DETERMINISTIC AND NON-DETERMINISTIC BASIS REDUCTION TECHNIQUES FOR NTRU LATTICES

By

Daniel Socek

A Thesis Submitted to the Faculty of
The Charles E. Schmidt College of Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Science

Florida Atlantic University
Boca Raton, Florida
December 2002

DETERMINISTIC AND NON-DETERMINISTIC BASIS REDUCTION TECHNIQUES FOR NTRU LATTICES

by

Daniel Socek

This thesis was prepared under the direction of the candidate's thesis advisor, **Dr. Spyros S. Magliveras**, **Department of Mathematical Sciences**, and it has been approved by the members of his supervisory committee. It was submitted to the faculty of **The Charles E. Schmidt College of Science** and was accepted in partial fulfillment of the requirements for the degree of **Master of Science**.

SUPERVISORY COMITTEE:

_____

Chairperson, Thesis Advisor

_____

_____

_____

Chairman, Department of Mathematical Sciences

_____

Dean, The Charles E. Schmidt College of Science

_____ _____

Division of Research and Graduate Studies Date

Acknowledgements

I would like to thank Professor Spyros S. Magliveras, my supervisor, mentor, and true academical father, for his many suggestions, ideas and constant support during this research. For close to 6 years now, I've had the opportunity and the privilege to work with Dr. Magliveras. It was a wonderful experience, filled with mutual benefits and exciting researching projects. With his extraordinary ability to spark an interest, I developed strong tendency towards cryptography and mathematics in general.

Secondly, I would very much like to thank Professor Ron C. Mullin and Professor Frederick Hoffman for their generosity and good will to read and dedicate their expertise to my work as the members of the examining committee. I greatly appreciate their comments and suggestions which improved the quality of this work.

I thank Professor Marcus Schmidmeier who supplied me with the manuscript of some of his recent work on NTRU ring units, which gave me a better perspective on my own results.

Many thanks to Professor James Brewer for expressing interest in my work as well as for taking time to read my thesis and comment on it.

The *Charles E. Schmidt Scholarship*, which was awarded to me for the period 2001–2002, was crucial to the successful completion of this project.

Most importantly, I am grateful to my parents, and close family for their constant support.

Abstract

Author:	**Daniel Socek**
Title:	**Deterministic and Non-Deterministic Basis Reduction Techniques for NTRU Lattices**
Institution:	**Florida Atlantic University**
Thesis Advisor:	**Dr. Spyros S. Magliveras**
Degree:	**Master of Science**
Year:	**2002**

Finding the shortest or a "short enough" vector in an integral lattice of substantial dimension is a difficult problem. The problem is not known to be but most people believe it is [7]. The security of the newly proposed NTRU cryptosystem depends solely on this fact. However, by the definition NTRU lattices possess a certain symmetry. This suggests that there may be a way of taking advantage of this symmetry to enable a new cryptanalytical approach in combination with existing good lattice reduction algorithms. The aim of this work is to exploit the symmetry inherent in NTRU lattices to design a non-deterministic algorithm for improving basis reduction techniques for NTRU lattices. We show how the non-trivial cyclic automorphism of an NTRU lattice enables further reduction. Our approach combines the recently published versions of the famous LLL algorithm for lattice basis reduction with our automorphism utilization techniques.

Table of Contents

List of Tables

List of Figures

Introduction

It is evident that a new era of communications and information exchange has been underway for at least two decades. This remarkable revolution in the way we interact with one another happened almost overnight, leaving many of us unaware of the true vast and open nature of the *cyberspace.* Every day people send out love messages, sensitive personal information, financial transactions, and corporate documents, and even make business deals over the global communications channel. In such a setting, one natural question that arises is the security and confidentiality of a *digital packet* of information. This leads us to one of today's hottest branches of applied mathematics: *cryptography.*

Fortunately, cryptography has come a long way since its beginnings. We are able to choose not only extremely secure cryptographic protocols, but also ones that are fast and practical. Practical systems, however, did not exist until the mid 1970s, when *public key* cryptography was born [2]. With the rise of public key cryptography, came the possibility of the practical implementation of large user networks, where communication flow is encrypted; i.e., altered in a way that only the authorized parties can decrypt and read the original messages. Furthermore, public key protocols can be used to implement authentication protocols and, thus, the possibility of implementing digital signature schemes came into existence. But what is *public key cryptography*?

Unlike private-key (or symmetric) cryptosystems where both communicating parties have to know a secret key, specific to them, in order to use the system, in public key cryptography each party P has a pair of keys, a *public key* E_P and a *private key* D_P. The public key E_P is publicly accessible to everyone, but the private key D_P is only known to party P. When Alice (user A) wants to send a message to Bob (user B), she looks up his public key E_B, encrypts her message x as $y = E_B(x)$, and sends the encrypted message y to Bob. Bob then applies his private key to recover the message: $x = D_B(y) = D_B(E_B(x))$. That is, the bijective functions E_B and D_B are inverses of each other. An eavesdropper cannot recover the message because knowledge of the the public key E_B in no way makes it possible for one to compute the unknown private key D_B. It is possible to design scenarios where the above assumptions hold, but we will not discuss these here. This remarkable idea made cryptography implementable for large networks of users. In addition, many public key schemes provided the ability to design corresponding digital signature mechanisms.

The security of any particular public key cryptosystem relies on some computationally infeasible problem. Such problems include factoring large integers, the *discrete logarithm problem*, the *subset sum problem* (also known as the *Knapsack problem*), and the *basis reduction problem* for an integral lattice. New techniques developed in last few decades, together with the growth in computer processor speed, forced almost all public key systems to use either very large integer arithmetic, or some computationally involved group structure such as an *elliptic curve group*. As a consequence, such systems are relatively slow, and the key sizes have to be larger then the key sizes in private key cryptosystems. However, in the systems that rely

on the lattice resolution problem (finding a short target vector), the linearity of lattice operations offers remarkable speed advantages. Currently, the best such system is NTRU, recently proposed by J. Hoffstein, J. Pipher, J.H. Silverman [3]. In this thesis, the goal is to propose new techniques for NTRU lattice basis reduction, and ultimately, NTRU basis resolution, which is equivalent to breaking the system.

Chapter 1

Preliminaries

This chapter introduces some basic facts from the theory of integral lattices, and presents the details of cryptosystem NTRU.

1.1 Basics of the Theory of Integral Lattices

By an integral *lattice* $\mathcal{L}$ we mean a discrete (finitely generated) subgroup of the additive group of the vector space $\mathbb{R}^n$. Thus, $\mathcal{L}$ can be thought of as the set of all integral linear combinations of a set of vectors $\mathcal{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m\} \subset \mathbb{R}^n$. If the elements of $\mathcal{B}$ are linearly independent over $\mathbb{Z}$ then we say that $\mathcal{B}$ is a *basis* for $\mathcal{L}$ and the dimension of $\mathcal{L}$ is m. If $\mathcal{L} \subset \mathbb{R}^n$ is of dimension n then $\mathcal{L}$ is called a *full-rank* lattice in $\mathbb{R}^n$.

If $n > 1$, a lattice $\mathcal{L}$ has infinitely many bases. Cassels showed that the following theorem holds [9]: Suppose that $\mathcal{L}$ is a lattice of dimension n, with basis $\mathcal{B}$ given by the columns of a matrix B. Let T be an $n \times n$ integral matrix with determinant ± 1 and let

$$B' = TB;$$

then the columns of B' form another basis for $\mathcal{L}$. Conversely, if B and B' are matrices representing two different bases for a lattice $\mathcal{L}$, then there exists an integral matrix T of determinant ± 1 such that

$$B' = TB.$$

From basic linear algebra it follows that if $\mathcal{B}$ and $\mathcal{B}'$ are two bases of the same lattice whose vectors form columns of B and B' respectively, then

$$|\det B| = |\det B'|.$$

Hence, the quantity $|\det B|$ is an invariant for lattice $\mathcal{L}$, independent of the basis $\mathcal{B}$ and is called the *volume* of $\mathcal{L}$. We write $\text{vol}\ \mathcal{L}$ for $|\det B|$.

Definition 1.1.1. We define the *inner product* of two vectors $\boldsymbol{v} = (v_1, v_2, ..., v_n)$ and $\boldsymbol{w} = (w_1, w_2, ..., w_n)$, denoted $\langle \boldsymbol{v}, \boldsymbol{w} \rangle$, in the standard way:

$$\langle \boldsymbol{v}, \boldsymbol{w} \rangle = \sum_{i=1}^{n} v_i w_i.$$

Definition 1.1.2. A set of vectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n\}$ is said to be *orthogonal* if $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = 0$ whenever $i \neq j$. An orthogonal set where $\langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle = 1$, for $1 \leq i \leq n$ is said to be an *orthonormal* set.

