2019

Basis Reduction in Lattice Cryptography

Raj Kane

Basis Reduction in Lattice Cryptography

Raj Kane

Honors Thesis

Mathematics Department

Colby College

May 2019

Contents

1 Acknowledgements

I would like to begin by thanking Professor Nora Youngs for her advice and patience throughout this year. This paper would not have been possible without her help.

This paper would also not have been possible without Professor David Krumm, who fed my interest in cryptography and suggested that I look into lattices.

Next, I would like to thank Professor Leo Livshits for agreeing to be a reader for this paper and providing many helpful suggestions.

I also would like to thank Professors Livshits, Krumm, Scott Taylor, Evan Randles, and Fernando Gouvêa for their guidance through the world of math. I am incalculably grateful for all their instruction.

Finally, I would like to thank my friends, especially Shabab Ahmed, Brian Long, Yashaswi Mohanty, and Kyle McDonell, for their constant support and faith throughout my mathematical and collegiate journeys.

2 Introduction

Cryptography stems from the desire to secure private communications in the face of unwanted eavesdroppers. Developments in ensuring the secrecy of messages have been spurred by developments in cryptanalysis: the study of finding ways to gain unintended access to private communications by third parties. The field has grown from simple Caesar ciphers to the RSA cryptosystem and beyond. Now, cryptography is a heavily mathematical field, with new systems of cryptography (or *cryptosystems*) being designed around difficult mathematical problems. The difficulty of a problem ensures the secrecy of the message and is measured by the power of existing computers to algorithmically solve it.

Typically, a cryptographic scenario is framed as Bob wanting to send a secret message to Alice, with Eve aiming to eavesdrop. Bob transforms his message from readable *plaintext* into encrypted *ciphertext* by using some cryptographic algorithm; this algorithm uses a *key*, which is some mathematical information determining the output of the algorithm. Alice receives the ciphertext and deciphers it, also using a key, to reveal the plaintext and read the original message. Traditionally, Bob and Alice had to physically exchange the key(s). The Diffie-Hellman key exchange revolutionized this by having Bob and Alice exchange values g^b, g^a publicly and computing the key g^{ba} privately. Another important innovation was the RSA cryptosystem by Rivest, Shamir, and Adleman in 1978. This was one of the first public-key cryptosystems, a concept which we introduce in Chapter 7. The RSA problem can be summarized as finding the k-th roots of an arbitrary composite number P modulo N.

With the advent of quantum computing, which is tremendously more powerful than traditional computing, many cryptosystems relying on traditional problems such as the Discrete Log Problem are in danger of being broken [12]. This fear has led to the development of *post-quantum cryptography*: problems believed to be safe against attacks even by quantum computers. These problems have to be both extremely hard to break and also practical to implement.

A leading candidate in the umbrella of post-quantum cryptography is lattice-based cryptography. Cryptosystems such as NTRU are based on difficult problems related to mathematical objects called *lattices*. In Chapter 3, we develop an understanding of lattices and analyze two of the problems which render lattice-based cryptography useful: the Shortest Vector Problem and the Closest Vector Problem. We then move our attention to cryptanalytic methods with which to solve these problems. In Chapter 4, we examine an algorithm used to solve the Closest Vector Problem. In Chapter 5, we examine an algorithm used to solve the Shortest Vector Problem in two dimensions. This progression builds to a discussion of the LLL Algorithm in Chapter 6, an algorithm which is useful for solving these problems in higher dimensions. In Chapter 7, we discuss an application of the LLL Algorithm in breaking knapsack cryptosystems.

3 Lattices

A lattice is similar to a real vector space, the only restriction being that a lattice is closed under scalar multiplication only with integers and not with all real numbers. We start with a few formal definitions.

3.1 Basic Definitions and Properties

We revisit a few basic definitions from linear algebra in order to establish notation.

Definition 3.1.1. For a vector space $\mathcal{V} \subset \mathbb{R}^m$, a *basis* is a set of linearly independent vectors $\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_{n-1}$ in $\mathcal{V}$ that span $\mathcal{V}$. A basis is referred to as *orthogonal* when $\langle \mathbf{b}_i, \mathbf{b}_j \rangle = 0$ for all $i \neq j$.

Many processes become simpler when dealing with an orthogonal basis. The Gram-Schmidt Algorithm is a standard algorithm for computing the orthogonalization of a given basis. We take for granted the result of the algorithm. We define the projection operator by $\mu_{\mathbf{u}}(\mathbf{v}) := \frac{\langle \mathbf{v}, \mathbf{u} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle}\mathbf{u}$.

Definition 3.1.2. Let $\mathcal{B}$ be a basis for a vector space $\mathcal{V} \subset \mathbb{R}^m$. *The Gram-Schmidt Algorithm* is as follows:

```
Set $\mathbf{b}_0^* = \mathbf{b}_0$.
Loop $i = 1, ..., n-1$.
    Set $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=0}^{i-1} \mu_{\mathbf{b}_j^*}(\mathbf{b}_i)$.
End Loop.
Return $\mathcal{B}^* = \{\mathbf{b}_0^*, \mathbf{b}_1^*, ..., \mathbf{b}_{n-1}^*\}$.
```

To build our initial understanding of a lattice, we see that it is generated by a basis. This is a linear algebraic approach, and we consider a geometric approach later.

Definition 3.1.3. Given a basis $\mathcal{B} = \mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_{n-1}$ of $\mathbb{R}^m$, we define the *lattice* generated by $\mathcal{B}$ as

$$\mathcal{L}(\mathcal{B}) := \left\{ \sum_{i=0}^{n-1} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}.$$

Notice how slightly the definition of a lattice differs from the definition of a real linear span, which should be familiar from linear algebra.

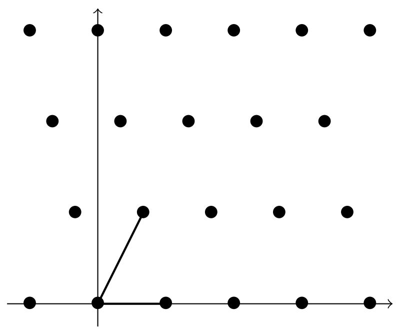

Figure 1: A basis $\mathcal{B} = \{(3,0),(2,4)\}$ and the generated lattice $\mathcal{L}(\mathcal{B})$.

Definition 3.1.4. Given a basis $\mathcal{B} = \mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_{n-1}$ of $\mathbb{R}^m$, we define the *real linear span* of $\mathcal{B}$ as

$$\mathcal{S}(\mathcal{B}) := \left\{ \sum_{i=0}^{n-1} x_i \mathbf{b}_i \mid x_i \in \mathbb{R} \right\}.$$

For brevity's sake, when we refer to a lattice we will omit mention of a basis unless necessary. So, we will typically denote an arbitrary lattice by $\mathcal{L}$, reserving the notation $\mathcal{L}(\mathcal{B})$ for when we are interested in the basis $\mathcal{B}$. In this paper, we exclusively deal with *full-rank lattices*, i.e. lattices with $n = m$. We generally assume that a given basis is a subset of $\mathbb{R}^n$ unless stated otherwise.

An important characteristic of any lattice is the length of its shortest nonzero vector. We refer to this characteristic as the minimum distance of the lattice. By length, we mean the Euclidean norm, which we will denote by $\|\|$.

Definition 3.1.5. The *minimum distance of a lattice* $\mathcal{L}$ is

$$\lambda_0(\mathcal{L}) := \inf\{\|\mathbf{v}\| \mid \mathbf{v} \in \mathcal{L} \setminus \{0\}\}.$$

Theorem 3.1.6. *For any lattice $\mathcal{L}(\mathcal{B})$, there exists a lattice vector $\mathbf{v} \in \mathcal{L}(\mathcal{B})$ such that $\|\mathbf{v}\| = \lambda_0(\mathcal{L})$.*

Proof. Proof adapted from Theorem 1.1 of [15].

Fix some $\mathcal{L}(\mathcal{B})$, where we have the basis $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_{n-1}\}$. First, we will establish that $\lambda_0(\mathcal{L})$ has a nonzero lower bound, and so that $\lambda_0(\mathcal{L})$ is positive. Consider the Gram-Schmidt orthogonalization

$$\mathcal{B}^* = \{\mathbf{b}_0^*, \mathbf{b}_1^*, ..., \mathbf{b}_{n-1}^*\}.$$

Let $\mathbf{b}^* = \min(\{\|\mathbf{b}_i^*\|\})$. We claim that

$$\lambda_0(\mathcal{L}) \geq \|\mathbf{b}^*\| > 0.$$

Consider some $\mathbf{v} \in \mathcal{L}$. Note that we can choose $x_0, x_1, ..., x_{n-1} \in \mathbb{Z}^n$ so that $\mathbf{v} = \sum_{i=0}^{n-1} x_i \mathbf{b}_i$. Let k be the largest index such that $x_k \neq 0$. We see that $\sum_{i=0}^{n-1} x_i \mathbf{b}_i = \sum_{i=0}^{k} x_i \mathbf{b}_i$. We prove that

$$\|\sum_{i=0}^{k} x_i \mathbf{b}_i\| \geq \|\mathbf{b}_k^*\| \geq \|\mathbf{b}^*\|.$$

We consider the dot product of $\mathbf{v}$ and $\mathbf{b}_k^*$:

$$\langle \sum_{i=0}^{k} x_i \mathbf{b}_i, \mathbf{b}_k^* \rangle = \sum_{i=0}^{k} \langle \mathbf{b}_i, \mathbf{b}_k^* \rangle x_i = \langle \mathbf{b}_k, \mathbf{b}_k^* \rangle x_k,$$

since $\mathbf{b}_i$ and $\mathbf{b}_k^*$ are orthogonal for all $i < k$.
By the Gram-Schmidt Algorithm and the linearity of the dot product:

$$\begin{aligned}\langle \mathbf{b}_k, \mathbf{b}_k^* \rangle x_k &= \langle \mathbf{b}_k^* + \sum_{i=0}^{k-1} \mu_{\mathbf{b}_i^*}(\mathbf{b}_k), \mathbf{b}_k^* \rangle x_k \\ &= \langle \mathbf{b}_k^*, \mathbf{b}_k^* \rangle x_k + \sum_{i=0}^{k-1} \frac{\langle \mathbf{b}_i^*, \mathbf{b}_k \rangle}{\langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle} \langle \mathbf{b}_i^*, \mathbf{b}_k^* \rangle x_k.\end{aligned}$$

It follows that $\langle \mathbf{b}_k, \mathbf{b}_k^* \rangle x_k = \|\mathbf{b}_k^*\|^2 x_k$. By the Cauchy-Schwarz Inequality:

$$\|\sum_{i=0}^{k} x_i \mathbf{b}_i\| \|\mathbf{b}_k^*\| \geq \left| \|\mathbf{b}_k^*\|^2 x_k \right| = \|\mathbf{b}_k^*\|^2 \, |x_k| \, \geq \|\mathbf{b}_k^*\|^2,$$

since x_k is a nonzero integer. Thus, we have shown that $\|\sum_{i=0}^{k} x_i \mathbf{b}_i\| \geq \|\mathbf{b}_k^*\| \geq \|\mathbf{b}^*\|$. Therefore, $\|\mathbf{v}\| > 0$ for any $\mathbf{v} \in \mathcal{L}(\mathcal{B})$. We have thus established a positive lower bound for $\lambda_0(\mathcal{L})$, and now all that remains is to show that there is a nonzero lattice vector of length exactly $\lambda_0(\mathcal{L})$.

Since $\lambda_0(\mathcal{L})$ is by definition the infimum of a set of real numbers, there must exist a sequence of lattice vectors $\mathbf{v}_i \in \mathcal{L}$ such that

$$\lim_{i \to \infty} \|\mathbf{v}_i\| = \lambda_0(\mathcal{L}).$$

Hence, for large enough indices i, $\|\mathbf{v}_i\| \leq 2\lambda_0(\mathcal{L})$ and so eventually the lattice vectors $\mathbf{v}_i$ belong to the closed ball $R = \{\mathbf{r} \in \mathbb{R}^m \mid \|\mathbf{r}\| \leq 2\lambda_0(\mathcal{L})\}$. Since R is compact, there is a convergent subsequence of vectors $\mathbf{v}_{i_j}$ with limit $\mathbf{w} \in R$. We will show that $\mathbf{w} \in \mathcal{L}$. Since $\lim\limits_{j\to\infty} \mathbf{v}_{i_j} = \mathbf{w}$, we have that $\lim\limits_{j\to\infty} \|\mathbf{v}_{i_j}\| = \|\mathbf{w}\|$ and hence that $\lim\limits_{j\to\infty} \|\mathbf{v}_{i_j} - \mathbf{w}\| = 0$. Hence, for large enough indices j, $\|\mathbf{v}_{i_j} - \mathbf{w}\| < \frac{\lambda_0(\mathcal{L})}{2}$. So, for large enough j and for all $l > j$,

$$\|\mathbf{v}_{i_j} - \mathbf{v}_{i_l}\| \leq \|\mathbf{v}_{i_j} - \mathbf{w}\| + \|\mathbf{v}_{i_l} - \mathbf{w}\| < \lambda_0(\mathcal{L}).$$

Note that since $\mathbf{v}_{i_j} - \mathbf{v}_{i_l} \in \mathcal{L}$ and $\|\mathbf{v}_{i_j} - \mathbf{v}_{i_l}\| < \lambda_0(\mathcal{L})$, it must be the case that $\mathbf{v}_{i_j} - \mathbf{v}_{i_l} = 0$. Thus, $\lim\limits_{l\to\infty} \mathbf{v}_{i_l} = \mathbf{v}_{i_j}$. In other words, $\mathbf{w} = \mathbf{v}_{i_j}$, and we have shown that $\mathbf{w} \in \mathcal{L}$, as desired.

□

3.2 Fundamental Domains

We now introduce the concept of a fundamental domain for lattices.

Definition 3.2.1. For any basis $\mathcal{B}$, we define the *fundamental domain* (or *fundamental parallelepiped*) of $\mathcal{L}(\mathcal{B})$ to be the set

$$\mathcal{F}(\mathcal{B}) := \left\{ \sum_{i=0}^{n-1} x_i \mathbf{b}_i \mid 0 \leq x_i < 1 \right\}.$$

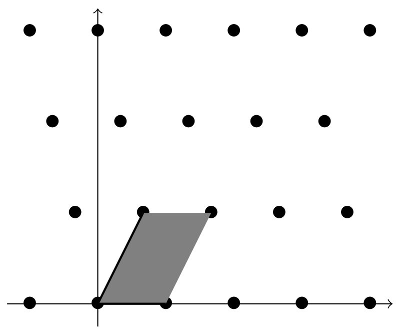

Figure 2: The fundamental domain for the lattice generated by $\{(3,0),(2,4)\}$.

Fundamental domains have many interesting properties. One essential property to note is that the volume of a lattice's fundamental domain is an invariant. To see this property, we first define a unimodular matrix.

Definition 3.2.2. A square matrix $\mathbf{U}$ of integer entries is *unimodular* if $|\det(\mathbf{U})| = 1$.

