 2010.

[104] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer Berlin / Heidelberg, 2010.