In general, a *norm function* $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ is defined as one satisfying the following conditions:

(1) $\|\boldsymbol{v}\| > 0$ for $\boldsymbol{v} \neq \boldsymbol{0}$, and $\|\boldsymbol{v}\| = 0$ if $\boldsymbol{v} = \boldsymbol{0}$

(2) $\|s\boldsymbol{v}\| = |s| \cdot \|\boldsymbol{v}\|$ for any scalar $s \in \mathbb{R}$

(3) $\|\boldsymbol{v} + \boldsymbol{w}\| \leq \|\boldsymbol{v}\| + \|\boldsymbol{w}\|$ for any vectors $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^n$.

In particular we use the *Standard (Euclidean) norm* of a vector $\boldsymbol{v} = (v_1, v_2, ..., v_n)$ defined as follows:

$$\|\boldsymbol{v}\| = \langle \boldsymbol{v}, \boldsymbol{v} \rangle^{\frac{1}{2}} = \sqrt{v_1^2 + \ldots + v_n^2}$$

One important measurement characterizing lattice bases is the *geometric mean* of the norms of basis vectors; i.e., the positive n^{th} root of the product of the norms.

Definition 1.1.3. Let $\mathcal{L}$ be a lattice of dimension n. If $\mathcal{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\}$ and $\mathcal{B}' = \{\boldsymbol{b}'_1, \boldsymbol{b}'_2, \ldots, \boldsymbol{b}'_n\}$ are bases of $\mathcal{L}$ satisfying:

$$\|\boldsymbol{b}_1\| \cdot \|\boldsymbol{b}_2\| \cdot \ldots \cdot \|\boldsymbol{b}_n\| \; > \; \|\boldsymbol{b}'_1\| \cdot \|\boldsymbol{b}'_2\| \cdot ... \cdot \|\boldsymbol{b}'_n\|$$

then we say that $\mathcal{B}'$ is *reduced relative to* $\mathcal{B}$.

We use the notation $\|\mathcal{B}\| = \|\mathbf{b}_1\| \cdot \|\mathbf{b}_2\| \cdot ... \cdot \|\mathbf{b}_n\|$ and call $\|\mathcal{B}\|$ the *weight* of basis $\mathcal{B}$.

Next, we mention two results regarding lower and upper bounds for the weight of a lattice. A well known fact due to Hadamard [7] is the following inequality:

$$\|\mathcal{B}\| \geq \text{vol } \mathcal{L}.$$

On the other hand, Hermite in 1850 introduced the following upper bound:

$$\|\mathcal{B}\| \leq c_n \text{vol } \mathcal{L}$$

where c_n is a constant that depends only on n. Currently, the best known value for c_n when n is large enough is about $1.43(0.97n)^n n^{\frac{1}{4}}$ [7].

The following two related problems are considered to be critical and significant in integral lattices. Unfortunately, in general they are both intractable.

Problem 1.1.1. (The Minimum Basis Problem) Given a lattice $\mathcal{L}$, find a basis $\mathcal{B}$ such that $\|\mathcal{B}\| = \min_\alpha\{\|\mathcal{B}_\alpha\|\}$, over all possible bases $\{\mathcal{B}_\alpha\}$ of $\mathcal{L}$.

Problem 1.1.2. (The Shortest Vector Problem) Given a lattice $\mathcal{L}$, find a nonzero vector $\boldsymbol{v}_m \in \mathcal{L}$ such that $\|\boldsymbol{v}\|$ is minimal.

The following classical theorem of Minkowski (applied to the Euclidean norm) gives an upper bound for the shortest vector in a lattice:

Theorem 1.1.1. ***(Minkowski's Upper Bound)*** *In every lattice $\mathcal{L}$ of dimension n there exists the shortest nonzero vector $\boldsymbol{v}$ such that*

$$\|\boldsymbol{v}\| < c\sqrt{n}\sqrt[n]{vol\ \mathcal{L}}$$

where c is a constant.

The best known value of c for large enough N is 0.3196 .

In [7] Lovász shows that the Minimum Basis Problem is $\mathcal{NP}$-hard. On the other hand, it is not known whether the Shortest Lattice Vector Problem is $\mathcal{NP}$-hard, but it is suspected to be. Known techniques for finding a minimal or a relatively short basis will be discussed in detail in Chapter 3.

1.2 Properties of Ring $\mathbb{Z}_m[X]/(X^N - 1)$

For a given positive integer m, $\mathbb{Z}_m$ denotes the ring of integers modulo m. $\mathcal{R}$ denotes the quotient ring of polynomials with integer coefficients modulo the ideal $(X^N - 1)$; i.e., $\mathcal{R} = \mathbb{Z}[X]/(X^N - 1)$. Similarly, $\mathcal{R}_m$ denotes $\mathbb{Z}_m[X]/(X^N - 1)$. Finally, with

$d_1, d_{-1} \in \mathbb{Z}$, $\mathcal{R}\{d_1, d_{-1}\}$ denotes the set of all polynomials in $\mathcal{R}$ with d_1 coefficients equal to 1, d_{-1} coefficients equal to -1, and all other coefficients equal to 0. We also make use of the following notation: if $x, y \in \mathbb{Z}$, then x symod y denotes the residue of x modulo y in the interval $[-\lceil \frac{y}{2} - 1 \rceil, \lfloor \frac{y}{2} \rfloor]$.

Fact 1.2.1. There is a homomorphism $\phi : \mathcal{R} \longrightarrow \mathcal{R}_m$, defined by $a_0 + a_1 X + ... + a_{N-1} X^{N-1} \longmapsto (a_0 \mod m) + (a_1 \mod m)X + ... + (a_{N-1} \mod m)X^{N-1}$.

If m is a prime, $\mathbb{Z}_m$ is a field so that $\mathcal{R}_m$ forms a vector space.

To understand the multiplication operation in $\mathcal{R}$ (or $\mathcal{R}_m$) it is best to think of polynomials in $\mathcal{R}$ (or $\mathcal{R}_m$) as N-dimensional vectors in $\mathbb{Z}^N$ and $\mathbb{Z}_m^N$ (respectively) whose coordinates correspond to the polynomial coefficients. Addition in $\mathcal{R}$ (or $\mathcal{R}_m$) is identical to vector component-wise addition, while multiplication in $\mathcal{R}$ (or $\mathcal{R}_m$) becomes a cyclic convolution:

$$(c_0, \ldots, c_{N-1}) = (v_0, \ldots, v_{N-1}) \cdot (w_0, \ldots, w_{N-1})$$

where

$$c_k = \sum_{i+j \equiv k \mod N} v_i w_j.$$

Proposition 1.2.1. *Let $\boldsymbol{v} = (v_0, \ldots, v_{N-1})$, $\boldsymbol{w} = (w_0, \ldots, w_{N-1})$ and $\boldsymbol{c} = (c_0, \ldots, c_{N-1})$ be polynomials in $\mathcal{R}$ (or $\mathcal{R}_m$) such that $\boldsymbol{c} = \boldsymbol{v} \cdot \boldsymbol{w}$. Then, the product can be expressed as a simple matrix product $C = V \cdot W$ as indicated below:*

$$\begin{bmatrix} c_0 \\ \vdots \\ c_{N-2} \\ c_{N-1} \end{bmatrix} = \begin{bmatrix} v_0 & v_{N-1} & \dots & v_2 & v_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ v_{N-2} & v_{N-3} & \dots & v_0 & v_{N-1} \\ v_{N-1} & v_{N-2} & \dots & v_1 & v_0 \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_{N-2} \\ w_{N-1} \end{bmatrix},$$

where, $C = (c_0, \dots, c_{N-1})^T$, $W = (w_0, \dots, w_{N-1})^T$ *and* V *is a circulant matrix whose bottom row is* $\boldsymbol{v} = (v_0, \dots, v_{N-1})$ *in reverse order and each row of* V *is a left cyclic shift of the row below. We denote* V *by cir(*$\boldsymbol{v}$*).*

Remark 1.2.1. The more efficient way to compute this product when N is large is to use Fast Fourier Transform, which requires $O(N \log N)$ operations. [3]

In general, the finite ring $\mathcal{R}_m$ is not a field and therefore it is of interest to talk about the number of units in this ring. The following result concerning the number of units in $\mathcal{R}_q$ (where q is a power of a prime) is due to Schmidmeier [6]:

Lemma 1.2.2. *Let* $q = p^k$ *where* p *is a prime, and assume that polynomial* $X^N - 1$ *factors as a product of irreducibles over* $\mathcal{R}_p$ *as* $X^N - 1 = t_1^{n_1} \cdots t_\nu^{n_\nu}$. *Then, the number of units in* $\mathcal{R}_q$ *is:*

$$|\mathcal{R}_q^*| = |\mathcal{R}_q| \cdot \prod_{i=1}^{\nu} (1 - \frac{1}{p^{\deg t_i}}).$$

This result is particularly useful since setting up the NTRU system requires computation of random units from $\mathcal{R}_q$, and for all standard parameters q is chosen to be a power of 2.