The crux of this paper, as we will see, lies in the idea that multiple bases can generate the same lattice. Given a basis $\mathcal{B}$, we refer to its matrix representation by $\mathbf{B}$, so that

$$\mathbf{B} = \begin{bmatrix} \mathbf{b_0} \\ \mathbf{b_1} \\ \vdots \\ \mathbf{b_{n-1}} \end{bmatrix}.$$

Now, we will formalize the condition that allows two bases to generate the same lattice. This condition has to do with the concept of a unimodular matrix.

Theorem 3.2.3. *For two bases $\mathcal{B}_0, \mathcal{B}_1$ for $\mathbb{R}^n$, $\mathcal{L}(\mathcal{B}_0) = \mathcal{L}(\mathcal{B}_1)$ if and only if there exists a unimodular matrix $\mathbf{U}$ such that $\mathbf{B}_0 = \mathbf{B}_1\mathbf{U}$.*

Proof. Proof adapted from Theorem 2 of [14].

Let two bases $\mathcal{B}_0, \mathcal{B}_1$ for $\mathbb{R}^n$ be given. Observe that if $\mathbf{U}$ is unimodular, then so is $\mathbf{U}^{-1}$. First, suppose that $\mathbf{B}_0 = \mathbf{B}_1\mathbf{U}$ for some unimodular $\mathbf{U}$. Since $\mathbf{B}_0 = \mathbf{B}_1\mathbf{U}$, by the definition of a lattice it follows that $\mathcal{L}(\mathcal{B}_0) \subset \mathcal{L}(\mathcal{B}_1)$. Since $\mathbf{B}_1 = \mathbf{B}_0\mathbf{U}^{-1}$, by the definition of a lattice it follows that $\mathcal{L}(\mathcal{B}_1) \subset \mathcal{L}(\mathcal{B}_0)$.

Now, we prove the other direction. Suppose that $\mathcal{L}(\mathcal{B}_1) = \mathcal{L}(\mathcal{B}_0)$. Then, by the definition of a lattice, there exists square integer matrices $\mathbf{M}, \mathbf{N}$ such that $\mathbf{B}_0 = \mathbf{B}_1\mathbf{N}$ and $\mathbf{B}_1 = \mathbf{B}_0\mathbf{M}$. It follows that $\mathbf{B}_0(\mathbf{I} - \mathbf{N}\mathbf{M}) = \mathbf{0}$, where $\mathbf{I}$ is the identity matrix. Since the row vectors of $\mathbf{B}_0$ are linearly independent, it follows that $\mathbf{I} = \mathbf{N}\mathbf{M}$. Hence,

$$\det(\mathbf{NM}) = \det(\mathbf{N})\det(\mathbf{M}) = \det(\mathbf{I}) = 1.$$

Since the entries of $\mathbf{N}, \mathbf{M}$ are integers, it follows that

$$|\det(\mathbf{N})| = |\det(\mathbf{M})| = 1.$$

□

We now define the determinant of a lattice.

Definition 3.2.4. For any lattice $\mathcal{L}(\mathcal{B})$, we define the *determinant* of that lattice to be $\det(\mathcal{L}(\mathcal{B})) := \mathrm{Vol}(\mathcal{F}(\mathcal{B}) = \prod_{i=0}^{n-1} \|\mathbf{b}_i^*\|$.

Next, we prove a different representation of the determinant of a lattice.

Theorem 3.2.5. *For any basis $\mathcal{B}$, we can express $det(\mathcal{L}(\mathcal{B}))$ as*

$$det(\mathcal{L}(\mathcal{B})) = \sqrt{det(\mathbf{B}^T\mathbf{B})}.$$

Proof. Proof adapted from Theorem 6 of [14].

Consider some basis $\mathcal{B}$. Consider also the Gram-Schmidt orthogonalization $\mathcal{B}^*$ and the corresponding matrix $\mathbf{B}^*$. Observe that, by the Gram-Schmidt Algorithm, there is an upper triangular matrix $\mathbf{T}$ with all 1s on the main diagonal and the coefficients $\frac{\langle \mathbf{b}_j^*, \mathbf{b}_i \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$ at the position $\mathbf{T}_{ji}$ for all $j < i$, such that $\mathbf{B} = \mathbf{B}^*\mathbf{T}$. Thus, we can write

$$\begin{aligned}\sqrt{\det(\mathbf{B}^T\mathbf{B})} &= \sqrt{\det(\mathbf{B}^T(\mathbf{B}^*\mathbf{T}))}\\ &= \sqrt{\det((\mathbf{T}^T\mathbf{B}^{*T})(\mathbf{B}^*\mathbf{T}))}\\ &= \sqrt{\det(\mathbf{T}^T)\det(\mathbf{B}^{*T}\mathbf{B}^*)\det(\mathbf{T})}.\end{aligned}$$

Since $\mathbf{T}, \mathbf{T}^T$ are triangular matrices, both $\det(\mathbf{T}), \det(\mathbf{T}^T)$ are the products of their respective diagonal entries. So, $\det(\mathbf{T}^T) = \det(\mathbf{T}^T) = 1$. Since both $\mathbf{B}, \mathbf{B}^*$ have orthogonal columns, the matrix $\mathbf{B}^*\mathbf{B}$ is diagonal. Hence, $\det(\mathbf{B}^*\mathbf{B})$, is the product of the diagonal entries of $\mathbf{B}^*\mathbf{B}$. Thus,

$$\det(\mathbf{B}^{*T}\mathbf{B}^*) = \prod_{i=0}^{n-1} \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle = \prod_{i=0}^{n-1} \|\mathbf{b}_i^*\|^2 = \det(\mathcal{L})^2,$$

and so

$$\begin{aligned}\sqrt{\det(\mathbf{B}^T\mathbf{B})} &= \sqrt{\det(\mathbf{T}^T)\det(\mathbf{B}^{*T}\mathbf{B}^*)\det(\mathbf{T})}\\ &= \sqrt{\det(\mathbf{B}^{*T}\mathbf{B}^*)}\\ &= \det(\mathcal{L}).\end{aligned}$$

□

We are now ready to prove that the volume of a lattice's fundamental domain is an invariant, using the previous few results.

Theorem 3.2.6. *For any two bases $\mathcal{B}_0, \mathcal{B}_1$, if $\mathcal{L}(\mathcal{B}_0)) = \mathcal{L}(\mathcal{B}_1))$, then*

$$det(\mathcal{L}(\mathcal{B}_0)) = det(\mathcal{L}(\mathcal{B}_1)).$$

Proof. Proof adapted from Theorem 7 of [14].

Consider two bases $\mathcal{B}_0, \mathcal{B}_1$ such that $\mathcal{L}(\mathcal{B}_0)) = \mathcal{L}(\mathcal{B}_1))$. By Theorem 3.2.3, there exists a unimodular matrix $\mathbf{U}$ such that $\mathbf{B}_0 = \mathbf{B}_1\mathbf{U}$. By Theorem 3.2.5, and since $\det(\mathbf{U}) = \det(\mathbf{U}^T) = 1$ we have that

$$\begin{aligned}
\det(\mathcal{L}(\mathcal{B}_0)) &= \sqrt{\det(\mathbf{B}_0^T\mathbf{B}_0)} \\
&= \sqrt{\det(\mathbf{B}_0^T(\mathbf{B}_1\mathbf{U}))} \\
&= \sqrt{\det((\mathbf{U}^T\mathbf{B}_1^T)(\mathbf{B}_1\mathbf{U}))} \\
&= \sqrt{\det(\mathbf{U}^T)\det(\mathbf{B}_1^T\mathbf{B}_1)\det(\mathbf{U})} \\
&= \sqrt{\det(\mathbf{B}_1^T\mathbf{B}_1)} \\
&= \det(\mathcal{L}(\mathcal{B}_1)).
\end{aligned}$$

□

Thus, we have shown that every fundamental domain of a lattice has the same volume. Another interesting property to note is the relationship between the volume of a lattice's fundamental domain and a measure of the orthogonality of its basis. This relationship is represented by the Hadamard Inequality.

Definition 3.2.7. For any lattice $\mathcal{L}(\mathcal{B})$, the *Hadamard Inequality* states that

$$\det(\mathcal{L})) \leq \prod_{i=0}^{n-1} \|\mathbf{b}_i\|.$$

Equality is achieved only when the given basis is orthogonal. We use the term *Hadamard ratio* to refer to the ratio

$$\frac{\prod_{i=0}^{n-1} \|\mathbf{b}_i\|}{\det(\mathcal{L})}.$$

This makes sense by noting that since the basis vectors form the sides of the fundamental domain, the volume of the fundamental domain is maximized precisely when the basis vectors are orthogonal to one another.

A crucial aspect of the fundamental domain is that it provides a tiling of $\mathbb{R}^n$.

Theorem 3.2.8. *For any basis $\mathcal{B}$, a given vector $\mathbf{w} \in \mathbb{R}^n$ can be expressed uniquely as the sum $\mathbf{f} + \mathbf{v}$ for a pair $\mathbf{f} \in \mathcal{F}(\mathcal{B})$, $\mathbf{v} \in \mathcal{L}(\mathcal{B})$.*

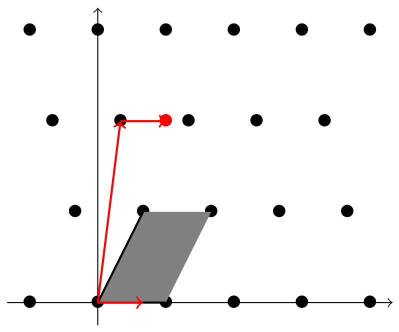

Figure 3: Any real vector is the sum of a unique pair of one lattice vector and one vector in the fundamental domain.

Proof. Proof adapted from Proposition 7.18 in [7].

Fix some basis $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_{n-1}\}$. Let some $\mathbf{w} \in \mathbb{R}^n$ be given. Since the vectors $\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_{n-1}$ are linearly independent, they generate $\mathbb{R}^n$. So, we can write

$$\mathbf{w} = \alpha_0 \mathbf{b}_0 + \alpha_1 \mathbf{b}_1 + ... + \alpha_{n-1} \mathbf{b}_{n-1}$$

for some $\{\alpha_i\} \subset \mathbb{R}$. We can further express each α_i as $\alpha_i = f_i + v_i$ for some $f_i \in [0, 1), v_i \in \mathbb{Z}$. So,

$$\begin{aligned} \mathbf{w} &= (f_0 + v_0)\mathbf{b}_0 + ... + (f_{n-1} + v_{n-1})\mathbf{b}_{n-1} \\ &= (f_0 \mathbf{b}_0 + ... + f_{n-1}\mathbf{b}_{n-1}) + (v_0 \mathbf{b}_0 + ... + v_{n-1}\mathbf{b}_{n-1}). \end{aligned}$$

Note that

$$\mathbf{f} = f_0 \mathbf{b}_0 + ... + f_{n-1}\mathbf{b}_{n-1} \in \mathcal{F}(\mathcal{B}),$$
$$\mathbf{v} = v_0 \mathbf{b}_0 + ... + v_{n-1}\mathbf{b}_{n-1} \in \mathcal{L}(\mathcal{B}),$$

giving us the existence of the desired pair of vectors so that $\mathbf{w} = \mathbf{f} + \mathbf{v}$. Now, we show uniqueness. Suppose $\mathbf{w} = \mathbf{e} + \mathbf{u}$, where

$$\mathbf{e} = e_0 \mathbf{b}_0 + ... + e_{n-1}\mathbf{b}_{n-1} \in \mathcal{F}(\mathcal{B}),$$

$$\mathbf{u} = u_0 \mathbf{b}_0 + ... + u_{n-1}\mathbf{b}_{n-1} \in \mathcal{L}(\mathcal{B}),$$

and $\{e_i\} \subset [0, 1)$, $\{u_i\} \subset \mathbb{Z}$. Then, since $\mathbf{b}_0, ..., \mathbf{b}_{n-1}$ are linearly independent, we have that

$$f_i + v_i = e_i + u_i$$

for all $0 \le i \le n-1$. Hence, for each i, $f_i - e_i = u_i - v_i \in \mathbb{Z}$. Since each $f_i, e_i \in [0,1)$, we must have that $f_i - e_i = 0$. So, $u_i - v_i = 0$. Thus, $\mathbf{f} = \mathbf{e}$ and $\mathbf{v} = \mathbf{u}$, as desired. $\square$

3.3 Successive Minima

It can be useful to consider not only the shortest nonzero distance of a lattice, but also the second shortest nonzero distance, the third shortest, and so on. We refer to these as the successive minima of a lattice, and denote them as $\lambda_0(\mathcal{L})$, $\lambda_1(\mathcal{L})$, $\lambda_2(\mathcal{L})$, $\lambda_3(\mathcal{L})$,...,$\lambda_{n-1}(\mathcal{L})$. The successive minima of a lattice are defined similarly to the minimum distance of a lattice, and a similar argument as the one above shows that they are in fact achieved by lattice vectors. We now present a few results which will build toward providing an upper bound for the successive minima of a lattice. The first of these theorems, Blichfeld's Theorem, makes a statement on the determinant of a lattice.

Theorem 3.3.1. *For any lattice $\mathcal{L}(\mathcal{B})$ and measurable set $S \subset \mathbb{R}^n$ with $Vol(S) > det(\mathcal{L})$, there are two distinct $\mathbf{z_1}, \mathbf{z_2} \in S$ such that $\mathbf{z_1} - \mathbf{z_2} \in \mathcal{L}$.*

Proof. Proof adapted from Theorem 8 in [17].

Consider some lattice $\mathcal{L}(\mathcal{B})$ and measurable set $S \subset \mathbb{R}^n$. By Theorem 3.1.9, the translates of $\mathcal{F}(\mathcal{B})$ partition $\mathbb{R}^n$. For $\mathbf{x} \in \mathcal{L}$, let

$$S_{\mathbf{x}} = S \cap (\mathbf{x} + \mathcal{F}(\mathcal{B})), \hat{S}_{\mathbf{x}} = S_{\mathbf{x}} - \mathbf{x}.$$

Since $S = \bigcup_{\mathbf{x} \in \mathcal{L}} S_{\mathbf{x}}$, we have that

$$\text{Vol}(S) = \sum_{\mathbf{x} \in \mathcal{L}} \text{Vol}(S_{\mathbf{x}}).$$

Note that $\hat{S}_{\mathbf{x}} \subset \mathcal{F}(\mathcal{B})$ and $\text{Vol}(\hat{S}_{\mathbf{x}}) = \text{Vol}(S_{\mathbf{x}})$. Thus,

$$\begin{aligned} \sum_{\mathbf{x} \in \mathcal{L}} \text{Vol}(\hat{S}_{\mathbf{x}}) &= \sum_{\mathbf{x} \in \mathcal{L}} \text{Vol}(S_{\mathbf{x}}) \\ &= \text{Vol}(S) \\ &> \text{Vol}(\mathcal{F}(\mathcal{B})). \end{aligned}$$

Therefore, there must be distinct $\mathbf{z_1}, \mathbf{z_2} \in \mathcal{L}$ such that $\hat{S}_{\mathbf{z_1}} \cap \hat{S_{\mathbf{z_2}}} \neq \varnothing$. Pick some $\mathbf{z} \in \hat{S}_{\mathbf{z_1}} \cap S_{\mathbf{z_2}}$. Note that

$$\mathbf{z} + \mathbf{z_1} \in S_{\mathbf{z_1}} \subset S, \mathbf{z} + \mathbf{z_2} \in S_{\mathbf{z_2}} \subset S.$$

Thus, $(\mathbf{z} + \mathbf{z_1}) - (\mathbf{z} + \mathbf{z_2}) = \mathbf{z_1} - \mathbf{z_2} \in \mathcal{L}$, as desired. $\square$

Before discussing the results of Blichfeld's Theorem, we provide a few definitions from geometry.