1.3 The NTRU Cryptosystem

There are currently two proposed public key cryptosystems that rely on the difficulty of lattice basis reduction problems: GGH and NTRU. The earlier one is the GGH cryptosystem (named after its authors O. Goldreich, S. Goldwasser, and S. Halevi). Unfortunately, the lattice basis reduction techniques available today force relatively large lattice dimension n, which makes GGH impractical. Its public keys include a full lattice basis, so that the public key must contain n^2 entries which is of enormous key size when n is chosen big enough to maintain high system security. For details regarding the GGH cryptosystem, please refer to [10].

On the other hand, the NTRU cryptosystem introduced by J. Hoffstein, J. Pipher and J.H. Silverman at CRYPTO'96, still maintains small public key size since the entire lattice basis can be constructed using a single vector. However, the cyclic symmetry inherent in the NTRU setting introduces possible vulnerabilities and opens up new ways to attack the system. This will be our main topic of discussion in Chapter 4. We present details of the NTRU cryptosystem in the next few pages.

The NTRU cryptosystem is specified by six integer parameters $\{N, p, q, d_f, d_g, d_r\}$. The integers p and q, are not necessarily prime, but are chosen to be relatively prime to each other. It is desirable that q is considerably larger than p. Standard parameters proposed by NTRU authors are $p = 3$, $q = 2^k$ and $N = 1+2r$ where r is a prime. The underlying algebraic structure of NTRU is $\mathcal{R}$, which was discussed in the previous section.

1.3.1 System Setup and Key Generation

Alice, wishing to use NTRU to communicate with Bob, begins by selecting two random polynomials $\boldsymbol{f} \in \mathcal{R}\{d_f, d_f - 1\}$ and $\boldsymbol{g} \in \mathcal{R}\{d_g, d_g\}$. Note that since the coefficients of $\boldsymbol{f}$ and $\boldsymbol{g}$ are 0's or ± 1's, $\boldsymbol{f}$ and $\boldsymbol{g}$ can be viewed as elements of $\mathcal{R}_p$ and $\mathcal{R}_q$ as well. Polynomial $\boldsymbol{f}$ is required to have multiplicative inverses in both $\mathcal{R}_q$ and $\mathcal{R}_p$. According to the standard parameter choices for $N \in \{251, 347, 503\}$, polynomial $X^N - 1$ factors either as $(X-1)P_1(X)P_2(X)$ when N is 347 or 503, or $(X-1)P_1(X)P_2(X)P_3(X)P_4(X)P_5(X)$ when N is 251 over $\mathbb{Z}_2$. Here, $P_i(X)$'s are irreducible polynomials over $\mathbb{Z}_2$. Using Lemma 1.2.2 it follows that the number of units in $\mathcal{R}_q$ is slightly less then a half the size of $\mathcal{R}_q$ in all these cases.

After Alice has determined such polynomials $\boldsymbol{f}$ and $\boldsymbol{g}$, she computes $\boldsymbol{F}_q = \boldsymbol{f}^{-1}$ in $\mathcal{R}_q$, and $\boldsymbol{F}_p = \boldsymbol{f}^{-1}$ in $\mathcal{R}_p$. Finally, she computes

$$\boldsymbol{h} = p\boldsymbol{F}_q\boldsymbol{g} \pmod{q}.$$

Alice's NTRU public key is the set $\{N, p, q, d_f, d_g, d_r, \boldsymbol{h}\}$, while $\boldsymbol{f}$ is her private key.

1.3.2 NTRU Encryption

The message space $\mathcal{M}$ for the NTRU system includes all polynomials in $\mathcal{R}_p$ with coefficients reduced symod p. Suppose Bob intends to send a message $\boldsymbol{m} \in \mathcal{M}$ to Alice. He chooses some random polynomial $\boldsymbol{r} \in \mathcal{R}\{d_r, d_r\} \subset \mathcal{M}$, and computes $\boldsymbol{e} \in \mathcal{R}_q$, the encryption of $\boldsymbol{m}$, as follows:

$$\boldsymbol{e} = \boldsymbol{r}\boldsymbol{h} + \boldsymbol{m} \pmod{\text{q}}.$$

Now, since $\boldsymbol{m}$ is securely encrypted into $\boldsymbol{e}$, Bob sends $\boldsymbol{e}$ over an open channel to Alice.

1.3.3 NTRU Decryption

At the other end, Alice receives polynomial $\mathbf{e}$ and performs the following two step computation in order to decrypt $\boldsymbol{e}$:

$$\boldsymbol{a} = \boldsymbol{fe} \pmod{q},$$

$$\boldsymbol{m}' = \boldsymbol{F}_p\boldsymbol{a} \pmod{p}.$$

If the NTRU parameters are chosen well, the probability that $\boldsymbol{m}' = \boldsymbol{m}$ is very close to 1. However, some parameter choices may cause decryption failure. The implementation of NTRU should therefore include some check-bits (error correcting encoding) for each message block, since the sender alone cannot perform the decryption and check the results. There are two types of decryption failure. *Centering failure* occurs when a message is centered improperly. Let (N, p, q, d_f, d_g, d_r) be parameters for an NTRU cryptosystem and $\boldsymbol{b} = p\boldsymbol{rg} + \boldsymbol{mf}$ a typical decryption polynomial with corresponding vector coordinates $(b_1, \ldots, b_{N-1})$. If $\max_{1\leq i<N}\{b_i\} \geq q/2$ or $\min_{1\leq i<N}\{b_i\} \leq q/2$ then the decryption fails and we say that the centering failure occurred [5]. In this case, Alice can recover the original message $\boldsymbol{m}$ by adding some small integer x to all the coefficients in $\boldsymbol{a}$ in the first stage of decryption. Another, more serious failure is the so-called *gap failure* for which $\boldsymbol{m}$ is unrecoverable. If $\max_{1\leq i<N}\{b_i\} - \min_{1\leq i<N}\{b_i\} \geq q$ we say that the gap failure occurred [5]. However, for well chosen parameters, the probability of a gap failure is so small that practical implementations of NTRU can even ignore gap failures.

1.3.4 Verifying Decryption Algorithm

The following verifies that the previously defined decryption process works, assuming there are no decryption failures:

$$\boldsymbol{a} = \boldsymbol{fe} = \boldsymbol{f}(\boldsymbol{rh} + \boldsymbol{m}) = \boldsymbol{f}(\boldsymbol{r}p\boldsymbol{F}_q\boldsymbol{g} + \boldsymbol{m}) = p\boldsymbol{rg} + \boldsymbol{fm} \pmod{q},$$

$$\boldsymbol{m}' = \boldsymbol{F}_p\boldsymbol{a} = \boldsymbol{F}_p p\boldsymbol{rg} + \boldsymbol{F}_p\boldsymbol{fm} = 0 + 1\boldsymbol{m} = \boldsymbol{m} \pmod{p}.$$

Chapter 2

Lattice Basis Reduction

When Arjen K. Lenstra, Hendrik W. Lenstra Jr., and Lásló Lovász, published their paper on factoring nonzero polynomials with one indeterminate over the field of rational numbers, they revealed an algorithm, called the LLL algorithm (or the L^3 algorithm), that turned out to be a breakthrough in lattice reduction theory. This chapter covers the basic L^3 algorithm as introduced in 1982 by the aforementioned authors [1]. The L^3 algorithm, together with its improved variations (due to Schnorr and others), is the only known polynomial time algorithm for obtaining a reduced basis of a lattice. In the next chapter we will connect L^3 with the cryptanalysis of NTRU.

2.1 Basic L^3 Algorithm

Before we start describing the L^3 algorithm, let us recall the Gram-Schmidt orthogonalization process.

For a given ordered set of linearly independent vectors $\{\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_m\}$ in $\mathbb{R}^n$, the Gram-Schmidt orthogonalization procedure produces a second ordered set of vectors

$\{\boldsymbol{b}_1^*, \boldsymbol{b}_2^*, ..., \boldsymbol{b}_m^*\}$ which are mutually orthogonal and which span the same subspace as the original set of vectors. When $m = n$ and the vectors $\{\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n\}$ are presented as the columns of a matrix B, the Gram-Schmidt algorithm computes a matrix B^* whose columns are the orthogonal vectors $\{\boldsymbol{b}_1^*, \boldsymbol{b}_2^*, ..., \boldsymbol{b}_n^*\}$ and also outputs the matrix $M = \mu_{i,j}$ with diagonal entries equal to 1, which satisfy the following:

$$B = (B^*)^T M.$$

Gram-Schmidt Orthogonalization Algorithm

Input: Matrix B as defined above

Output: Matrices B^* and M as defined above

$M \leftarrow n \times n$-zero matrix

$i \leftarrow 1$

(1) $j \leftarrow 1$

$\mathbf{b}_i^* \leftarrow \mathbf{b}_i$

(2) $\mu_{i,j} \leftarrow \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$

$\mathbf{b}_i^* \leftarrow \mathbf{b}_i^* - \mu_{i,j}\mathbf{b}_j^*$

$j \leftarrow j + 1$

if $j < i$ go to (2)

$i \leftarrow i + 1$

if $i \leq n$ go to (1)

TERMINATE

Definition 2.1.1. Let $\mathcal{L}$ be a lattice, $\mathcal{B} = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)$ an ordered basis of $\mathcal{L}$, $(\boldsymbol{b}_1^*, \boldsymbol{b}_2^*, \ldots, \boldsymbol{b}_n^*)$ its Gram-Schmidt orthogonalized basis and $\mu_{i,j}$ be the entries of the $n \times n$ matrix M produced by the Gram-Schmidt algorithm. Then we say that the basis $\mathcal{B}$ is L^3-*reduced* if the following two conditions hold:

(1) $|\mu_{i,j}| \leq \frac{1}{2}$, for $1 \leq i < j \leq n$;

(2) $\|\boldsymbol{b}^*_{j+1} + \mu_{j+1,j}\boldsymbol{b}^*_j\| \geq \frac{3}{4}\|\boldsymbol{b}^*_j\|^2$, for $j = 1, 2, \ldots, n-1$. [1]

The L^3 algorithm transforms a given basis $\mathcal{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\}$ of lattice $\mathcal{L}$ into a basis $\mathcal{B}' = \{\boldsymbol{b}'_1, \boldsymbol{b}'_2, \ldots, \boldsymbol{b}'_n\}$ which is reduced relative to $\mathcal{B}$. In addition, $\mathcal{B}'$ is L^3-reduced.