Definition 3.3.2. A set is $S \subset \mathbb{R}^n$ is *centrally symmetric* if for any $\mathbf{x} \in S$, there also exists $-\mathbf{x} \in S$.

Definition 3.3.3. A set $S \subset \mathbb{R}^n$ is *convex* if for any $\mathbf{x}, \mathbf{y} \in S$ and $\alpha \in [0,1]$, there also exists $\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in S$.

As a result of Blichfeld's Theorem, we attain the following corollary due to Minkowski, which is called the Convex Body Theorem.

Theorem 3.3.4. *For any lattice $\mathcal{L}(\mathcal{B})$ and centrally symmetric convex set $S \subset \mathbb{R}^n$, if $Vol(S) > 2^n det(\mathcal{L})$ then S contains a nonzero $\mathbf{v} \in \mathcal{L}$.*

Proof. Proof adapted from Theorem 9 in [17].

Consider some lattice $\mathcal{L}(\mathcal{B})$ and centrally symmetric convex set $S \subset \mathbb{R}^n$ such that $\mathrm{Vol}(S) > 2^n \det(\mathcal{L})$. Let $\hat{S} = \frac{1}{2}S$. Note that

$$\mathrm{Vol}(\hat{S}) = 2^{-n}\mathrm{Vol}(S) > \det(\mathcal{L}).$$

Then, by Blichfeld's Theorem, there are two distinct $\mathbf{z_1}, \mathbf{z_2} \in \hat{S}$ such that $\mathbf{z_1} - \mathbf{z_2} \in \mathcal{L}$. By construction, $2\mathbf{x_1}, 2\mathbf{z_2} \in S$. Since S is centrally symmetric, $-2\mathbf{x_2} \in S$. Since S is convex, $\frac{2\mathbf{z_1}-2\mathbf{z_2}}{2} = \mathbf{z_1} - \mathbf{z_2} \in S$, as desired. □

We are now ready to show a bound for the minimum distance and the successive minima of a lattice. The following corollary is called Minkowski's First Theorem.

Theorem 3.3.5. *For any lattice $\mathcal{L}(\mathcal{B})$, $\lambda_0(\mathcal{L}) \leq \sqrt{n}(det(\mathcal{L}))^{1/n}$.*

Proof. Proof adapted from Corollary 2 in [17].

Consider some lattice $\mathcal{L}(\mathcal{B})$. We denote the open neighborhood of radius a centered at b by $\mathcal{N}_a(b)$. Let $\epsilon > 0$. Consider the cube

$$X = \left\{ \mathbf{x} \in \mathbb{R}^n \mid |x_i| < \frac{2\epsilon}{n}, 0 \leq i \leq n-1 \right\},$$

where we write $\mathbf{x} = (x_0, x_1, ..., x_{n-1})$. Note that $X \subset \mathcal{N}_\epsilon(0)$, and that each side of X has length $\frac{2\epsilon}{\sqrt{n}}$. Thus, $\mathrm{Vol}(\mathcal{N}_\epsilon(0)) \geq \frac{2\epsilon}{\sqrt{n}}^n$. Also note that $\mathcal{L} \cap \mathcal{N}_{\lambda_0(\mathcal{L})}(0) = \varnothing$. By the Convex Body Theorem,

$$\frac{(2\lambda_0(\mathcal{L}))^n}{(\sqrt{n})^n} \leq \mathrm{Vol}(\mathcal{N}_{\lambda_0(\mathcal{L})}(0)) \leq 2^n \det(\mathcal{L}).$$

It follows that $\lambda_0(\mathcal{L}) \leq \sqrt{n}(\det(\mathcal{L}))^{1/n}$, as desired. □

The following result, called Minkowski's Second Theorem, strengthens the bound given by Minkowski's First Theorem by giving a bound that considers the geometric mean of all successive minima $\lambda_i, 0 \leq i \leq n-1$.

Theorem 3.3.6. *For any lattice $\mathcal{L}(\mathcal{B})$,*

$$\Big(\prod_{i=0}^{n-1} \lambda_i(\mathcal{L})\Big)^{1/n} \leq \sqrt{n}(det(\mathcal{L}))^{1/n}.$$

Proof. Proof adapted from Theorem 3 of [17].

Consider some lattice $\mathcal{L}(\mathcal{B})$. Let $X = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{n-1}\} \subset \mathcal{L}$ be a basis for $\mathbb{R}^n$ such that each $\|\mathbf{x}_i\| = \lambda_i(\mathcal{L})$. Let X^* be the Gram-Schmidt orthogonalization of X. Consider the open ellipsoid E with axes $\mathbf{x_0}^*, \mathbf{x_1}^*, ..., \mathbf{x_{n-1}}^*$:

$$E = \Big\{\mathbf{y} \in \mathbb{R}^n \mid \sum_{i=0}^{n-1} \Big(\frac{\langle \mathbf{y}, \mathbf{x_i}^*\rangle}{\|\mathbf{x_i}^*\|\lambda_i(\mathcal{L})}\Big)^2 < 1\Big\}.$$

We will first show that E does not contain any nonzero lattice vector. Pick some nonzero $\mathbf{w} \in \mathcal{L}$ and the largest index k such that $\|\mathbf{w}\| \geq \lambda_k(\mathcal{L})$. By our choice of X, it follows that $\mathbf{w} \in \mathcal{S}(\{\mathbf{x}_0^*, ..., \mathbf{x}_k^*\})$. Observe that

$$\begin{aligned}
\sum_{i=0}^{n-1} \Big(\frac{\langle \mathbf{w}, \mathbf{x_i}^*\rangle}{\|\mathbf{x_i}^*\|\lambda_i(\mathcal{L})}\Big)^2 &= \sum_{i=0}^{k} \Big(\frac{\langle \mathbf{w}, \mathbf{x_i}^*\rangle}{\|\mathbf{x_i}^*\|\lambda_i(\mathcal{L})}\Big)^2 \\
&\geq \sum_{i=0}^{n-1} \Big(\frac{\langle \mathbf{w}, \mathbf{x_i}^*\rangle}{\|\mathbf{x_i}^*\|\lambda_k(\mathcal{L})}\Big)^2 \\
&= \lambda_k(\mathcal{L})^{-2} \sum_{i=0}^{n-1} \Big(\frac{\langle \mathbf{w}, \mathbf{x_i}^*\rangle}{\|\mathbf{x_i}^*\|}\Big)^2 \\
&= \frac{\|\mathbf{w}\|^2}{\lambda_k(\mathcal{L})^2} \\
&\geq 1.
\end{aligned}$$

Thus, $\mathbf{w} \notin E$. Now, by the Convex Body Theorem,

$$\text{Vol}(E) \geq 2^n \text{det}(\mathcal{L}).$$

We also have that

$$\text{Vol}(E) = \Big(\prod_{i=0}^{n-1} \lambda_i\Big)\text{Vol}(\mathcal{N}_1(0)) \geq \Big(\prod_{i=0}^{n-1} \lambda_i\Big)\Big(\frac{2}{\sqrt{n}}\Big)^n.$$

Therefore,

$$\Big(\prod_{i=0}^{n-1} \lambda_i\Big)\Big(\frac{2}{\sqrt{n}}\Big)^n \leq 2^n \det(\mathcal{L}),$$

i.e.

$$\Big(\prod_{i=0}^{n-1} \lambda_i\Big)^{1/n} \leq \sqrt{n}\det(\mathcal{L})^{1/n}.$$

□

We have thus far introduced a linear algebraic concept of lattices. This definition is convenient for use in computer science. At the same time, it can also be helpful to look at lattices geometrically. We will reconcile the linear algebraic and geometric definitions of lattices.

Definition 3.3.7. A *discrete additive subgroup* of $\mathbb{R}^n$ is an additive subgroup $\mathcal{G}$ of $\mathbb{R}^n$ such that for all $\mathbf{v} \in \mathcal{G}$, there exists $\epsilon \in \mathbb{R}$ such that

$$\{\mathbf{w} \in \mathcal{G} \mid \|\mathbf{v} - \mathbf{w}\| < \epsilon\} = \{\mathbf{v}\}.$$

Theorem 3.3.8. *For any subset $L \subset \mathbb{R}^n$, the following statements are equivalent.*

1. $L = \Big\{\sum_{i=0}^{n-1} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\Big\}$ *for some basis* $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_{n-1}\}$.
2. *L is a discrete additive subgroup of* $\mathbb{R}^n$.

In proving Theorem 3.1.10, we will use the concept of a closed fundamental domain.

Definition 3.3.9. For any basis $\mathcal{B}$, we define the *closed fundamental domain* of $\mathcal{L}(\mathcal{B})$ to be the set

$$\overline{\mathcal{F}}(\mathcal{B}) := \left\{\sum_{i=0}^{n-1} x_i \mathbf{b}_i \mid 0 \leq x_i \leq 1\right\}.$$

We now proceed to the proof of Theorem 3.1.10.

Proof. Let $\mathcal{B} \subset \mathbb{R}^n$ be a set of linearly independent vectors $\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_{n-1}$ generating a subset $L \subset \mathbb{R}^n$. Since L is the set of all integer linear combinations of $\mathcal{B}$, for any vectors $\sum_{i=0}^{n-1} x_i \mathbf{b}_i$ and $\sum_{i=0}^{n-1} y_i \mathbf{b}_i$ in L we have that

$$\sum_{i=0}^{n-1} (x_i + y_i)\mathbf{b}_i, \sum_{i=0}^{n-1} (-x_i)\mathbf{b}_i \in L.$$

Note that $\sum_{i=0}^{n-1} 0\mathbf{b}_i = \mathbf{0} \in L$. Thus, L is an additive subgroup of $\mathbb{R}^n$. Then, for any distinct $\mathbf{v}, \mathbf{w} \in L$, we must have that $\mathbf{v} - \mathbf{w} \in L$. It follows that $\|\mathbf{v} - \mathbf{w}\| \geq \lambda_0(L)$. Let $\epsilon = \lambda_0(L)$. Then,

$$\{\mathbf{z} \in L \mid \|\mathbf{v} - \mathbf{z}\| < \epsilon\} = \{\mathbf{v}\},$$

as desired. Therefore, L is a discrete additive subgroup of $\mathbb{R}^n$.

We have shown that Definition 3.1.3 implies Definition 3.1.10. We will now prove the opposite direction. The proof in this direction is adapted from [21].

Let L be a discrete additive subgroup of $\mathbb{R}^n$. Choose $\mathbf{y} \in L$ such that there is no $\hat{\mathbf{y}} \in L$ where $\hat{\mathbf{y}} = \alpha\mathbf{y}$ and $\alpha \in (0, 1)$. Let $\mathbf{b}_0 = \mathbf{y}$. We now describe a recursive method for selecting the vectors $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{n-1}$.

For $0 \leq i < n - 1$, suppose $\mathbf{b}_0, ..., \mathbf{b}_i$ have already been chosen. Choose $\mathbf{w}_i \in L$ such that $\mathbf{w}_i \notin \mathcal{S}(\{\mathbf{b}_0, ..., \mathbf{b}_i\})$. Now, consider the closed fundamental domain $\overline{\mathcal{F}}(\{\mathbf{b}_0, ..., \mathbf{b}_i, \mathbf{w}_i\})$. Notice that $\overline{\mathcal{F}}(\{\mathbf{b}_0, ..., \mathbf{b}_i, ..., \mathbf{w}_i\})$ contains $\mathbf{w}_i$ and so contains at least one member of L. Observe that the intersection $L^{'} = L \cap \overline{\mathcal{F}}(\{\mathbf{b}_0, ..., \mathbf{b}_i, ..., \mathbf{w}_i\})$ is closed, and so since $\overline{\mathcal{F}}(\{\mathbf{b}_0, ..., \mathbf{b}_i, ..., \mathbf{w}_i\})$ is compact $L^{'}$ must also be compact. Further observe that $L^{'}$ is discrete. Hence, by a result of topology, $L^{'}$ is finite. By construction, we must have a member of L in the set difference

$$\overline{\mathcal{F}}_i = \overline{\mathcal{F}}(\{\mathbf{b}_0, ..., \mathbf{b}_i, \mathbf{w}_i\}) \setminus \mathcal{S}(\{\mathbf{b}_0, ..., \mathbf{b}_i\}).$$

We can express the distance from any singleton $\{\alpha\}$ to a set B as

$$\text{dist}(\{\alpha\}, B) = \inf\{\|\alpha - \beta\| \mid \beta \in B\}.$$

It is a well-known result that this distance is well-defined and is achieved by some $\beta \in B$ if B is closed. Thus, since $L \cap \overline{\mathcal{F}}_i$ is compact, for any $\alpha \in L \cap \overline{\mathcal{F}}_i$ we can find β_α such that $\text{dist}(\{\alpha\}, B) = \|\alpha - \beta_\alpha\|$. Further, since $L \cap \overline{\mathcal{F}}_i$ is finite, we can find some $\mathbf{w_{i+1}} \in L \cap \overline{\mathcal{F}}_i$ such that

$$\text{dist}(\{\mathbf{w}_{i+1}\}, L \cap \overline{\mathcal{F}}_i) = \min\left\{\text{dist}(\{\alpha\}, \mathcal{S}(\{\mathbf{b}_0, ..., \mathbf{b}_i\}) \mid \alpha \in L \cap \overline{\mathcal{F}}_i\right\}.$$

Let $\mathbf{b}_{i+1} = \mathbf{w}_{i+1}$.

Since by construction each $\mathbf{b}_{i+1} \notin \mathcal{S}(\{\mathbf{b}_0, ..., \mathbf{b}_i\})$, the vectors $\mathbf{b}_0, ..., \mathbf{b}_{n-1}$ are linearly independent. We claim that

$$L = \left\{\sum_{i=0}^{n-1} x_i\mathbf{b}_i \mid x_i \in \mathbb{Z}\right\}.$$

Note that each $\mathbf{b}_i \in L$ and that L has closure under additivity as it is an additive subgroup of $\mathbb{R}^n$. It follows that

$$\left\{ \sum_{i=0}^{n-1} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\} \subset L.$$

Now, consider some $\mathbf{z} \in L$. Since $\mathbf{b}_0, ...\mathbf{b}_{n-1}$ are linearly independent, they generate $\mathbb{R}^n$. Hence, we can write

$$\mathbf{z} = \sum_{i=0}^{n-1} x_i \mathbf{b}_i$$

for some $x_0, ..., x_{n-1} \in \mathbb{R}$. We will show that, in fact, $x_0, ...x_{n-1} \in \mathbb{Z}$. Consider $\mathbf{z}' = \sum_{i=0}^{n-1} \lfloor x_i \rfloor \mathbf{b}_i \in L$. By closure,

$$\mathbf{z} - \mathbf{z}' = \sum_{i=0}^{n-1} (x_i - \lfloor x_i \rfloor) \mathbf{b}_i \in L.$$

Now, consider the distance

$$\text{dist}(\{\mathbf{z} - \mathbf{z}'\}, \mathcal{S}(\{\mathbf{b}_0, ..., \mathbf{b}_{n-2}\}).$$

Notice that this distance equals $(x_{n-1} - \lfloor x_{n-1} \rfloor)\|\mathbf{b}^*_{n-1}\|$, which is the orthogonal component of the vector from $\mathbf{z} - \mathbf{z}'$ to the closest point in $\mathcal{S}(\{\mathbf{b}_0, ..., \mathbf{b}_{n-2}\})$. Thus, we have that

$$\text{dist}(\{\mathbf{z} - \mathbf{z}'\}, \mathcal{S}(\{\mathbf{b}_0, ..., \mathbf{b}_{n-2}\}) = (x_{n-1} - \lfloor x_{n-1} \rfloor)\|\mathbf{b}^*_{n-1}\|.$$

Similarly,

$$\text{dist}(\{\mathbf{b}_{n-1}\}, \mathcal{S}(\{\mathbf{b}_0, ..., \mathbf{b}_{n-2}\}) = \|\mathbf{b}^*_{n-1}\|.$$

Since $0 \leq x_{n-1} - \lfloor x_{n-1} \rfloor < 1$, we have that

$$(x_{n-1} - \lfloor x_{n-1} \rfloor)\|\mathbf{b}^*_{n-1}\| < \|\mathbf{b}^*_{n-1}\|.$$

However, recall that by construction $\mathbf{b}_{n-1}$ is the closest member of L to $\mathcal{S}(\{\mathbf{b}_0, ..., \mathbf{b}_{n-2})$. Therefore, $x_{n-1} - \lfloor x_{n-1} \rfloor = 0$ and so $x_{n-1} \in \mathbb{Z}$. We repeat this strategy for all $x_{n-2}, ...x_0$ to find that all the coefficients x_i are integers. Thus, $\mathbf{z} \in \left\{ \sum_{i=0}^{n-1} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}$ and so $L \subset \left\{ \sum_{i=0}^{n-1} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}$, as desired. $\square$

We will primarily use the linear algebraic definition of lattices going forward.

3.4 Computational Problems

Any mathematical problem that is very difficult to solve without a crucial piece of information makes a good candidate for a cryptosystem. Lattice-based cryptosystems are based on two optimization problems thought to be intractable, namely the Closest Vector Problem (CVP) and the Shortest Vector Problem (SVP). The crux of the CVP is to find the lattice vector closest to a given vector. The crux of the SVP is to find the shortest vector of a given lattice, i.e. the to find the lattice vector whose length is $\lambda_0(\mathcal{L})$. We first introduce the two problems formally, and then we discuss their connection to the lattice properties which we have introduced.

Definition 3.4.1. Given a lattice $\mathcal{L}(\mathcal{B})$ and a vector $\mathbf{w} \in \mathbb{R}^n$, the *Closest Vector Problem (CVP)* for $(\mathcal{L}, \mathbf{w})$ is to find some $\mathbf{v} \in \mathcal{L}$ that minimizes $\|\mathbf{v} - \mathbf{w}\|$.

Definition 3.4.2. Given a lattice $\mathcal{L}(\mathcal{B})$, the *Shortest Vector Problem (SVP)* for $\mathcal{L}$ is to find some nonzero $\mathbf{v} \in \mathcal{L}$ that minimizes $\|\mathbf{v}\|$.

Note that there may be multiple shortest vectors or closest vectors, which is why the problems ask not for *the* vector but for *some* vector. The SVP can be considered to be a variant of the CVP, as in the SVP we are tasked with finding the closest nonzero lattice vector to the origin. The SVP varies from the CVP in this regard only by the inclusion of the word "nonzero", as solving the CVP given the origin yields the origin itself. These two problems are considered to be extremely computationally difficult, and their difficulty grows with the dimension of the given lattice. In fact, the CVP has been shown to be $\mathcal{NP}$-difficult [5]. The SVP has been shown to be no harder than the CVP [6], though under certain conditions it also is $\mathcal{NP}$-difficult [1]. However, as the dimension grows, it is also harder to form an efficient cryptosystem based on these problems as it becomes increasingly computationally taxing to encrypt information. Hence, in practice, we are usually more interested in approximation variants of the SVP and CVP.

A naive yet straightforward approach to finding a shortest vector in the lattice is determining the shortest basis vector. This approach tends to have good results when dealing with a "nice" basis, i.e. an orthogonal or near-orthogonal basis. However, there is no guarantee that the approach will yield a satisfactory result when dealing with a "bad" basis, i.e. a far from orthogonal basis.

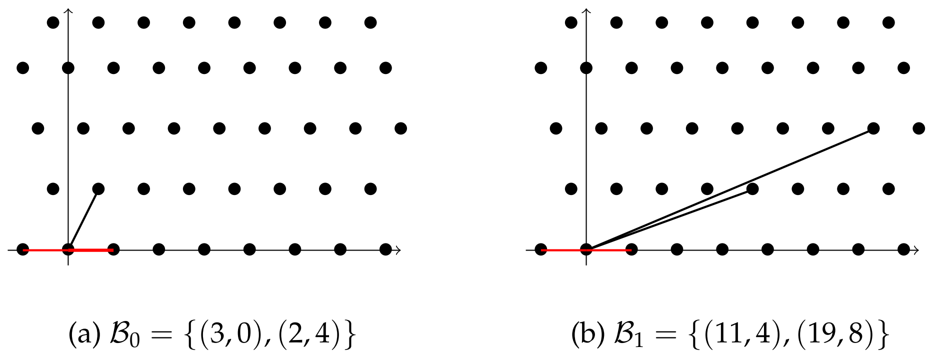
(a) $\mathcal{B}_0 = \{(3,0),(2,4)\}$ (b) $\mathcal{B}_1 = \{(11,4),(19,8)\}$

Figure 4: The set of shortest vectors (marked in red) from the origin of the lattice generated by $\mathcal{B}_0$ includes the shortest basis vector $(3,0)$. The same cannot be said for $\mathcal{B}_1$.

Example 3.4.3. Consider a lattice generated by $\mathcal{B}_0 = \{(3,0),(2,4)\}$. The Hadamard ratio for $\mathcal{B}_0$ is $\frac{\|(3,0)\|\|(2,4)\|}{12} = \frac{3\times 2\sqrt{5}}{12} \approx 1.12$. Though this basis is not orthogonal, it is close enough to being orthogonal that the shortest basis vector, $(3,0)$, is a shortest lattice vector along with $(-3,0)$.

Example 3.4.4. The same lattice as in the previous example can be generated by $\mathcal{B}_1 = \{(11,4),(19,8)\}$, since

$$\begin{aligned} 2(11,4) - (19,18) &= (3,0), \\ -5(11,4) + 3(19,8) &= (2,4), \\ 3(3,0) + (2,4) &= (11,4), \\ 5(3,0) + 2(2,4) &= (19,8). \end{aligned}$$

However, the Hadamard ratio for $\mathcal{B}_1$ is $\frac{\|(11,4)\|\|(19,8)\|}{40} = \frac{\sqrt{137}\times\sqrt{425}}{40} \approx 6.03$. The set of shortest vectors here does not contain the shortest basis vector, $(11,4)$. So, computing the shortest vector of $\mathcal{L}(\mathcal{B}_1)$ is a more complex problem than simply surveying the basis for a vector of minimum length.

See Figure 4 for an illustration of the preceding examples. We have begun to see how having a ”nice” basis would be beneficial for breaking a cryptosystem based on the SVP.

In the following chapters, we discuss algorithms aimed at solving the SVP and the CVP. We also develop an understanding of the desirability for orthogonal bases.

4 Babai's Closest Vertex Algorithm

The most common approach to solve the CVP is to use Babai's Closest Vertex Algorithm. In this chapter we introduce the algorithm and consider examples that illustrate how the success of the algorithm is contingent on the choice of basis. This chapter draws considerably from Section 7.6 in [7].

Suppose we want to solve the CVP for a given lattice $\mathcal{L}(\mathcal{B})$ and vector $\mathbf{w} \in \mathbb{R}^n$, where $\mathcal{B}$ is orthogonal. Since $\mathcal{B}$ spans $\mathbb{R}^n$, we can write

$$\mathbf{w} = \sum_{i=0}^{n-1} a_i \mathbf{b}_i$$

for some $a_0, a_1, ..., a_{n-1} \in \mathbb{R}$. Then, for any lattice vector $\mathbf{v} = \sum_{i=0}^{n-1} x_i \mathbf{b}_i$, it follows that

$$\|\mathbf{v} - \mathbf{w}\|^2 = \sum_{i=0}^{n-1} (x_i - a_i)^2 \|\mathbf{b}_i\|^2.$$

We can see that to solve the CVP for $(\mathcal{L}, \mathbf{w})$, we must minimize the sum of $(x_i - a_i)^2$ over all i. Since each x_i is an integer, each individual difference $|x_i - a_i|$ is minimized if $x_i = \lfloor a_i \rceil$, where we denote the nearest integer to a_i by $\lfloor a_i \rceil$. Recall that, by Theorem 3.2.8, the translates of $\mathcal{F}(\mathcal{B})$ by lattice vectors tile $\mathbb{R}^n$. So, $\mathbf{w}$ is contained in some unique translate $\mathcal{F}(\mathcal{B}) + \mathbf{z} \subset \mathbb{R}^m$. The vertex of the translate $\mathcal{F}(\mathcal{B}) + \mathbf{z}$ closest to $\mathbf{w}$ is a candidate for a solution to the CVP. See Figure 5 for an illustration.

The method described above tends to be successful only when the given basis is "nice", i.e. orthogonal or near-orthogonal. The method does not work when the given basis is "bad", i.e. far from orthogonal.

Definition 4.0.1. For any lattice $\mathcal{L}(\mathcal{B})$ and vector $\mathbf{w} \in \mathbb{R}^n$, *Babai's Closest Vertex Algorithm* is as follows:

```
Find $\{a_i\} \subset \mathbb{R}$ such that $\mathbf{w} = \sum_{i=0}^{n-1} a_i \mathbf{b}_i$.
Loop $i = 0, ..., n-1$.
    Set $x_i = \lfloor a_i \rceil$.
End Loop.
Return $\mathbf{v} = \sum_{i=0}^{n-1} x_i \mathbf{b}_i$.
```

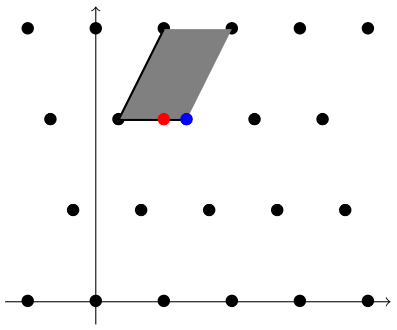

Figure 5: For the target vector $(3,8)$, the vertex $(4,8)$ of the unique translate $\mathcal{F}(\mathcal{B}) + (1,8)$ is our candidate to solve the CVP.

We consider two examples of the algorithm demonstrating varying levels of success. See Figure 6 for an illustration.

Example 4.0.2. Suppose we are given a basis $\mathcal{B}_0 = \{(3,0),(2,4)\}$ and a vector $\mathbf{w} = (3,8)$. Recall from Example 2.2.3 that the Hadamard ratio for $\mathcal{B}_0$ is approximately 1.12. We wish to solve the CVP for $(\mathcal{L}(B_0), \mathbf{w})$. We can express $\mathbf{w}$ as a linear combination $(3,8) = a_1(3,0) + a_2(2,4)$. We can solve the system to find $a_1 = -\frac{1}{3}$, $a_2 = 2$ and thus $x_1 = 0$, $x_2 = 2$. This yields the lattice vector $\mathbf{v} = 0(3,0) + 2(2,4) = (4,8)$. We find that $\|\mathbf{v} - \mathbf{w}\| = 1$. Given a near orthogonal basis, we have found the closest lattice vector to $\mathbf{w}$.

Example 4.0.3. Suppose we are given a basis $\mathcal{B}_1 = \{(11,4),(19,8)\}$ and a vector $\mathbf{w} = (3,8)$. By Example 3.4.4, $\mathcal{B}_1$ generates the same lattice as $\mathcal{B}_0$ did in Example 4.0.2. Recall from Example 2.2.4 that the Hadamard ratio for $\mathcal{B}_0$ is approximately 6.03. We wish to solve the CVP for $(\mathcal{L}(B_1), \mathbf{w})$. We can express $\mathbf{w}$ as a linear combination $(3,8) = a_1(11,4) + a_2(19,8)$. We can solve the system to find $a_1 = -\frac{32}{3}$, $a_2 = \frac{19}{3}$ and thus $x_1 = -11$, $x_2 = 6$. This yields the lattice vector $\mathbf{v} = -11(11,4) + 6(19,8) = (-7,4)$. We find that $\|\mathbf{v} - \mathbf{w}\| = \sqrt{116} \approx 10.77$. Given a basis that is far from being orthogonal, we have found a lattice vector quite far from $\mathbf{w}$.

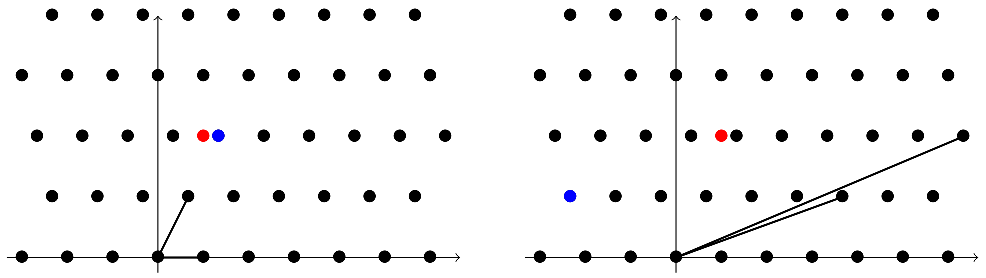
(a) $\mathcal{B}_0 = \{(3,0),(2,4)\}$ (b) $\mathcal{B}_1 = \{(11,4),(19,8)\}$

Figure 6: Target vectors marked red and the output of Babai's algorithm marked blue. The algorithm works well for near orthogonal bases like $\mathcal{B}_1$ but poorly for far from orthogonal bases like $\mathcal{B}_1$.