Basic L^3 Algorithm

Input: basis $\{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n\}$ of the lattice $\mathcal{L}$

Output: reduced basis $\{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n\}$ of $\mathcal{L}$

$\mathbf{b}^*_i \leftarrow \mathbf{b}_i$ for $1 \leq i \leq n$

$\mu_{i,j} \leftarrow \langle \mathbf{b}_i, \mathbf{b}^*_j \rangle / \langle \mathbf{b}^*_j, \mathbf{b}^*_j \rangle$ for $1 \leq i < j \leq n$

$\mathbf{b}^*_i \leftarrow \mathbf{b}^*_i - \mu_{i,j}\mathbf{b}^*_j$ for $1 \leq i < j \leq n$

$B_i \leftarrow \langle \mathbf{b}^*_i, \mathbf{b}^*_i \rangle$ for $1 \leq i \leq n$

$k \leftarrow 2$

(1) $l \leftarrow k - 1$

perform $(\star)$

if $B_k < (\frac{3}{4} - \mu^2_{k,k-1})B_{k-1}$ then go to (2)

for $l \leftarrow k-2, k-3, ..., 1$ perform $(\star)$

if $k = n$ then TERMINATE

$k = k + 1$

go to (1)

(2) $\mu \leftarrow \mu_{k,k-1}$

$B \leftarrow B_k + \mu^2 B_{k-1}$

$\mu_{k,k-1} \leftarrow \mu B_{k-1}/B$

$B_k \leftarrow B_{k-1}B_k/B$

$B_{k-1} \leftarrow B$

exchange b_{k-1} and b_k

for $j \leftarrow 1, 2, ..., k-2$ exchange $\mu_{k-1,j}$ and $\mu_{k,j}$

for $i = k+1, k+2, ..., n$ compute $\mu_{i,k-1} = \mu_{k,k-1}\mu_{i,k-1} + \mu_{i,k}(1 - \mu\mu_{k,k-1})$

and $\mu_{i,k} = \mu_{i,k-1} - \mu\mu_{i,k}$

if $k < 2$ then $k = k - 1$

go to (1)

$(\star)$ if $|\mu_{k,l}| > \frac{1}{2}$ then:

$r \leftarrow$ integer nearest to $\mu_{k,l}$

$\mathbf{b}_k \leftarrow \mathbf{b}_k - r\mathbf{b}_l$

$\mu_{k,j} \leftarrow \mu_{k,j} - r\mu_{lj}$ for $j \leftarrow 1, 2, \ldots, l-1$

$\mu_{k,l} \leftarrow \mu_{k,l} - r$

RETURN

Remark 2.1.1. The above algorithm replaces the set of input vectors by the set of output vectors. In other words, after the execution, the set $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\}$ will correspond to the basis $\mathcal{B}'$.

It can be seen that the L^3 is a polynomial-time algorithm, with running time $O(n^4 \log B)$ where n is number of vectors in the basis and $2 \leq B \leq \max_{i=1}^{n}\{\|\boldsymbol{b}_i\|\}$, as shown in [1].

2.2 The L^3 Algorithm in Practice

The algorithm presented above is the original version of L^3 published in 1982 [1]. Since then, the L^3 algorithm was slightly improved for practical purposes in [11], [12], [13] and [14]. Schnorr's earlier work on lattice reduction was based on the use of *block Korkin-Zolotarev bases* (sometimes referred as *BKZ*). For detailed discussion of block Korkin-Zolotarev reduction (BKZ-L^3) see [12] and [13]. The BKZ-L^3 algorithm basically generalizes the classical L^3 algorithm from vector blocks of size 2 to vector

blocks of a larger size. When compared with the original L^3, the BKZ-L^3 algorithm leads to higher-quality bases; i.e., bases with shorter vectors. Victor Shoup's NTL C/C++ library [8] contains an excellent practical implementation of BKZ-L^3. For blocksize 2, the BKZ-L^3 is a better polynomial-time algorithm than the originally proposed L^3. Unfortunately, BKZ-L^3 running time is exponential for larger blocksizes.

It may be worthwhile mentioning that the latest work of H. Koy and C.P. Schnorr improves the general lattice reduction even further [14].

In the next chapter, we discuss how NTRU can be attacked via basis reduction. BKZ-L^3 with larger blocksizes is currently one of the best methods for producing a *resolution* of the NTRU lattice. However, for large enough N, BKZ-L^3 requires large blocksizes in order to compute "short" and even "close-to-short" vectors in the given lattice. In Chapter 4, we introduce a new method that uses a combination of the polynomial-time L^3 algorithm (BKZ-L^3 with blocksize 2) and our *birotation* procedure to obtain high-quality reduced bases. This technique attempts to take advantage of the existence of cyclic automorphisms of the NTRU lattice. Furthermore, this method can be parallelized.

Chapter 3

Lattice Reduction and NTRU Cryptosystem

A consequence of viewing multiplication in the ring $\mathcal{R}$ as a matrix product (as seen in Proposition 1.2.1) enables us to attack the NTRU cryptosystem by using lattice reduction techniques (due to [15]). In the next few pages, we show that if one can find a certain short vector in a given lattice, one can completely recover the NTRU secret key; i.e., the secret polynomial $\boldsymbol{f}$. Unfortunately, current lattice reduction techniques (such as the L^3 algorithm and its various improvements) are not effective for larger values of N. In such a case, after applying standard L^3 with blocksize 2 to the initial basis, the shortest resulting vectors are not short enough to break the system. If we use larger blocksize, the running time becomes exceedingly high.

3.1 Lattice Reduction Attack on NTRU

Let $\{N, p, q, d_f, d_g, d_r, \boldsymbol{h}\}$ be Bob's NTRU public key, and let $\boldsymbol{f}$ be his NTRU private key. Consider the following $2N \times 2N$ block matrix:

$$L = \left[\begin{array}{c|c} I & O \\ \hline \mathrm{cir}(\boldsymbol{h}) & qI \end{array}\right].$$

The columns of this matrix are linearly independent, and hence form a $2N$-dimensional basis for a lattice $\mathcal{L}$. We say that $\mathcal{L}$ is an *NTRU-like lattice* if the basis of $\mathcal{L}$ can be written in the above block matrix form. For $\boldsymbol{x}, \boldsymbol{y}$ in $\mathcal{R}$, denote by $(\boldsymbol{x}, \boldsymbol{y})$ the $2N$-dimensional vector obtained by concatenating $\boldsymbol{x}$ and $\boldsymbol{y}$.

Proposition 3.1.1. *If $\boldsymbol{f}, \boldsymbol{g}$ and p are parameters of an NTRU system, then $(\boldsymbol{f}, p\boldsymbol{g}) \in \mathcal{L}$.*

Proof. By the definition of $\boldsymbol{h}$ it follows that $L \cdot (\boldsymbol{f}, \boldsymbol{0})^T = (\boldsymbol{f}, p\boldsymbol{g})^T$. Therefore $(\boldsymbol{f}, p\boldsymbol{g})$ can be expressed as an integral linear combination of vectors from the basis of $\mathcal{L}$, which means that $(\boldsymbol{f}, p\boldsymbol{g}) \in \mathcal{L}$. □

Since the parameters d_f,d_g, and p are known publicly, one can easily calculate the norm of the vector $(\boldsymbol{f}, p\boldsymbol{g})$ even though $\boldsymbol{f}$ and $\boldsymbol{g}$ are not public:

$$\|(\boldsymbol{f}, p\boldsymbol{g})\| = \sqrt{2(d_f + d_g p^2) - 1}.$$

Since the standard choice for parameter p is 3, and since both d_f and d_g are (by definition) less than $\frac{1}{2}N$, we have that $\|(\boldsymbol{f}, p\boldsymbol{g})\| < \sqrt{10N - 1}$. By applying the Minkowski's upper bound (Theorem 1.1.1) to the NTRU-like lattice $\mathcal{L}$, we see that the shortest vector in $\mathcal{L}$ has a norm less than $c\sqrt{2N}\sqrt[2N]{vol\ \mathcal{L}} = c\sqrt{2N}\sqrt[2N]{q^N} = c\sqrt{2Nq} = 0.3196\sqrt{2Nq}$. If follows that $\|(\boldsymbol{f}, p\boldsymbol{g})\|$ is considerably smaller than $0.3196\sqrt{2Nq}$. This suggest that $\|(\boldsymbol{f}, p\boldsymbol{g})\|$ is probably a very short vector in $\mathcal{L}$.