Let us discuss the intuition behind this variability in the success of Babai's Closest Vertex Algorithm. The algorithm works by first finding the translate of $\mathcal{F}(\mathcal{B})$ containing the given vector $\mathbf{w}$. Call this translate $\mathcal{F} + \mathbf{t}$. Next, it finds the closest vertex of $\mathcal{F} + \mathbf{t}$ to $\mathbf{w}$. When $\mathcal{B}$ is "nice", the vertices of $\mathcal{F}(\mathcal{B})$ are relatively close to each other. So, the closest lattice vector to $\mathbf{w}$ must be one of the vertices of $\mathcal{F} + \mathbf{t}$. When $\mathcal{B}$ is "bad", however, the vertices of $\mathcal{F}(\mathcal{B})$ are relatively far from each other; thus, the algorithm overlooks several lattice vectors in its attempt to find the closest vertex of $\mathcal{F} + \mathbf{t}$ to $\mathbf{w}$. In our working example, given the "bad" basis $\{(11,4),(9,8)\}$, the algorithm overlooks vectors such as $(4,8)$, $(1,8)$, $(2,4)$, $(-1,4)$, et cetera, that are close to $(3,8)$ and settles for the translate's closest vertex $(-7,4)$.

From our discussion up to now, we have seen that a "nice" basis is desirable when attempting to solve the SVP or the CVP. The following chapters will focus on algorithms that aim to replace a given basis with a more suitable one; we know that it is possible to do so and still generate the same lattice by Theorem 3.2.3.

5 Lagrange's Reduction Algorithm

There are many lattice reduction algorithms, all aiming to transform any given basis into a "nice" basis. Each algorithm has its own definition of what constitutes a "nice" basis, but a common goal is to produce basis vectors which are short and near orthogonal. In two dimensions, the algorithm for finding a lattice's optimal basis is due to Lagrange. It is often misattributed to Gauss, who described the same procedure later. In this section, we will present Lagrange's Reduction Algorithm along with an example. This will be an appropriate introduction for the next chapter, which will discuss a generalization of Lagrange's Reduction Algorithm to higher dimensions: the LLL algorithm.

Recall the Gram-Schmidt Algorithm which we introduced in Definition 3.1.2. One might wonder why we don't simply utilize the Gram-Schmidt Algorithm if we wish to have an orthogonalized basis. After all, while we are looking for algorithms to return a near-orthogonal basis, here lies an algorithm which returns an orthogonal basis. The issue with using the Gram-Schmidt Algorithm is that the projection coefficients $\frac{\langle \mathbf{b}_{\mathbf{i-1}}^*, \mathbf{b}_\mathbf{i} \rangle}{\langle \mathbf{b}_{\mathbf{i-1}}^*, \mathbf{b}_{\mathbf{i-1}}^* \rangle}$ are not necessarily integers. Thus, the resulting vectors in $\mathcal{B}^*$ do not necessarily lie in the given lattice.

Since the Gram-Schmidt Algorithm is not enough when dealing with lattices, we instead focus here on the Lagrange Reduction Algorithm. This algorithm produces what we call a Lagrange-reduced basis, which is the first type of a "nice" basis that we consider.

Definition 5.0.1. A basis $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1\} \subset \mathbb{R}^2$ is *Lagrange-reduced* if it satisfies the following conditions:

1. $\|\mathbf{b}_0\| \leq \|\mathbf{b}_1\|$.
2. $\frac{|\langle \mathbf{b}_1, \mathbf{b}_0 \rangle|}{\langle \mathbf{b}_0, \mathbf{b}_0 \rangle} \leq \frac{1}{2}$.

We can refer to these two properties as saying that the basis is *ordered* and *near-orthogonal*, respectively. Note that the fraction in the second property is the coefficient in $\mu_{\mathbf{b}_0}(\mathbf{b}_1)$, As the basis vectors become closer to being orthogonal, this coefficient gets smaller. In fact, when the vectors are orthogonal, this coefficient equals 0.

A Lagrange-reduced basis is handy not just for rendering Babai's Closest Vertex Algorithm useful, but also for solving the SVP in dimension 2.

Theorem 5.0.2. *If a basis $\mathcal{B} \subset \mathbb{R}^2$ is Lagrange-reduced, then $\mathbf{b}_0$ is a shortest vector of $\mathcal{L}(\mathcal{B})$.*

Proof. Proof adapted from Proposition 7.66 of [7].

Let $\mathcal{B} \subset \mathbb{R}^2$ be a Lagrange-reduced basis. Then, we have that

$$\|\mathbf{b}_0\| \leq \|\mathbf{b}_1\|, \tag{1}$$
$$\frac{|\langle \mathbf{b}_0, \mathbf{b}_1 \rangle|}{\langle \mathbf{b}_0, \mathbf{b}_0 \rangle} \leq \frac{1}{2}. \tag{2}$$

Let $\mathbf{v} \in \mathcal{L}(\mathcal{B})$ such that $\|\mathbf{v}\| = \lambda_0(\mathcal{L}(\mathcal{B}))$ be given. This means that $\mathbf{v}$ is a shortest vector of $\mathcal{L}(\mathcal{B})$. (Recall that a lattice may have multiple shortest vectors.) We will show that $\|\mathbf{b}_0\| = \|\mathbf{v}\|$.

We can express the given shortest vector as $\mathbf{v} = x_0\mathbf{b}_0 + x_1\mathbf{b}_1$ for some integers x_0, x_1. Since $\mathcal{B}$ is Lagrange-reduced, it follows that

$$\begin{aligned}
\|\mathbf{v}\|^2 &= \|x_0\mathbf{b}_0 + x_1\mathbf{b}_1\|^2 \\
&= x_0^2\|\mathbf{b}_0\|^2 + x_1^2\|\mathbf{b}_1\|^2 + 2x_0x_1\langle \mathbf{b}_0, \mathbf{b}_1 \rangle \\
&\geq x_0^2\|\mathbf{b}_0\|^2 + x_1^2\|\mathbf{b}_1\|^2 - 2|x_0x_1|\langle \mathbf{b}_0, \mathbf{b}_1 \rangle \\
&\geq x_0^2\|\mathbf{b}_0\|^2 + x_1^2\|\mathbf{b}_1\|^2 - |x_0x_1|\langle \mathbf{b}_0, \mathbf{b}_0 \rangle \quad \text{by (2)} \\
&\geq x_0^2\|\mathbf{b}_0\|^2 + x_1^2\|\mathbf{b}_0\|^2 - |x_0x_1|\langle \mathbf{b}_0, \mathbf{b}_0 \rangle \quad \text{by (1)} \\
&= (x_0^2 + x_1^2 - |x_0x_1|)\|\mathbf{b}_0\|^2.
\end{aligned}$$

Note that $x_0^2 + x_1^2 - |x_0x_1| = 0$ only when $x_0 = x_1 = 0$. Since x_0, x_1 are integers, it follows that

$$(x_0^2 + x_1^2 - |x_0x_1|)\|\mathbf{b}_0\|^2 \geq \|\mathbf{b}_0\|^2.$$

Thus, $\|\mathbf{v}\|^2 \geq \|\mathbf{b}_0\|^2$. Since $\mathbf{b}_0 \in \mathcal{L}(\mathcal{B})$ and $\mathbf{v}$ is a shortest vector of the lattice, it must be that $\|\mathbf{v}\| = \|\mathbf{b}_0\|$. Therefore, $\mathbf{b}_0$ is a shortest vector of $\mathcal{L}(\mathcal{B})$. □

Thus, finding a Lagrange-reduced basis for a given lattice in dimension 2 can solve the SVP.

The time is ripe to examine Lagrange's Reduction Algorithm. Any basis for a subset of $\mathbb{R}^2$ consists of two basis vectors. The idea behind Lagrange's Reduction Algorithm is to create a Lagrange-reduced basis by subtracting multiples of one basis vector from the other basis vector until no improvement is possible. By Theorem 3.2.3, multiple bases can generate the same lattice. That result ensures the validity of Lagrange's Reduction

Algorithm, which improves a given basis and produces another basis that generates the same lattice. There are two main steps in the algorithm: a *swap* step and a *reduction* step, which ensure that the output of the algorithm abides by the two properties of a Lagrange-reduced basis.

Definition 5.0.3. For any lattice $\mathcal{L}(\mathcal{B}) \subset \mathbb{R}^2$, *Lagrange's Reduction Algorithm* is as follows:

```
Set $\mathbf{b}_0' = \mathbf{b}_0, \mathbf{b}_1' = \mathbf{b}_1$.
Loop
    If $\|\mathbf{b}_0'\| > \|\mathbf{b}_1'\|$
        Swap $\mathbf{b}_0', \mathbf{b}_1'$.
    Compute $m = \left\lfloor \frac{\langle \mathbf{b}_0', \mathbf{b}_1' \rangle}{\langle \mathbf{b}_0', \mathbf{b}_0' \rangle} \right\rceil$.
    If $m = 0$
        Break Loop.
    Set $\mathbf{b}_1' = \mathbf{b}_1' - m\mathbf{b}_0'$.
Return $\mathcal{B}' = \{\mathbf{b}_0', \mathbf{b}_1'\}$.
```

For a proof for the termination of the algorithm, refer to Proposition 3.1 in [20] . The key point to note is that the value $\min\{\|\mathbf{b}_0, \mathbf{b}_1\|\}$ decreases note is that on each iteration of the algorithm, and so the projection coefficient (which is the length of the projection) also decreases and is eventually rounded to the nearest integer 0.

Notice that Lagrange's Reduction Algorithm basis is Lagrange-reduced. Swapping ensures that $\|\mathbf{b}_0'\| < \|\mathbf{b}_1'\|$, and reducing $\mathbf{b}_1'$ by $m\mathbf{b}_0'$ ensures that $\frac{|\langle \mathbf{b}_0', \mathbf{b}_1' \rangle|}{\langle \mathbf{b}_0', \mathbf{b}_0' \rangle} \leq \frac{1}{2}$. Let us now consider an example.

Example 5.0.4. Suppose we are given a basis $\mathcal{B} = \{(11, 4), (19, 8)\}$. We apply Lagrange's Reduction Algorithm to $\mathcal{B}$.

1. We set $\mathbf{b}_0' = (11, 4), \mathbf{b}_1' = (19, 8)$.

2. Since $\|\mathbf{b}_0'\| = \|\sqrt{137}\| < \|\mathbf{b}_1'\| = \sqrt{425}$, we do not swap.

3. We compute $m = \lfloor \frac{209+32}{137} \rceil = 2$. So, we set

$$\mathbf{b}_1' = (19, 8) - 2(11, 4) = (-3, 0).$$

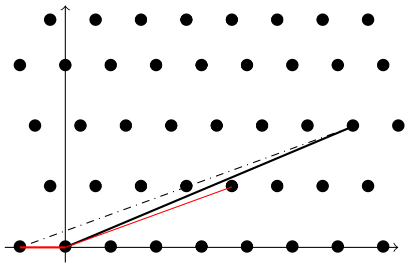

Figure 7: The lattice generated by $\{(11,4),(19,8)\}$. In Step 3, the shorter basis vector $(11,4)$ is subtracted twice from the longer basis vector $(19,8)$ to yield the vector $(-3,0)$. The new values are $\mathbf{b}_0^{'} = (11,4), \mathbf{b}_1^{'} = (-3,0)$, marked in red.

4. Since $\|\mathbf{b}_0^{'}\| = \|\sqrt{137}\| > \|\mathbf{b}_1^{'}\| = \sqrt{9}$, we swap. Now,
$$\mathbf{b}_0^{'} = (-3,0), \mathbf{b}_1^{'} = (11,4).$$
5. We compute $m = \lfloor \frac{-33+0}{9} \rceil = -4$. So, we set
$$\mathbf{b}_1^{'} = (11,4) - (-4(-3,0)) = (-1,4).$$

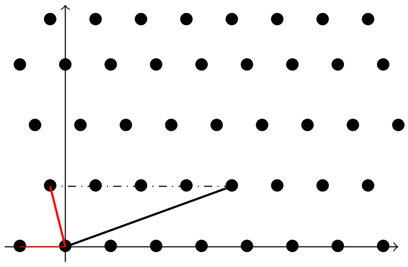

Figure 8: The lattice generated by $\{(-3,0),(11,4)\}$. Note that the lattice is invariant throughout the algorithm despite the changing basis vectors. In Step 5, the shorter basis vector $(-3,0)$ is subtracted four times from the longer basis vector $(11,4)$ to yield the vector $(-1,4)$. The new values are $\mathbf{b}_0^{'} = (-3,0), \mathbf{b}_1^{'} = (-1,4)$, marked in red.

6. Since $\|\mathbf{b}_0^{'}\| = \|\sqrt{9}\| < \|\mathbf{b}_1^{'}\| = \sqrt{17}$, we do not swap.
7. We compute $m = \lfloor \frac{3+0}{9} \rceil = 0$. So, we return $\mathcal{B}^{'} = \{(-3,0),(-1,4)\}$.

Note that $(-3, 0)$ is a shortest vector of $\mathcal{L}(\mathcal{B})$, as we demonstrated in Example 3.2.3. Thus, the algorithm easily solves the SVP for the given lattice in dimension 2.

Lagrange's Reduction Algorithm is an important introduction to the LLL Algorithm, which can be thought of a rough generalization of the two dimensional algorithm to higher dimensions. In practice, lattice cryptosystems in dimension 2 are easy to break by applying Lagrange's Reduction Algorithm. This makes lattices in dimension 2 highly impractical. It is far more common to encounter lattices in higher dimensions. So, we reserve a deeper algorithmic analysis of time complexity, applications, etc. for the LLL Algorithm.

6 The LLL Algorithm

As the dimension n increases, it becomes increasingly challenging to solve the SVP. It is unclear how to construct a perfect analogue of the Lagrange Reduction Algorithm in higher dimensions because of several issues. Firstly, it is uncertain what the right combination of preceding vectors $\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_{i-1}$ is with which to reduce $\mathbf{b}_i$; this is straightforward to do in the dimension 2 case since for any basis vector there is only one other basis vector by which it can be reduced. Secondly, in higher dimensions it requires many more computations to order a basis as each basis vector must be compared with multiple vectors.

In 1982, Lenstra, Lenstra, and Lovász published a new basis reduction algorithm [10]. Their algorithm, which we call the LLL Algorithm, was originally aimed at factoring polynomials over $\mathbb{Q}$. The algorithm has many applications in combinatorics and number theory, such as approximating the minimal polynomial of an algebraic number, and integer programming with a fixed number of variables [11]. In cryptography, there are many known LLL-based attacks on knapsack cryptosystems and RSA cryptosystems [13].

In this chapter, we discuss the algorithm's usefulness with for solving the SVP and CVP for lattices. First, we introduce the concept of an LLL-reduced basis. Then, we present the algorithm. Finally, we provide an example and analyze the algorithm's complexity.

Recall that we denote the i-th basis vector as $\mathbf{b}_i$ and the i-th Gram-Schmidt vector as $\mathbf{b}_i^*$.

Definition 6.0.1. For $\frac{1}{4} \leq \delta < 1$, a basis $\mathcal{B}$ is *δ-LLL-reduced* if it satisfies the following conditions:

1. $\delta \|\mathbf{b}_{i-1}^*\|^2 \leq \left\| \frac{|\langle \mathbf{b}_{i-1}^*, \mathbf{b}_i \rangle|}{\langle \mathbf{b}_{i-1}^*, \mathbf{b}_{i-1}^* \rangle} \mathbf{b}_{i-1}^* + \mathbf{b}_i^* \right\|^2$ for $1 \leq i < n$.

2. $\frac{|\langle \mathbf{b}_j^*, \mathbf{b}_i \rangle|}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \leq \frac{1}{2}$ for $1 \leq j < i \leq n$.

Similar to the property of a Lagrange-reduced basis, the second property ensures reduction of basis vectors, and we refer to this property as saying that the basis is *near-orthogonal*. For convenience, we generally use $\delta = \frac{3}{4}$ and refer to a $\frac{3}{4}$-LLL-reduced basis as an LLL-reduced basis. The value of δ plays a crucial role in the ordering of the basis. The first property, which we call the *Lovász condition*, serves to order the basis. δ acts as a scaling factor, ordering the basis not based on the lengths of vectors but on the scaled lengths of vectors. The scaling serves to relax the Lovász condition, making it easier for a basis to qualify as LLL-reduced. So, to satisfy the Lovász condition, a basis need not be perfectly ordered, i.e. a basis does not need to have the property that $\|\mathbf{b}_0\| \leq ... \leq \|\mathbf{b}_{n-1}\|$. This allows an algorithm which transforms a basis into an LLL-reduced basis to be more efficient as it only needs to check a more relaxed set of criteria. As the value of δ increases, the Lovász condition becomes harder to meet. We do not allow $\delta = 1$ even though this would ensure perfect ordering; as we will see later, such a value of δ is thoroughly impractical. To see more readily the ordering present in an LLL-reduced basis, observe that from the Lovász condition it follows that

$$\begin{aligned}\frac{3}{4}\|\mathbf{b}_{i-1}^*\|^2 &\leq \left\| \frac{|\langle \mathbf{b}_{i-1}^*, \mathbf{b}_i \rangle|}{\langle \mathbf{b}_{i-1}^*, \mathbf{b}_{i-1}^* \rangle} \mathbf{b}_{i-1}^* + \mathbf{b}_i^* \right\|^2 \\ &= \left| \frac{\langle \mathbf{b}_{i-1}^*, \mathbf{b}_i \rangle}{\langle \mathbf{b}_{i-1}^*, \mathbf{b}_{i-1}^* \rangle} \right|^2 \|\mathbf{b}_{i-1}^*\|^2 + \|\mathbf{b}_i^*\|^2,\end{aligned}$$

with the equality resulting from the orthogonality of $\mathbf{b}_{i-1}^*$ and $\mathbf{b}_i^*$. Hence,

$$\begin{aligned}\|\mathbf{b}_i^*\|^2 &\geq \left(\frac{3}{4} - \left| \frac{\langle \mathbf{b}_{i-1}^*, \mathbf{b}_i^* \rangle}{\langle \mathbf{b}_{i-1}^*, \mathbf{b}_{i-1}^* \rangle} \right|^2\right) \|\mathbf{b}_{i-1}^*\|^2 \\ &> \frac{1}{2}\|\mathbf{b}_{i-1}^*\|^2.\end{aligned}$$

This shows how, in Regev's words, the Lovász condition provides that each vector $\mathbf{b}_i^*$ is not much shorter than the preceding vector $\mathbf{b}_{i-1}^*$ [18]. It is easy to verify that the statement

$$\|\mathbf{b}_i^*\|^2 \geq \left(\frac{3}{4} - \left| \frac{\langle \mathbf{b}_{i-1}^*, \mathbf{b}_i^* \rangle}{\langle \mathbf{b}_{i-1}^*, \mathbf{b}_{i-1}^* \rangle} \right|^2\right) \|\mathbf{b}_{i-1}^*\|^2$$

implies the Lovász condition as stated in Definition 6.0.1. The ordering given by the Lovász condition is desirable for solving the SVP. We now introduce a theorem which gives an understanding of an LLL-reduced basis's desirability.

Theorem 6.0.2 (Proposition 1.11 in [10]). *Let $\mathcal{B}$ be an LLL-reduced basis. Then,*

$$\|\mathbf{b}_0\| \leq 2^{(n-1)/2}\|\mathbf{v}\|$$

for any $\mathbf{v} \in \mathcal{L}(\mathcal{B})$.

Proof. Proof adapted from Proposition 1.11 in [10]. First, we verify a lemma.

Lemma 6.0.3. Let $\mathcal{B}$ be an LLL-reduced basis. Then, $\|\mathbf{b}_j\|^2 \leq 2^i \|\mathbf{b}_i^*\|^2$ for all $0 \leq j \leq i \leq n-1$.

Proof of Lemma. We adapt the inductive argument from [10].

Let $\mathcal{B}$ be an LLL-reduced basis. By induction, we will show that

$$\|\mathbf{b}_j^*\|^2 \leq 2^{i-j} \|\mathbf{b}_i^*\|^2$$

for all $0 \leq j \leq i \leq n-1$. For the base case, let $j = 0$. Then, we see that

$$\|\mathbf{b}_0^*\|^2 \leq 2\|\mathbf{b}_1^*\|^2 \leq ... \leq 2^{n-1}\|\mathbf{b}_{n-1}^*\|^2.$$

Now, suppose that

$$\|\mathbf{b}_k^*\|^2 \leq 2\|\mathbf{b}_{k+1}^*\|^2 \leq ... \leq 2^{n-1-k}\|\mathbf{b}_{n-1}^*\|^2.$$

It follows immediately that

$$\|\mathbf{b}_{k+1}^*\|^2 \leq 2\|\mathbf{b}_{k+2}^*\|^2 \leq ... \leq 2^{n-k-2}\|\mathbf{b}_{n-1}^*\|^2.$$

Thus, $\|\mathbf{b}_j^*\|^2 \leq 2^{i-j}\|\mathbf{b}_i^*\|^2$ for all $0 \leq j \leq i \leq n-1$.

Let $0 \leq j \leq i \leq n-1$. By the Gram-Schmidt Algorithm,

$$\|\mathbf{b}_i\|^2 \leq \|\mathbf{b}_i^*\|^2 + \sum_{j=0}^{i-1} \frac{\langle \mathbf{b}_j^*, \mathbf{b}_i \rangle^2}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle^2} \|\mathbf{b}_j^*\|^2.$$

By the size condition of an LLL-reduced basis,

$$\begin{aligned} \|\mathbf{b}_i^*\|^2 + \sum_{j=0}^{i-1} \frac{\langle \mathbf{b}_j^*, \mathbf{b}_i \rangle^2}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle^2} \|\mathbf{b}_j^*\|^2 &\leq \|\mathbf{b}_i^*\|^2 + \sum_{j=0}^{i-1} \frac{2^{i-j}}{4} \|\mathbf{b}_i^*\|^2 \\ &= (1 + \frac{2^i - 2}{2}) \|\mathbf{b}_i^*\|^2 \\ &\leq 2^i \|\mathbf{b}_i^*\|^2. \end{aligned}$$

Hence,

$$\|\mathbf{b}_j\|^2 \leq 2^j \|\mathbf{b}_j^*\|^2 \leq 2^i \|\mathbf{b}_i^*\|^2$$

for all $0 \leq j \leq i \leq n-1$, as desired. □

We continue with the proof of Theorem 6.0.2. Consider any arbitrary nonzero $\mathbf{v} \in \mathcal{L}(\mathcal{B})$ and write

$$\begin{aligned}\mathbf{v} &= \sum_{i=0}^{n-1} x_i \mathbf{b}_i \\ &= \sum_{i=0}^{n-1} y_i \mathbf{b}_i^*\end{aligned}$$

for some $x_i \in \mathbb{Z}, y_i \in \mathbb{R}$. Let k be the largest index such that $x_k \neq 0$. Then, by the Gram-Schmidt Algorithm, $x_k = y_k$. So, since y_k is a nonzero integer,

$$\|\mathbf{b}_k^*\|^2 \leq y_k^2 \|\mathbf{b}_k^*\|^2 \leq \|\mathbf{v}\|^2.$$

Hence,

$$2^k \|\mathbf{b}_k^*\|^2 \leq 2^k \|\mathbf{v}\|^2 \leq 2^{n-1} \|\mathbf{v}\|^2.$$

Then, by the lemma,

$$\|\mathbf{b}_0\|^2 \leq 2^k \|\mathbf{b}_k^*\|^2 \leq 2^{n-1} \|\mathbf{v}\|^2$$

and so

$$\|\mathbf{b}_0\| \leq 2^{(n-1)/2} \|\mathbf{v}\|,$$

as desired. □

Thus, finding an LLL-reduced basis for a lattice finds an approximate solution to the SVP within a factor of $2^{(n-1)/2}$. The LLL Algorithm, which reduces a given basis into an LLL-reduced basis, is thus an approximation algorithm for the SVP.

6.1 The Algorithm

We now describe the LLL Algorithm. There are a few moving parts to keep track of. There are three objects which we need to follow.

1. $\mathcal{B}$, the given set of basis vectors which we reduce and eventually return upon the algorithm's termination.

2. $\mathcal{B}^*$, the set of Gram-Schmidt vectors.

3. k, which we call the *working index*. This value loops through the cardinality of the basis. The corresponding basis vector $\mathbf{b}_k$, which we call the *working vector*, is the vector in $\mathcal{B}$ which we are currently trying to reduce.

The algorithm broadly consists of two parts.

1. Given a basis $\mathcal{B}$, we subtract from the working vector $\mathbf{b}_k$ appropriate integer multiples of the preceding vectors $\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_{k-1}$. These reductions are done in stages, and they ensure that the size condition is met.

2. We decide, based on the Lovász condition, whether the working vector is set as the next basis vector or whether it replaces the preceding basis vector. If the Lovász condition is not met, we swap the working vector with the preceding basis vector and perform reduction again.

Upon termination of the algorithm, every basis vector will have undergone reduction at least once. Though the basis vectors undergo many changes, the algorithm ensures that they always form a basis for the same lattice. Thus, when we calculate the Hadamard ratios to compare the orthogonality of the input basis to that of the returned basis, the value $\text{Vol}(\mathcal{F}(\mathcal{B})$ is the same in both calculations.

Definition 6.1.1. For any lattice $\mathcal{L}(\mathcal{B})$, the *LLL Algorithm* is as follows:

```
Compute $\mathcal{B}^*$.
Set $k = 1$.
Loop while $k \leq n - 1$.
    Loop $j = k - 1, k - 2, ..., 0$.
        Set $\mathbf{b}_k = \mathbf{b}_k - \left\lceil \frac{\langle \mathbf{b}_j^*, \mathbf{b}_k \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \right\rfloor \mathbf{b}_j$.
        As necessary, recompute $\mathcal{B}^*$.
    If $\|\mathbf{b}_k^*\|^2 \geq \left( \frac{3}{4} - \frac{\langle \mathbf{b}_{k-1}^*, \mathbf{b}_k \rangle^2}{\langle \mathbf{b}_{k-1}^*, \mathbf{b}_{k-1}^* \rangle^2} \right) \|\mathbf{b}_{k-1}^*\|^2$
        Set $k = k + 1$.
    Else
        Swap $\mathbf{b}_k, \mathbf{b}_{k-1}$.
        As necessary, recompute $\mathcal{B}^*$.
        Set $k = \max\{k - 1, 1\}$.
Return $\mathcal{B}$.
```

Observe that, if the LLL Algorithm terminates, the returned basis is LLL-reduced. The j-loop ensures that each $\mathbf{b}_k$ is reduced with regard to every preceding vector $\mathbf{b}_j$, where $0 \leq j < k$, so that the size condition is met. Also, if the algorithm terminates, we have $k = n$ which means that every basis vector will have passed the Lovász condition. An implementation of the LLL Algorithm can be accessed on GitHub at this link [8]. It is written in Python 2.7 and uses NumPy.

We provide an example of the LLL Algorithm in Appendix 8.0.1. The example features a 3-dimensional basis in order to emphasize the LLL Algorithm's application to higher dimensions.

The astute eye sees that several computations in the example could have been avoided. For example, when $k = 1$, it is unnecessary to compute $\mathbf{b}_2^*$ as $\mathbf{b}_2^*$ is not used in that pass of the algorithm. Such computations are included in the example for the untrained eye to fully understand the algorithm step-by-step, but this proves to be a naive implementation of the algorithm. An astute implementation is able to update the appropriate Gram-Schmidt vectors and projection operators as needed without recomputing $\mathcal{B}^*$ upon every change to $\mathcal{B}$. When dealing with very high dimensions, it is necessary to make smart configurations to the algorithm in order for the algorithm to terminate in a reasonable amount of time. In the following section, we discuss some concerns related to the running time and complexity of the LLL Algorithm.

6.2 Complexity Analysis

It is not immediately clear that the LLL Algorithm terminates in finitely many steps. Note that while incrementing k takes the algorithm closer to termination, k can be decremented when we set $k = \max\{k-1, 1\}$. This leads us to question whether the algorithm will ever exit the main k-loop and terminate. This question is resolved in the discussion on Theorem 7.71 of [7]. In fact, Lenstra, Lenstra, and Lovász show that the LLL Algorithm can terminate in polynomial time.

Theorem 6.2.1 (Proposition 1.26 in [10]). *For a basis $\mathcal{B}$, let $\omega > \max(\{\|\mathbf{b}_i\|^2\})$. The number of arithmetic operations needed by the LLL Algorithm on $\mathcal{B}$ is in the order of $n^4 log(\omega)$.*

Proof. See Proposition 1.26 in [10] for the full proof. The overall idea is as follows. Initialization of the algorithm takes a number of arithmetic operations in the order of n^3. Verifying the size condition requires a number of operations in the order of n. Verifying the Lovász condition takes a number of operations in the order of n^2. Since the number of times that the Lovász condition is checked is in the order of $n^2 \log(\omega)$, the number of operations taken by the algorithm is in the order of $n^3 + n + n^2(n^2 \log(\omega))$ which is equivalent to the order of $n^4 \log(\omega)$. □

Let us make a few remarks on the choice of δ in the algorithm. Note that the value of δ, which we conventionally hold to be $\frac{3}{4}$, appears in the algorithm when we check the Lovász condition. Using $\delta < 1$ allows the algorithm to run in polynomial time but can squander several opportunities for reduction, since we do not swap whenever $\|\mathbf{b}_k^*\|^2 < \|\mathbf{b}_{k-1}^*\|^2$. The closer δ is to 1, the more difficult the condition to increment the working index becomes, thus leading to a better reduced and more orthogonal output.

When we use $\delta = 1$, we refer to the LLL Algorithm as the *optimal LLL Algorithm*, since it provides the best output. Earlier, we alluded to the fact that the optimal LLL Algorithm is impractical though it ensures perfect ordering. This is so because ensuring that the basis is perfectly ordered is a very hard problem. If $\delta = 1$, then the Lovász condition is very difficult to meet; this results in the algorithm being stuck in the k-loop for a very long time. In fact, given current implementations and computing power, the optimal LLL Algorithm does not run in polynomial time. However, there is still much research yet to be done in this area. In 2003, Akhavi, for the first time, gave a proof for a linear bound for the number of iterations of the optimal LLL Algorithm.