In a random lattice of dimension $n = 2N$ a Gaussian estimate (see [3, 4]) of a lower bound to the length of shortest vector is

$$\lambda_{Gauss}(\mathcal{L}) = \sqrt{\frac{Nq}{\pi e}}.$$

(We will further discuss the Gaussian heuristic in the next section.)

Since $\sqrt{10N - 1}$ is normally less than $\lambda_{Gauss}(\mathcal{L})$, even though our lattice is not random, we may guess that the vector $\|(\boldsymbol{f}, p\boldsymbol{g})\|$ is indeed a very short vector in $\mathcal{L}$.

A consequence of Proposition 3.1.1 is that if we could somehow significantly reduce the given basis of $\mathcal{L}$ to obtain the unknown short vector $(\boldsymbol{f}, p\boldsymbol{g})$ then we would surely break the cryptosystem. Hence, we refer to the vector $\tau = (\boldsymbol{f}, p\boldsymbol{g})$ as the *target vector*.

There exists yet another short vector in the given NTRU-like lattice that is known in advance. For good parameter choices, such vector will have smaller norm than the target vector τ. Unfortunately, this gives no advantage in finding the unknown short vector τ.

Proposition 3.1.2. *Let vectors* $\boldsymbol{i} = (1, 1, \ldots, 1), \boldsymbol{o} = (0, 0, \ldots, 0) \in \mathbb{R}^N$. *Then the* $2N$*-dimensional vector* $(\boldsymbol{i}, \boldsymbol{o}) \in \mathcal{L}$.

Proof. Adding the first N columns of matrix L would produce vector $(\boldsymbol{i}, \boldsymbol{o})$ if the coefficients of $\boldsymbol{h}$ add up to 0; i.e., if $\boldsymbol{h}(1) = 0$. However, $\boldsymbol{g} \in \mathcal{R}(d_g, d_g)$ implies that $\boldsymbol{g}(1) = 0$, which implies that $\boldsymbol{h}(1) = p\boldsymbol{F}_q(1)\boldsymbol{g}(1) = 0$. Hence $(\boldsymbol{i}, \boldsymbol{o})$ can be expressed as an integer linear combination of columns of L; i.e., $(\boldsymbol{i}, \boldsymbol{o}) \in \mathcal{L}$ □

Remark 3.1.1. The norm of vector $(\boldsymbol{i}, \boldsymbol{o})$ is always $\sqrt{N}$.

The vector $(\boldsymbol{i}, \boldsymbol{o})$ is most likely the shortest vector in $\mathcal{L}$ (clearly less then the Minkowski's smallest vector upper bound), and this is one of the reasons why we do not refer to the target vector τ as the shortest, but as a short vector in the lattice.

Up to this date, lattice reduction constitutes the most effective attack on the NTRU cryptosystem. Therefore, a choice of system parameters must be made in such a way, to support the highest resistance against the best known lattice reduction techniques. In the next section we introduce criteria for good NTRU parameters.

3.2 Selecting Good NTRU Parameters

There are two main considerations for choosing cryptographically good NTRU parameters: *i*) maximizing the security of the NTRU cryptosystem and *ii*) minimizing the complexity of its application. One of the main security weaknesses in NTRU would occur if $\gcd(p, q) > 1$. Such a property would significantly shrink the search space, and in the case $\gcd(p, q) = p$ we would have that $\boldsymbol{e} = \boldsymbol{m} \pmod{p}$ which means that the plaintext and the ciphertext would be the same! Due to the NTRU setup and key generation, there is an important constraint that must be satisfied. Namely, the polynomial $\boldsymbol{f}$ must have multiplicative inverses in both $\mathcal{R}_p$ and $\mathcal{R}_q$. Hence, we want to select parameters in such a way that the number of units in both $\mathcal{R}_p$ and $\mathcal{R}_q$ is maximized. A useful result from [3] suggests that if $\gcd(\boldsymbol{f}(1), pq) = 1$, and if N is chosen so that the order (as a group element) of every prime dividing either p or q is large in $\mathbb{Z}_N^*$, then almost all such $\boldsymbol{f}$'s will be invertible in $\mathcal{R}_p$ and $\mathcal{R}_q$. According to this criterion, choosing the polynomial $\boldsymbol{f}$ from $\mathcal{R}(d_f, d_f - 1)$ is an extremely good idea since then $\boldsymbol{f}(1) = 1$ and therefore $\gcd(\boldsymbol{f}(1), pq) = \gcd(1, pq) = 1$. To achieve

large order of every prime divisor of p or q in $\mathbb{Z}_N^*$ a good choice would be for N to be a *Sophie Germain prime*. A prime N is called a Sophie Germain prime if $(N-1)/2$ is also a prime. For such N, a prime element in $\mathbb{Z}_N^*$ would have order $(N-1)$ or $(N-1)/2$.

Example 3.2.1. Integer 503 is a Sophie Germain prime since 503 is a prime and $(503-1)/2 = 251$ is also a prime. The authors of NTRU suggest $N = 503$ as one of the standard high-security parameters.

Additionally, maximizing the security of NTRU cryptosystem involves minimizing vulnerability to lattice reduction techniques. To achieve this, we need to explore all possible alternatives that an attacker may utilize to help the lattice reduction procedure. It is not hard to notice that an attacker may form a slightly more general NTRU-like lattice $\mathcal{L}'$ using the block matrix

$$L' = \left[\begin{array}{c|c} \alpha I & O \\ \hline \mathrm{cir}(\boldsymbol{h}) & qI \end{array}\right]$$

where α is a real number.

We call such an $\mathcal{L}'$ a *generalized NTRU-like lattice*. In this case, the target vector becomes $\tau' = (\alpha \boldsymbol{f}, p\boldsymbol{g})$. A lattice reduction method (such as the L^3 algorithm) would have the highest probability to find target vector τ' if the attacker selects α to maximize the ratio between the expected norm of a smallest vector in $\mathcal{L}'$ (we call it σ) and the norm of τ' [3, 4].

The *Gaussian heuristic* provides a method for predicting properties of random lattices. This heuristic predicts [4] that the shortest vector in a random lattice $\mathcal{L}'$ of

large dimension has approximate norm

$$\sigma = \sqrt{\frac{n}{2\pi e}}\, V^{\frac{1}{n}}.$$

where $n = \dim \mathcal{L}'$ and $V = \text{vol } \mathcal{L}'$.

The short vectors are expected to have a norm value that is very close to σ. An analogous expression for the generalized NTRU-like lattice $\mathcal{L}'$ is

$$\sigma = \sqrt{\frac{\alpha N q}{\pi e}}$$

since $\dim \mathcal{L}' = 2N$ and $\text{vol } \mathcal{L}' = \alpha^N q^N$.

Hence, in order to maximize the probability of breaking the NTRU system using lattice reduction, the attacker should choose α to maximize the following ratio:

$$(\frac{\sigma}{\|\tau'\|})^2 = (\sqrt{\frac{\alpha N q}{\pi e}} \cdot \frac{1}{\sqrt{\|\alpha \boldsymbol{f}\| + \|p\boldsymbol{g}\|}})^2 = \frac{\alpha N q}{\pi e\,(\alpha^2 \|\boldsymbol{f}\| + \|p\boldsymbol{g}\|)} = \frac{1}{Const \cdot (\alpha \|\boldsymbol{f}\| + \alpha^{-1} \|p\boldsymbol{g}\|)}\,.$$

This expression will be maximized if the attacker chooses α as follows:

$$\alpha = \frac{\|p\boldsymbol{g}\|}{\|\boldsymbol{f}\|}\,.$$

Since both $\|\boldsymbol{f}\|$ and $\|\boldsymbol{g}\|$ are known publicly, the attacker can easily calculate the value of α, and use the appropriate generalized NTRU-like lattice as input for the chosen lattice reduction method. Since α is a real number, possibly even irrational, using α directly would create messy floating point operations and calculations. We now show how the attacker can get around this problem.

Let a/b be a rational approximation of α for some integers a and b. Consider the lattice $\mathcal{L}^*$ given by the following block matrix:

$$L^* = \left[\begin{array}{c|c} aI & O \\ \hline \mathrm{cir}(b\boldsymbol{h}) & qI \end{array}\right].$$

Since $L^* \cdot (\boldsymbol{f}, \mathbf{0})^T = (a\boldsymbol{f}, bp\boldsymbol{g})^T$, by the argument in Proposition 3.1.1. the vector $(a\boldsymbol{f}, bp\boldsymbol{g})$ is in the lattice $\mathcal{L}^*$. In this case, the vector $\tau^* = (a\boldsymbol{f}, bp\boldsymbol{g})$ is our target vector.