Theorem 6.2.2 (Theorem 15 in [2]). *For a basis $\mathcal{B}$, let $\theta > \max(\{\|\mathbf{b}_i\|\})$. For all constants $C > \left(\frac{2}{\sqrt{3}}\right)^{1/6}$, the number of iterations of the optimal LLL Algorithm on $\mathcal{B}$ is in the order of $C^{n^3}\log(\theta)$.*

Proof. See Theorem 15 in [2] for the full proof. We omit the proof here. $\square$

To show that the optimal LLL Algorithm could run in polynomial time, all that remains to be done is to show that each iteration takes polynomial time. Since the number of iterations is linearly bounded, if each iteration were to take polynomial time then the entire algorithm would clearly run in polynomial time. However, it is still an open problem to find a polynomial bound for the running of each iteration for the optimal LLL Algorithm.

6.3 Applying LLL to Cryptanalysis

The LLL Algorithm has a host of applications to cryptanalysis of cryptosystems, such as GGH and NTRU, that are based on lattices. In the following section, we demonstrate the algorithm's application to knapsack cryptosystems. The algorithm easily breaks knapsack cryptosystems in low dimensions, and producing secure versions of knapsack cryptosystems requires dimensions so high that encryption becomes impractical. Thus, knapsack cryptosystems have fallen out of use.

7 Knapsack Cryptosystems

7.1 Basics on Public Key Cryptography

We quickly review a few basic ideas from public key cryptography in order to develop a basic intuition without floundering in the intricate details.

Public key cryptography is a broad system of cryptography using two sets of *keys*: a *public key* which is shared with the world and a *private key* which is kept secret to a person. Bob wishes to send a secret to Alice which only shee can read. Bob first comes up with his plaintext message and encrypts it using Alice's public key. He sends the ciphertext to Alice who uses her private key to decrypt it and recover the original message. If Eve intercepts the ciphertext on its way from Bob to Alice, she cannot decrypt the message because she does not have access to Alice's private key. In some cases, Eve is able to figure out Alice's private key if it has not been generated with due diligence.

In theory, all public key cryptosystems are susceptible to *brute force attacks* in which an eavesdropper attempts to decrypt the message by repeatedly guessing the private key. Such attacks are usually extremely inefficient. There is always research being done to construct newer, faster, and more powerful attack algorithms.

7.2 The Subset-Sum Problem

In the 1970s, Merkle and Hellman designed one of the earliest public key cryptosystems. Their cryptosystem, which we call the Merkle-Hellman cryptosystem, is based on a version of the classical knapsack problem. We first introduce the classical knapsack problem, also called the subset-sum problem.

Definition 7.2.1. Given a set of weights $W = \{w_0, w_1, ..., w_{n-1}\} \subset \mathbb{Z}^+$ and a value $s \in \mathbb{Z}$, the *subset-sum problem* is to find a subset $\{w_{i_0}, w_{i_1}, ..., w_{i_k}\}$, if one exists, such that

$$\sum_{j=0}^{k} w_{i_j} = s,$$

i.e. to find a subset of W whose elements sum to s.

An intuitive way to think of the subset-sum problem is the following: if a person is handed a knapsack of capacity s and various items each with size w_j, can they fill the knapsack perfectly with a sampling of the items? We can formulate a cryptosystem based on this problem as follows:

Suppose the set W is Alice's public key. Bob encrypts a message by first choosing a secret vector $x = (x_0, x_1, ..., x_{n-1})$ where each $x_i \in \{0,1\}$, and then sending the sum $s = \sum_{i=0}^{n-1} x_i w_i$ to Alice. Thus, Bob has used the set W as a type of alphabet, picking which letters he wishes to send by assigning each index i to a value in the set $\{0,1\}$. Alice, wishing to decrypt Bob's message, must find either x or another binary vector that yields s. By finding a binary vector yielding s, Alice can find a subset of W, choosing the integers w_j where $x_j = 1$ and discarding the integers w_j where $x_j = 0$.

Alice can employ brute force and compute every possible vector to find x. This approach requires a number of computations in the order of 2^n and is thus regarded as incredibly inefficient, though a simple observation provides a better approach that divides the exponent by 2. Consider the following, more efficient, approach, which is described by Odlyzko in [16]:

Upon receiving the sum s from Bob, Alice splits the dimension n into half, restricting her attention to the subsets of integer indices

$$I = \left\{0 \le i \le \left\lceil \frac{n-1}{2} \right\rceil \right\}, J = \left\{ \left\lceil \frac{n-1}{2} \right\rceil < j \le n-1 \right\} \subset \mathbb{Z}.$$

Alice generates the sets of sums

$$\mathcal{I} = \left\{ \sum_{i \in I} x_i w_i \mid x_i \in \{0,1\} \right\}, \mathcal{J} = \left\{ s - \sum_{j \in J} x_j w_j \mid x_j \in \{0,1\} \right\}.$$

She then looks for an element in $\mathcal{I} \cap \mathcal{J}$. Observe that an element arises in the intersection precisely when a solution to the subset-sum problem is found, since if

$$\sum_{i \in I} x_i w_i = s - \sum_{j \in J} x_j w_j$$

for some selection of $x_i, x_j \in \{0,1\}$, then

$$\begin{aligned} s &= \sum_{i \in I} x_i w_i + \sum_{j \in J} x_j w_j \\ &= \sum_{i=0}^{n-1} x_i w_i. \end{aligned}$$

This approach requires a number of computations in the order of $n2^{n/2}$, which is a significant reduction but still not close to practical. For a verification of this approach refer to Proposition 7.3 in [7]. Of course, this level of difficulty in finding the vector **x** is good for protecting a message from an eavesdropping Eve. However, it is equally difficult for the intended recipient Alice unless we assume she has staggeringly more computational power than any eavesdropper. Since this is not a practical assumption, the general subset-sum problem does not generate a tenable cryptosystem.

7.3 The Merkle-Hellman Cryptosystem

In the above discussion, what Alice needs to make decryption feasible is some secret information about W which allows her to more easily find $\mathbf{x}$. This leads us to discuss the Merkle-Hellman Cryptosystem, which is built on an easier version of the general subset-sum problem.

Definition 7.3.1. A set of positive integers $a_0, a_1, ..., a_k$ is *superincreasing* if $a_i > \sum_{j=0}^{i-1} a_j$ for all $1 \leq i \leq k$.

A simple example of a superincreasing set is $\{1, 2, 2^2, 2^3, ...\}$. Observe that if W is superincreasing, then there is a simple recursive approach for solving the subset-sum problem. Suppose Alice receives the sum $s = \sum_{i=0}^{n-1} x_i w_i$ knowing that W is superincreasing. She solves for $x_{n-1}, x_{n-2}, ..., x_0$ in that order. First, she notices that $x_{n-1} = 1$ if and only if $s > \sum_{i=0}^{n-2} w_i$. So Alice is able to find $x_i \in \{0, 1\}$ based on the sum of the preceding weights. Next, she notices that $x_{n-2} = 1$ if and only if $s - (x_{n-1}w_{n-1}) > \sum_{i=0}^{n-3} w_i$ and accordingly finds $x_{n-2} \in \{0, 1\}$. She employs this recursive approach to find the entire vector x.

The Merkle-Hellman Cryptosystem is based on the superincreasing subset-sum problem, making it easy for a recipient to decrypt a message. However, it is just as easy for an eavesdropper to decrypt a message if both the recipient and the eavesdropper have access to the same information. The Merkle-Hellman Cryptosystem avoids this issue through the use of secret modular linear transformations to disguise the superincreasing W. Eavesdroppers will see the disguised public key P and assume that it is a general knapsack, while Alice keeps the superincreasing W private. Eavesdroppers are kept busy trying to solve the very difficult subset-sum problem, while Alice applies the inverse of the secret modular linear transformations to yield a superincreasing subset-sum problem. We now describe the cryptosystem in detail. There are three major portions of the cryptosystem: key generation, encryption, and decryption.

1. Key generation: Alice begins by generating a system to allow others to send her messages.

1.1. Alice chooses a superincreasing $W = \{w_0, w_1, ..., w_{n-1}\}$ such that $w_0 > 2^n$. She also chooses positive coprime integers A, B such that $B > \sum_{i=0}^{n-1} w_i$.

1.2. Alice computes the vector $p = (p_0, p_1, ..., p_{n-1})$ where

$$p_i = Aw_i \mod B.$$

Alice shares the public key p and keeps the private key (W, A, B) secret.

2. Encryption: Bob converts a plaintext message into ciphertext, which he then sends to Alice.

2.1. A plaintext message can be easily converted to binary using ASCII encoding. Bob wishes to send a binary plaintext vector $x = (x_0, x_1, ..., x_{n-1})$, where each $x_i \in \{0, 1\}$. He computes the dot product

$$s = \langle x, p \rangle = \sum_{i=0}^{n-1} x_i p_i.$$

The sum s is the ciphertext which he sends Alice.

3. Decryption: Alice receives a ciphertext from Bob, which she converts back into the original plaintext message.

3.1. Alice receives s. Knowing A, B, she computes

$$\begin{aligned} s' &= A^{-1}s \mod B \\ &= A^{-1} \sum_{i=0}^{n-1} x_i p_i \mod B \\ &= A^{-1} \sum_{i=0}^{n-1} x_i A w_i \mod B \\ &= \sum_{i=0}^{n-1} x_i w_i \mod B. \end{aligned}$$

Here, A^{-1} refers to the multiplicative inverse of A modulo B. Now, Alice solves the subset-sum problem for s' using the superincreasing W and the recursive method outlined above. Since $\sum_{i=0}^{n-1} x_i w_i < \sum_{i=0}^{n-1} w_i < B$, which is the modulus, Alice knows that she has found an exact solution to the subset-sum problem and not a congruence. Alice recovers the vector x and reads the message sent by Bob.

Let us have a brief discussion on the complexity of this cryptosystem. Merkle and Hellman recommended applying several more modular operations to add additional layers of security; these recommendations are discussed at length by Odlyzko [16]. For a reasonable standard of security, we must insist that $w_0 > 2^n$. If $w_0 > 2^n$, it follows from the superincreasing nature of W that the sum s is in the order of 2^{2n}. This results in the public key p being quite large compared to the public keys in the RSA cryptosystem. Though the large key size is a slight inconvenience in terms of information capacity, Merkle-Hellman turns out to be significantly faster than RSA because it requires very few modular operations. Encryption requires no modular operations, and decryption requires only a few modular multiplications. Thus, Merkle-Hellman was attractive for its high speed.

However attractive the Merkle-Hellman cryptosystem was due to its speed, several doubts were raised regarding its security. Fundamental flaws in knapsack cryptosystems were uncovered in the 1980s by Shamir [19] and others, but these were mainly of theoretical interest. The LLL Algorithm provided a practical attack, which we now present.

7.4 The LLL Attack on Knapsack Cryptosystems

The following attack is described elegantly in [9]. Suppose Bob sends an encrypted message s to Alice, who has made p public but has kept (W, A, B) private. Unfortunately, an eavesdropping Eve intercepts the ciphertext. She can connect the subset-sum problem to lattice as follows. Eve forms an $(n+1) \times (n+1)$ matrix

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & p_0 \\ 0 & 1 & \cdots & 0 & p_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & p_{n-1} \\ 0 & 0 & \cdots & 0 & -s \end{bmatrix},$$

and constructs a lattice basis $\mathcal{B}$ with the rows of the matrix, assigning $\mathbf{b}_0 = (1, 0, \ldots, 0, p_0), \mathbf{b}_1 = (0, 1, \ldots, 0, p_1)$, and so on. Thus Eve generates $\mathcal{L}(\mathcal{B}) \subset \mathbb{R}^{n+1}$. Eve wants to recover the vector x. She considers the lattice vector

$$\begin{aligned} X &= \sum_{i=0}^{n-1} x_i \mathbf{b}_i - \mathbf{b}_n \\ &= (x_0, x_1, \ldots, x_n, 0). \end{aligned}$$

Note that this vector is quite short, since each $x_i \in \{0, 1\}$. So, there is a good chance that X solves the SVP for $\mathcal{L}(\mathcal{B})$. Thus, we can transform the subset-sum problem into a problem with which we are now very familiar. We can utilize the LLL Algorithm to find the shortest vector of the lattice $\mathcal{L}(\mathcal{B})$, and this vector has a high chance of being X. Lagarias and Odlyzko have found that for almost all problems where $\frac{n}{\log_2 \max\{p_i\}} < 0.645$, the vector X does indeed solve the SVP [9]. Further, they found that for almost all problems where $\frac{n}{\log_2 \max\{p_i\}} < \frac{1}{n}$, the first basis vector of the LLL-Reduced basis is in fact X. Such problems where the value $\frac{n}{\log_2 \max\{p_i\}}$ is quite low are called problems of *low density*. Even if the LLL Algorithm does not return X, Eve can apply the algorithm nonetheless and find X in the LLL-reduced basis $\mathcal{B}$. Having found X, Eve can easily recover x. Let us provide a short example in which the algorithm does not return X but includes X in the returned basis.

Example 7.4.1. Example adapted from [3].

Alice chooses the superincreasing set $W = \{2, 5, 9, 21, 45, 103, 215, 450, 946\}$ and the coprime $A = 1289, B = 2003$. These make up her private key. Notice that $2003 > 1796 = \sum_{i=0}^{8} w_i$. She computes the vector

$$\begin{aligned} p &= 1289(2, 5, 9, 21, 45, 103, 215, 450, 946) \mod 2003 \\ &= (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570), \end{aligned}$$

which she shares as her public key. Suppose Bob wishes to send Alice the binary message $x = (1, 0, 1, 1, 0, 0, 1, 1, 1)$. He encrypts the message by computing the sum $s = \langle x, p \rangle = 6665$, which he sends to Alice. Suppose this message is intercepted by the eavesdropping Eve, who wishes to recover x from s. All Eve knows is the public key p and the ciphertext s.

Eve forms the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 575 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 436 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1586 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1030 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1921 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 569 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 721 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1183 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1570 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6665 \end{bmatrix}$$

and forms $\mathcal{B}$ by letting $\mathbf{b}_0$ be the first row of the matrix, and so on. Applying the LLL Algorithm to $\mathcal{B}$ yields the LLL-reduced basis with vectors

$$\begin{aligned} \mathbf{b}_0 &=(-2,-1,1,0,0,0,0,0,0,0), \\ \mathbf{b}_1 &=(0,0,0,-1,0,1,-1,1,0,1), \\ \mathbf{b}_2 &=(1,-1,0,-2,1,0,0,0,0,0), \\ \mathbf{b}_3 &=(0,1,-1,-1,1,0,2,-1,0,0), \\ \mathbf{b}_4 &=(1,-1,-1,1,0,2,-1,0,0,0), \\ \mathbf{b}_5 &=(0,0,-1,0,1,0,-1,-1,1,1), \\ \mathbf{b}_6 &=(1,0,1,1,0,0,1,1,1,0), \\ \mathbf{b}_7 &=(-1,-1,0,1,-1,0,1,1,0,2), \\ \mathbf{b}_8 &=(-2,0,-1,0,-1,0,1,2,1,0), \\ \mathbf{b}_9 &=(1,1,0,-1,-2,0,1,0,2,0). \end{aligned}$$

It is trivial for Eve to scan the LLL-reduced basis and find that $\mathbf{b}_6$ solves the superincreasing subset-sum problem. Thus, Eve finds $X = (1,0,1,1,0,0,1,1,1,0)$ and easily recovers the plaintext $x = (1,0,1,1,0,0,1,1,1)$.

We have built an understanding of lattices and basis reduction, and have seen how a classical cryptosystem can be connected to lattices and thus attacked by basis reduction algorithms.

8 Appendix

Appendix 8.0.1. Example adapted from [4].

We apply the LLL Algorithm to the basis

$$\mathcal{B} = \{(1,1,1), (-1,0,2), (3,5,6)\}.$$

1. We compute $\mathcal{B}^*$. We set

   $$\begin{aligned}
   \mathbf{b}_0^* &= \mathbf{b}_0 = (1,1,1),\\
   \mathbf{b}_1^* &= \mathbf{b}_1 - \mu_{\mathbf{b}_0^*}(\mathbf{b}_1) = \Big(\frac{-4}{3}, \frac{-1}{3}, \frac{5}{3}\Big),\\
   \mathbf{b}_2^* &= \mathbf{b}_2 - \mu_{\mathbf{b}_0^*}(\mathbf{b}_2) - \mu_{\mathbf{b}_1^*}(\mathbf{b}_2) = \mathbf{b}_1 - \left\lceil \frac{\langle \mathbf{b}_0^*, \mathbf{b}_1\rangle}{\langle \mathbf{b}_0^*, \mathbf{b}_0^*\rangle} \right\rfloor \mathbf{b}_0 = \Big(\frac{-6}{14}, \frac{9}{14}, \frac{-3}{14}\Big).
   \end{aligned}$$

   We set $k = 1$ and enter the k-loop.

2. We enter the j-loop and set $j = k - 1 = 0$. We set

   $$\mathbf{b}_1 = \mathbf{b}_1 - \left\lceil \frac{\langle \mathbf{b}_0^*, \mathbf{b}_1\rangle}{\langle \mathbf{b}_0^*, \mathbf{b}_0^*\rangle} \right\rfloor \mathbf{b}_0 = (-1,0,2) - 0(1,1,1) = (-1,0,2).$$

   No reduction occurs for $\mathbf{b}_1$. We exit the j-loop.

3. We check the Lovász condition.

   $$\|\mathbf{b}_1^*\|^2 = \frac{14}{3} > \Big(\frac{3}{4} - \frac{\langle \mathbf{b}_0^*, \mathbf{b}_1\rangle^2}{\langle \mathbf{b}_0^*, \mathbf{b}_0^*\rangle^2}\Big)\|\mathbf{b}_0^*\|^2 = \frac{23}{12}.$$

   Since the Lovász condition is met, we set $k = 2$.

4. Since $k = 2 = n - 1$, we stay in the k-loop. We enter the j-loop and set $j = k - 1 = 1$. We set

   $$\mathbf{b}_2 = \mathbf{b}_2 - \left\lceil \frac{\langle \mathbf{b}_1^*, \mathbf{b}_2\rangle}{\langle \mathbf{b}_1^*, \mathbf{b}_1^*\rangle} \right\rfloor \mathbf{b}_1 = (3,5,6) - 1(-1,0,2) = (4,5,4).$$

   Now we set $j = k - 2 = 0$. We set

   $$\mathbf{b}_2 = \mathbf{b}_2 - \left\lceil \frac{\langle \mathbf{b}_0^*, \mathbf{b}_2\rangle}{\langle \mathbf{b}_0^*, \mathbf{b}_0^*\rangle} \right\rfloor \mathbf{b}_0 = (4,5,4) - 4(1,1,1) = (0,1,0).$$

   We must recompute $\mathcal{B}^*$, since we now have

   $$\mathcal{B} = \{(1,1,1), (-1,0,2), (0,1,0)\}.$$

Note that only $\mathbf{b}_2^*$ requires an update. We set

$$\mathbf{b}_2^* = \mathbf{b}_2 - \mu_{\mathbf{b}_0^*}(\mathbf{b}_2) - \mu_{\mathbf{b}_1^*}(\mathbf{b}_2) = \Big(\frac{-6}{14}, \frac{9}{14}, \frac{-3}{14}\Big)$$

and we exit the j-loop.

5. We check the Lovász condition.

$$\|\mathbf{b}_2^*\|^2 = \frac{9}{14} < \Big(\frac{3}{4} - \frac{\langle \mathbf{b}_1^*, \mathbf{b}_2\rangle^2}{\langle \mathbf{b}_1^*, \mathbf{b}_1^*\rangle^2}\Big)\|\mathbf{b}_1^*\|^2 = \frac{73}{21}.$$

Since the Lovász condition is not met, we swap $\mathbf{b}_2, \mathbf{b}_1$. We now have $\mathcal{B} = \{(1,1,1),(0,1,0),(-1,0,2)\}$, so we must recompute $\mathcal{B}^*$. Note that only $\mathbf{b}_1^*, \mathbf{b}_2^*$ require an update. We set

$$\mathbf{b}_1^* = \mathbf{b}_1 - \mu_{\mathbf{b}_0^*}(\mathbf{b}_1) = \Big(\frac{-1}{3}, \frac{2}{3}, \frac{-1}{3}\Big),$$
$$\mathbf{b}_2^* = \mathbf{b}_2 - \mu_{\mathbf{b}_0^*}(\mathbf{b}_2) - \mu_{\mathbf{b}_1^*}(\mathbf{b}_2) = \Big(\frac{-3}{2}, 0, \frac{3}{2}\Big).$$

and we set $k = \max\{k-1, 1\} = 1$.

6. Since $k = 1 < n-1$, we stay in the k-loop. We enter the j-loop and set $j = k-1 = 0$. We set

$$\mathbf{b}_1 = \mathbf{b}_1 - \Big\lceil \frac{\langle \mathbf{b}_0^*, \mathbf{b}_1\rangle}{\langle \mathbf{b}_0^*, \mathbf{b}_0^*\rangle} \Big\rfloor \mathbf{b}_0 = (0,1,0) - 0(1,1,1) = (0,1,0).$$

No reduction occurs for $\mathbf{b}_1$. We exit the j-loop.

7. We check the Lovász condition.

$$\|\mathbf{b}_1^*\|^2 = \frac{2}{3} < \Big(\frac{3}{4} - \frac{\langle \mathbf{b}_0^*, \mathbf{b}_1\rangle^2}{\langle \mathbf{b}_0^*, \mathbf{b}_0^*\rangle^2}\Big)\|\mathbf{b}_0^*\|^2 = \frac{23}{12}.$$

Since the Lovász condition is not met, we swap $\mathbf{b}_1, \mathbf{b}_0$. We now have $\mathcal{B} = \{(0,1,0),(1,1,1),(-1,0,2)\}$, so we must recompute $\mathcal{B}^*$. We set

$$\mathbf{b}_0^* = \mathbf{b}_0 = (0,1,0),$$
$$\mathbf{b}_1^* = \mathbf{b}_1 - \mu_{\mathbf{b}_0^*}(\mathbf{b}_1) = (1,0,1),$$
$$\mathbf{b}_2^* = \mathbf{b}_2 - \mu_{\mathbf{b}_0^*}(\mathbf{b}_2) - \mu_{\mathbf{b}_1^*}(\mathbf{b}_2) = \Big(\frac{-3}{2}, 0, \frac{3}{2}\Big)$$

and we set $k = \max\{k-1, 1\} = 1$.

8. Since $k = 1 < n - 1$, we stay in the k-loop. We enter the j-loop and set $j = k - 1 = 0$. We set

$$\mathbf{b}_1 = \mathbf{b}_1 - \left\lceil \frac{\langle \mathbf{b}_0^*, \mathbf{b}_1 \rangle}{\langle \mathbf{b}_0^*, \mathbf{b}_0^* \rangle} \right\rfloor \mathbf{b}_0 = (1,1,1) - 1(0,1,0) = (1,0,1).$$

We now have $\mathcal{B} = \{\mathcal{B} = \{(0,1,0),(1,0,1),(-1,0,2)\}$, so we must recompute $\mathcal{B}^*$. Note that only $\mathbf{b}_1^*, \mathbf{b}_2^*$ require an update. We set

$$\begin{aligned} \mathbf{b}_1^* &= \mathbf{b}_1 - \mu_{\mathbf{b}_0^*}(\mathbf{b}_1) = (1,0,1), \\ \mathbf{b}_2^* &= \mathbf{b}_2 - \mu_{\mathbf{b}_0^*}(\mathbf{b}_2) - \mu_{\mathbf{b}_1^*}(\mathbf{b}_2) = \Big(\frac{-3}{2}, 0, \frac{3}{2}\Big). \end{aligned}$$

Note that there have been no changes to $\mathcal{B}^*$. We exit the j-loop.

9. We check the Lovász condition.

$$\|\mathbf{b}_1^*\|^2 = 2 > \Big(\frac{3}{4} - \frac{\langle \mathbf{b}_0^*, \mathbf{b}_1 \rangle^2}{\langle \mathbf{b}_0^*, \mathbf{b}_0^* \rangle^2}\Big) \|\mathbf{b}_0^*\|^2 = \frac{3}{4}.$$

Since the Lovász condition is met, we set $k = 2$.

10. The algorithm makes one more pass through the k-loop in which no reductions or swaps occur. Hence, we omit the steps of this pass. At the end of the pass, we set $k = 3 > n - 1$. So, we exit the k-loop and return the LLL-reduced basis

$$\mathcal{B} = \{(0,1,0),(1,0,1),(-1,0,2)\}.$$

In this case, the algorithm does solve the SVP as $\|(0,1,0)\| = \lambda_0(\mathcal{L}(\mathcal{B}))$. This is feasible because we are working in very low dimensions. Let us compare the orthogonality of the input basis to the returned basis. The Hadamard ratio of the input is

$$\frac{\|(1,1,1)\|\|(-1,0,2)\|\|(3,5,6)\|}{|-3|} \approx 10.80,$$

while the Hadamard ratio of the returned basis is

$$\frac{\|(0,1,0)\|\|(1,0,1)\|\|(-1,0,2)\|}{|-3|} \approx 1.05.$$

The returned basis is clearly much nicer than the input.

9 References

[1] Ajtai, Mikls. ″The Shortest Vector Problem in L2 Is NP-hard for Randomized Reductions (Extended Abstract).″ Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing - STOC 98, 1998, 10-19. doi:10.1145/276698.276705.

[2] Akhavi, Ali. ″The Optimal LLL Algorithm Is Still Polynomial in Fixed Dimension.″ Theoretical Computer Science 297, no. 1-3 (2003): 3-23. doi:10.1016/s0304-3975(02)00616-3.

[3] Bakker, Jennifer. ″The Knapsack Problem And The LLL Algorithm.″ 2004. Accessed April 24, 2019. `https://www.math.ucsd.edu/~crypto/Projects/JenniferBakker/Math187/#Anchor-Th-1435`.

[4] Bosma, Wieb. ″4. LLL.″ Class lecture from Radboud Universiteit Nijmegen, Nijmegen, Netherlands.

[5] Dinur, I., Kindler, G., Raz, R. et al. Combinatorica (2003) 23: 205. https://doi.org/10.1007/s00493-003-0019-y

[6] Goldreich, O., D. Micciancio, S. Safra, and J.-P. Seifert. ″Approximating Shortest Lattice Vectors Is Not Harder than Approximating Closest Lattice Vectors.″ Information Processing Letters 71, no. 2 (1999): 55-61. doi:10.1016/s0020-0190(99)00083-6.

[7] Hoffstein, Jeffrey Pipher, and Joseph H. Silverman. An Introduction to Mathematical Cryptography. 2nd ed. Springer Science & Business Media, 2014.

[8] Kane, Raj. ″LLL″. April 2019. `https://github.com/rrkane/LLL`.

[9] Lagarias, J. C., and A. M. Odlyzko. ″Solving Low-Density Subset Sum Problems.″ Journal of the ACM 32, no. 1 (1985): 229-46. doi:10.1145/2455.2461.

[10] Lenstra, A. K., H. W. Lenstra, Jr., and L. Lovász. ″Factoring Polynomials with Rational Coefficients.″ Mathematische Annalen 261, no. 4 (1982): 515-34. doi:10.1007/bf01457454.

[11] Lenstra, H. W., Jr. ″Integer Programming with a Fixed Number of Variables.″ Mathematics of Operations Research 8, no. 4 (1983): 538-48. doi:10.1287/moor.8.4.538.

[12] Mavroeidis, Vasileios, Kamer Vishi, Mateusz D., and Audun Jsang. ″The Impact of Quantum Computing on Present Cryptography.″ International Journal of Advanced Computer Science and Applications 9, no. 3 (2018). doi:10.14569/ijacsa.2018.090354.

[13] May, Alexander. ″Using LLL-Reduction for Solving RSA and Factorization Problems: A Survey.″ The LLL Algorithm Information Security and Cryptography, 2009, 315-48. doi:10.1007/978-3-642-02295-1_10.

[14] Micciancio, Daniele. ″Lattices and Bases.″ Class lecture, Lattice Algorithms and Applications from University of California San Diego, San Diego, CA, USA. Spring 2007.

[15] Micciancio, Daniele, and Shafi Goldwasser. Complexity of Lattice Problems: A Cryptographic Perspective. Springer Science & Business Media, 2012.

[16] Odlyzko, A. M. ″The Rise and Fall of Knapsack Cryptosystems.″ Proceedings of Symposia in Applied Mathematics Cryptology and Computational Number Theory, 1991, 75-88. doi:10.1090/psapm/042/1095552.

[17] Regev, Oded. ″Introduction.″ Class lecture, Lattices in Computer Science from Tel Aviv University, Tel Aviv, Israel. Fall 2004.

[18] Regev, Oded. ″LLL Algorithm.″ Class lecture, Lattices in Computer Science from Tel Aviv University, Tel Aviv, Israel. Fall 2004.

[19] Shamir, Adi. ″A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem.″ Advances in Cryptology, 1983, 279-88. doi:10.1007/978-1-4757-0602-4_27.

[20] Tian, Zhaofei, and Sanzheng Qiao. ″A Complexity Analysis of a Jacobi Method for Lattice Basis Reduction.″ Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering - C3S2E 12, 2012. doi:10.1145/2347583.2347590.

[21] Vaikuntanathan, Vinod. ″Lecture 2.″ Class lecture, Lattices in Computer Science from Massachusetts Institute of Technology, Cambridge, MA, USA. September 20, 2011.