Knowing all this, one should set up the NTRU cryptosystem parameters accordingly. It is useful to define a parameter c_h as a ratio of $\|\tau'\|$ and σ when α is $\|p\boldsymbol{g}\|/\|\boldsymbol{f}\|$:

$$c_h = \sqrt{\frac{2\pi e \|\boldsymbol{f}\| \|p\boldsymbol{g}\|}{Nq}}.$$

The parameter c_h is a measure of how far the lattice $\mathcal{L}'$ departs from a random lattice. For smaller values of c_h it is easier to apply lattice reduction and find the small target vector. When the polynomials $\boldsymbol{f}$ and $\boldsymbol{g}$ are generated, one should calculate the value of c_h to check the system security. The values of c_h that are close to 1 are considered good and the attacker will have a hard time breaking the system via lattice reduction techniques (such as variants of the L^3 algorithm).

Remark 3.2.1. To simplify arguments and experiments we will set $\alpha = 1$ and choose system parameters accordingly.

We were able to utilize a lattice reduction attack on NTRU by combining Victor Shoup's NTL (Number Theory Library) package [8] and programming language APL.

Using our software, we were able to break NTRU cryptosystems for most values of $N < 45$ and randomly generated polynomials $\boldsymbol{f}$ and $\boldsymbol{g}$ by using BKZ-L^3 algorithm with blocksize 2. For slightly larger values of N we were successful only after applying the same algorithm with blocksize > 2. The running-time was notably longer for bigger blocksizes. Unfortunately, the needed blocksize to break the system grows exponentially with respect to the growth of N.

Chapter 4

Parallel Symmetrized Attack on NTRU Cryptosystem

In this chapter we introduce a new attack based on the fact that NTRU lattices have non-trivial automorphism groups. This attack relies on the symmetry of the NTRU-like lattice and can be parallelized.

4.1 Notation

Definition 4.1.1. Let $\boldsymbol{v} = (v_1, v_2, ..., v_{2N})$. Define *birotation* of $\boldsymbol{v}$ by k positions, denoted $birotate_k(\boldsymbol{v})$, as the vector

$$\big(v_{1+k \bmod N}, v_{2+k \bmod N}, \ldots, v_{N+k \bmod N}, v_{N+(1+k \bmod N)}, \ldots, v_{N+(N+k \bmod N)}\big).$$

Example 4.1.1. Let $\boldsymbol{v} = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)$. Then, birotation of $\boldsymbol{v}$ by 2 positions is $birotate_2(\boldsymbol{v}) = (3, 4, 5, 6, 1, 2, 9, 10, 11, 12, 7, 8)$.

Definition 4.1.2. A *permutation matrix* is a square zero-one matrix having exactly one entry equal to 1 in each row and each column.

We now define permutation matrices algebraically.

Definition 4.1.3. Let S_n be the symmetric group of degree n, and let $\pi \in S_n$. The *permutation matrix P_π associated with* π is an $n \times n$ zero-one matrix which has 1's in positions $(i,\pi(i))$, $1 \leq i \leq n$, and 0's elsewhere.

Example 4.1.2. If $\pi = (1\ 5\ 3)(2\ 4)$, then a matrix associated with π is

$$P_\pi = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

As the name suggests, applying an $n \times n$ permutation matrix P_π to an $n \times n$ matrix M will permute either the rows or the columns of M, depending on which side we apply P_π. For example, multiplying M by P on the left will permute the rows of M according to π^{-1}, while multiplying M by P on the right will permute the columns of M according to π.

Let P_k denote the permutation matrix that performs a cyclic shift by k positions on the symbols $1, 2, \ldots, n$. Then $n \times n$ matrix P_k, where $k \in \mathbb{Z}_n$, can be expressed with following block-matrix form:

$$P_k = \left[\begin{array}{c|c} O_* & I_{n-k} \\ \hline I_k & O_*^T \end{array}\right],$$

where I_i denotes the $i \times i$ identity matrix, and O_* denotes the $(n-k) \times k$ zero matrix. We call P_k a *k-shift permutation matrix.*

Note that we can denote a k-shift permutation matrix by using the permutation $\pi = (1\ \ 2\ \ \ldots\ \ n)$ as follows:

$$P_k = P_{\pi^k} = (P_\pi)^k.$$

The following fact can be easily verified:

Fact 4.1.1. If $\boldsymbol{v} = (v_1, v_2, \ldots, v_{2N}) \in \mathcal{R}$, then

$$birotate_k(\boldsymbol{v}) = \left[\begin{array}{c|c} P_k & O \\ \hline O & P_k \end{array}\right] \boldsymbol{v},$$

where P_k is the $N \times N$ k-shift permutation matrix, and O is the $N \times N$ zero matrix.

4.2 BIROT Algorithm

Before we discuss the details of our BIROT algorithm, we observe the following simple lemma:

Lemma 4.2.1. *Let $\boldsymbol{h} = (h_1, h_2, \ldots, h_n) \in \mathcal{R}$, $H = cir(\boldsymbol{h})$, and P_k be an $n \times n$ k-shift permutation matrix. Then, $P_k H = H P_k$. (This follows directly from the definition of H and P_k)*

Next, we introduce the following proposition:

Proposition 4.2.2. *Let $\mathcal{L}$ be an NTRU-like lattice. Then, $\boldsymbol{v} \in \mathcal{L}$ if and only if $birotate_k(\boldsymbol{v}) \in \mathcal{L}$.*

Proof. Let $\mathcal{L}$ be an NTRU-like lattice, L a $2N \times 2N$ matrix whose columns form a basis for $\mathcal{L}$, and P_k the $N \times N$ k-shift permutation matrix. Let $\boldsymbol{v} \in \mathcal{L}$, and let $\overline{P}$ be the following block matrix:

$$\overline{P} = \left[\begin{array}{c|c} P_k & O \\ \hline O & P_k \end{array}\right], \quad \text{where } O \text{ is the } N \times N \text{ zero matrix.}$$

Then, $\boldsymbol{v}$ can be expressed as an integer linear combination of columns of L; i.e., there exists a vector $\boldsymbol{x} = (x_1, \ldots, x_{2N})$, with all integer coefficients, such that

$$\boldsymbol{v} = L\boldsymbol{x}.$$

Multiplying both sides by $\overline{P}$ yields:

$$\overline{P}\boldsymbol{v} = \overline{P}L\boldsymbol{x} = \overline{P}L\overline{P}^{-1}\overline{P}\boldsymbol{x}.$$

By using Fact 4.1.1 we get that:

$$birotate_k(\boldsymbol{v}) = \overline{P}L\overline{P}^{-1} \cdot birotate_k(\boldsymbol{x}).$$

On the other hand, by Lemma 4.2.1 the following equality holds:

$$\overline{P}L\overline{P}^{-1} = \left[\begin{array}{c|c} P_k & O \\ \hline O & P_k \end{array}\right] \left[\begin{array}{c|c} I & O \\ \hline \mathrm{cir}(\boldsymbol{h}) & qI \end{array}\right] \left[\begin{array}{c|c} P_k^{-1} & O \\ \hline O & P_k^{-1} \end{array}\right] =$$

$$= \left[\begin{array}{c|c} P_k & O \\ \hline P_k\mathrm{cir}(\boldsymbol{h}) & qP_k \end{array}\right] \left[\begin{array}{c|c} P_k^{-1} & O \\ \hline O & P_k^{-1} \end{array}\right] =$$

$$= \left[\begin{array}{c|c} P_k P_k^{-1} & O \\ \hline P_k \mathrm{cir}(\boldsymbol{h}) P_k^{-1} & qP_k P_k^{-1} \end{array}\right] = \left[\begin{array}{c|c} I & O \\ \hline \mathrm{cir}(\boldsymbol{h}) & qI \end{array}\right] = L.$$

Finally we have that $birotate_k(\boldsymbol{v}) = L \cdot birotate_k(\boldsymbol{x})$. Since $birotate_k(\boldsymbol{x})$ is some vector of integer entries, $birotate_k(\boldsymbol{v})$ can be written as an integer linear combination of columns of L; i.e., $birotate_k(\boldsymbol{x}) \in \mathcal{L}$. For the converse, assume $birotate_k(\mathbf{v}) \in \mathcal{L}$ and observe that $birotate_{N-k}(birotate_k(\boldsymbol{v})) = birotate_N(\boldsymbol{v}) = \boldsymbol{v}$ so by the direct statement of this proposition it follows that $\boldsymbol{v} \in \mathcal{L}$. □

The following lemma follows easily by elementary considerations:

Lemma 4.2.3. *Let $\mathcal{L}$ be any integral lattice, and suppose that vectors $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ form a basis for $\mathcal{L}$. If $\boldsymbol{w} \in \mathcal{L}$ then, for any i, $1 \leq i \leq n$, the set of vectors $V' = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{i-1}, \boldsymbol{w}, \boldsymbol{v}_{i+1}, \ldots, \boldsymbol{v}_n\}$ forms a spanning set for a sublattice $\mathcal{L}'$ of $\mathcal{L}$. If vectors in V' are linearly dependent over $\mathbb{Z}$, then $\det V' = 0$ and dim $\mathcal{L}' <$ dim $\mathcal{L}$. When the vectors in V' are linearly independent over $\mathbb{Z}$, then the sublattice $\mathcal{L}'$ is of the same dimension as $\mathcal{L}$ and V' forms a basis of $\mathcal{L}'$.*

Proposition 4.2.4. *If $\mathcal{L}'$ is a sublattice of an integral lattice $\mathcal{L}$ and L', L are their bases respectively, both of dimension $n \times n$, then $\det(L) \mid \det(L')$. Furthermore, $\mathcal{L} = \mathcal{L}'$ if and only if $|\det L'| = |\det L|$.*

Proof. Since $\mathcal{L}'$ is a sublattice of $\mathcal{L}$, the columns of L' can be written as integer linear combinations of columns of L. Thus, there exists an $n \times n$ matrix X such that $L' = XL$. This implies that $\det L' = \det X \cdot \det L$. Since entries of X are all integers it follows that $\det X \in \mathbb{Z}$. Thus, we have that $\det L | \det L'$. □

Using these facts (see also page 5 in the Preliminaries), one can observe the following consequences: Let $\{N, p, q, d_f, d_g, d_r, \boldsymbol{h}\}$ be an NTRU cryptosystem, and let L be a basis of the corresponding lattice $\mathcal{L}$. Without loss of generality, assume that the columns of L (denoted by $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{2N}$) are sorted by their norm values in acceding order. Let L' be $2N \times 2N$ matrix whose columns are $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{2N-1}, birotate_k(\mathbf{c}_1)$. By Proposition 4.2.2, $birotate_k(\mathbf{c}_1)$ is in the lattice $\mathcal{L}$, and thus, if the columns of L' are linearly independent, then by Lemma 4.2.3 L' is a basis of a sublattice of $\mathcal{L}$. Furthermore, by Proposition 4.2.4, $\det(L')/\det(L)$ is an integer. From the previous discussion in the Preliminaries we conclude that if this value is ± 1 then L' is another basis for $\mathcal{L}$, having weight less than or equal to the weight of the original basis L.

BIROT Algorithm

Input: $2N \times 2N$ matrix L (a basis of an NTRU-like lattice), and integers m, n $(1 \leq m < n \leq 2N)$

Output: $2N \times 2N$ matrix L' with weight less then the weight of L

$L'' \leftarrow L$

$i \leftarrow 1$

$a \leftarrow \det(L)$

(1) n^{th} column of $L'' \leftarrow birotate_i(m^{th}$ column of $L'')$

$b \leftarrow \det(L'')$

if $\frac{b}{a} = \pm 1$ go to (2)

$i \leftarrow i + 1$

if $i < N$ go to (1)

$L' \leftarrow L$

TERMINATE

(2) $L' \leftarrow L''$

TERMINATE

We say that algorithm BIROT had a hit if the weight of the output basis is lower

than that of the input basis. Numerous experimental results suggest that BIROT will always result in a hit for a basis output by the basic L^3 algorithm. Furthermore, applying L^3 to a BIROT reduced basis results in further reduction. This leads us to an algorithm that iteratively combines a number of L^3 and BIROT computational blocks to reduce the basis of an NTRU-like lattice. We call this algorithm "GAME". In addition, as we will soon observe, portions of GAME can be parallelized.

4.3 Let's play GAME

Algorithm GAME takes an NTRU-like lattice as input and outputs a reduced lattice basis of smaller weight, possibly containing the short vector we seek to break the associated NTRU cryptosystem.

Remark 4.3.1. Algorithm L^3 alone produces the same output when applied once or multiple consecutive times assuming the L^3 parameters are constant. The idea is to somehow randomize the L^3 output lattice so that applying the same L^3 process again would have further reduction effects.

To keep the notation simple, we call BKZ-L^3 with blocksize 2 simply L^3. We use this version of L^3 since it includes the latest improvements and, when used with blocksize 2, it preserves polynomial time execution. In the initial step, GAME applies the L^3 algorithm to the NTRU-like lattice basis. The output lattice basis from this initial step we call *the first stage basis*. In the next step, we apply the BIROT algorithm to reduce the basis and to randomize the L^3 output basis. Now we apply L^3. It is well known that the degree of a success in reducing a basis of a given lattice

by means of L^3 is sensitive to the ordering of the vectors of the input basis. Thus, it is reasonable to submit a number of differently ordered bases to the L^3 algorithm and select the output basis of the smallest weight. In the next step of GAME, we produce p bases $L_1, L_2, \ldots, L_p$ each of which is obtained by randomly permuting the columns of the BIROT output basis, and then applying L^3 to each of them. After selecting the output basis with the smallest weight, we have concluded a round. Now, we simply iterate the algorithm from the BIROT procedure. There is no rule as to how many times one can perform the GAME loop. At some point the basis will not be reducible further and we either have lattice resolution or a significant basis reduction (in comparison to the first stage basis). The BIROT step combined with the random column permutation in the GAME algorithm does exactly what we need: it randomizes the basis to allow consecutive L^3 reductions.

The pseudo code for the algorithm GAME is given below:

GAME Algorithm

Input: $2N \times 2N$ matrix L (a basis of an NTRU-like lattice), and integers p and r

Output: $2N \times 2N$ matrix L' with weight less then the weight of L

$L'' \leftarrow L$

$i \leftarrow 1$

(1) perform $(\star)$

$m \leftarrow 1$

if norm of $1st$ column of $L_{min} = \sqrt{N}$ then $m \leftarrow 2$

$n \leftarrow 2N$

(2) $L'' \leftarrow \mathsf{BIROT}(L_{min}, m, n)$

if $L'' = L_{min}$ go to (3)

$i \leftarrow i + 1$

if $i \leq r$ go to (1)

$L' \leftarrow L''$

TERMINATE

(3) $n \leftarrow n - 1$

go to (2)

($\star$) $L_{min} \leftarrow L''$

$j \leftarrow 1$

($1^\star$) $L''_\star \leftarrow$ randomly permuted columns of L''

$L'_\star \leftarrow \mathsf{LLL}(L''_\star)$

if weight of $L'_\star <$ weight of L_{min} then $L_{min} \leftarrow L'_\star$

$j \leftarrow j + 1$

if $j \leq p$ go to ($1^\star$)

sort columns of L_{min} in acceding order by their norms

TERMINATE

4.4 Parallelization

The step where we perform the L^3 algorithm on n lattices clearly can be parallelized into n independent (parallel) processes. In addition, part of the BIROT algorithm itself can be parallelized into $N-1$ parallel processes, one for each $birotate_i$ call (where i runs from 1 to $N-1$). This feature makes the GAME algorithm practical by significantly reducing the running time. We call the parallel counterparts of algorithms BIROT and GAME simply P-BIROT and P-GAME. To minimize the running time, our implementation of P-GAME should use P-BIROT as a subroutine.

Figure 4.1 shows the parallel implementation of BIROT algorithm (P-BIROT) using the schematic diagram:

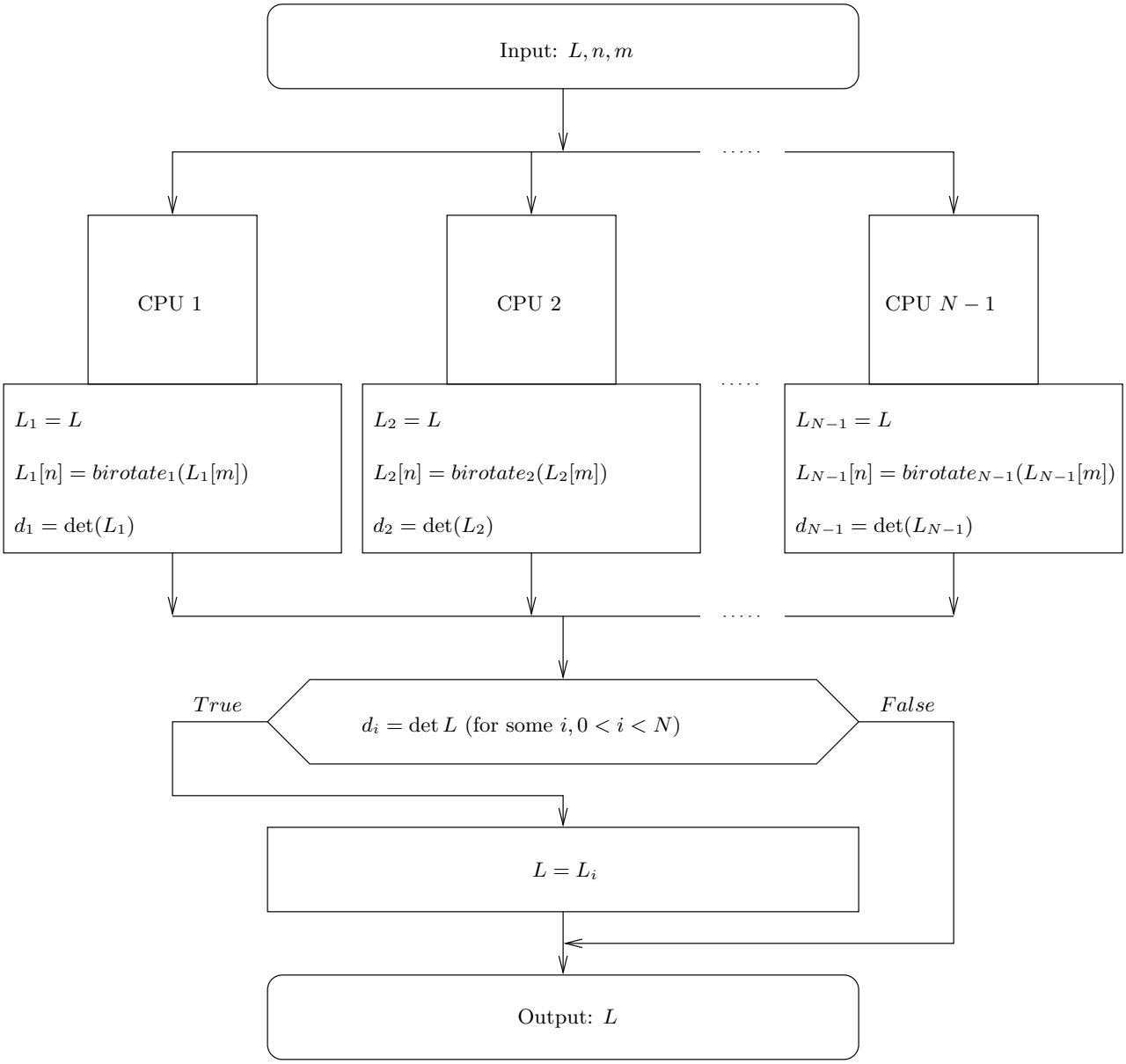

Figure 4.1: Schematic diagram representing the parallelized BIROT algorithm

Next, we introduce the parallelized GAME algorithm (P-GAME) in Figure 4.2 using the same schematic diagram representation:

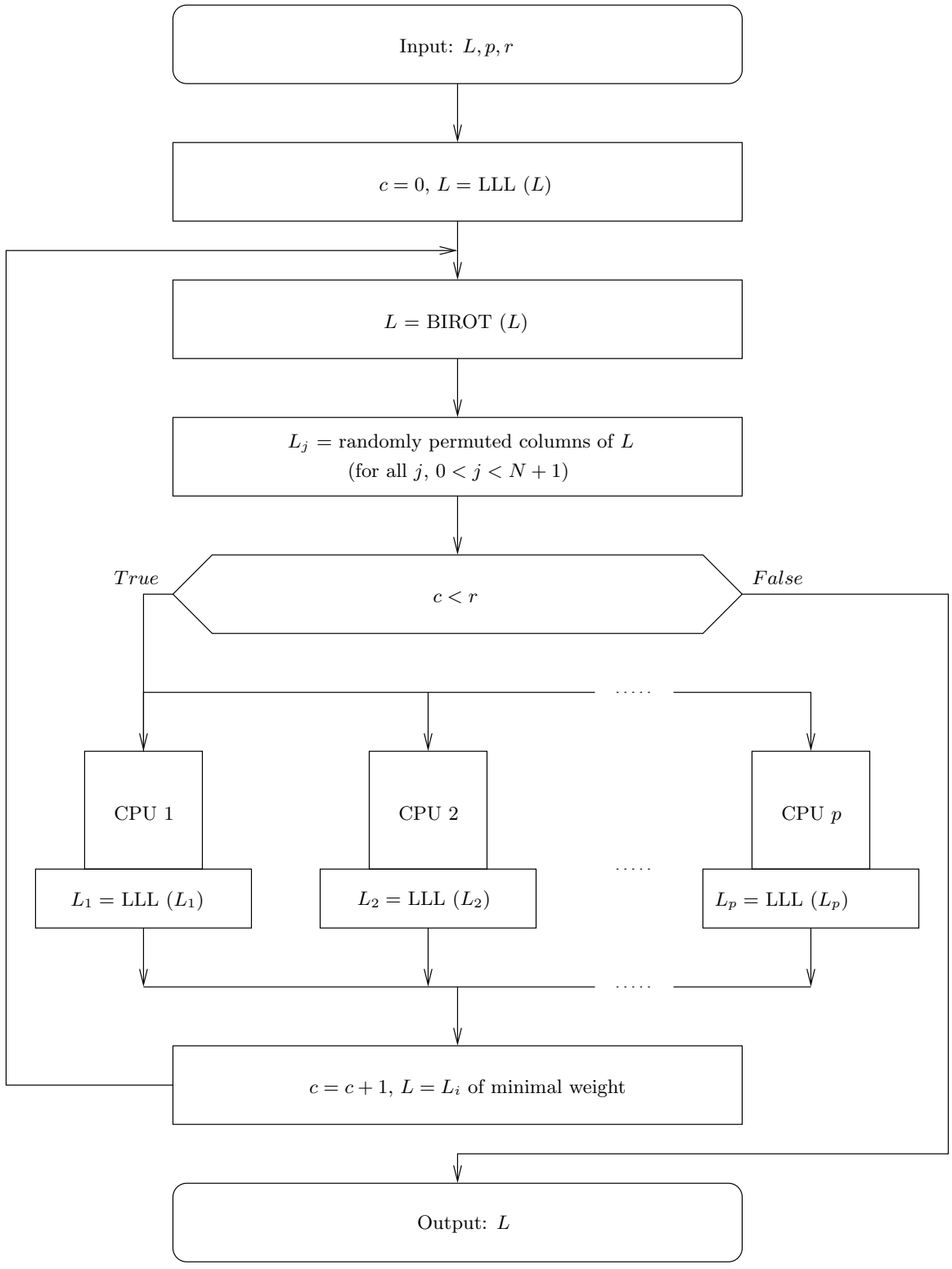

Figure 4.2: Schematic diagram representing the parallelized GAME algorithm

Chapter 5

Tests of Performances

We performed experiments to test the performance of our GAME algorithm and compare it to BKZ-L^3 algorithm with large blocksize. We have compiled C routines that include Victor Shoup's NTL (Number Theory Library) [8] package into DLL's (Dynamic Link Libraries) and used them together within an APL 4.0 programming environment. Due to long running times of both algorithms, large blocksize BKZ-L^3 and GAME, we limited our tables to relatively small values of N, yet not too small since in that case BKZ-L^3 with blocksize 2 would result in lattice resolution after the initial run.

The following example was run on an Intel Pentium® processor with a clock-speed of 450MHz and it illustrates an attack on the NTRU cryptosystem case using our algorithm GAME:

Example 5.0.1. Let us define an NTRU cryptosystem by using the following parameters:

$N = 41$,

$p = 3$,

$q = 2^7 = 128,$

$df = 6,$

$d_g = 6,$

$d_r = 4.$

The following are the private key components $\boldsymbol{f}$ and $\boldsymbol{g}$:

$\boldsymbol{f} = (-1, 0, 0, 0, -1, 0, 1, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, -1, 1, 0, 0, 0, 1, 0, 0,$
$-1, 0, 0, 1, 0, 0, 0, 0, 0, 0),$

$\boldsymbol{g} = (-1, -1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, -1, -1, 0, 0, -1, 0, 0, 0, 0,$
$0, 0, 0, 0, -1, 0, 0, 0, 0, 0).$

The public key polynomial $\boldsymbol{h}$ is generated as follows:

$\boldsymbol{h} = (34, 87, 118, 85, 24, 89, 37, 85, 72, 50, 118, 105, 123, 109, 52, 42, 95, 38, 103, 127, 37,$
$99, 99, 126, 79, 14, 78, 39, 82, 90, 105, 69, 74, 74, 94, 18, 79, 121, 110, 81, 39).$

Therefore, the target vector τ (see Chapter 3) is:

$\tau = (\boldsymbol{f}, p\boldsymbol{g}) = (-1, 0, 0, 0, -1, 0, 1, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, -1, 1, 0,$
$0, 0, 1, 0, 0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 0, -3, -3, 3, 0, 3, 0, 0, 0, 3, 0, 0, 0, 0, 3, 0, 0, 3, 0, 0, 0, 3,$
$0, 0, -3, -3, 0, 0, -3, 0, 0, 0, 0, 0, 0, 0, 0, -3, 0, 0, 0, 0).$

Suppose we only know the public information for this particular NTRU case; i.e., we pretend that we do not know polynomials $\boldsymbol{f}$ and $\boldsymbol{g}$. From the previous discussion we are able to calculate the norm value of the target vector $(\boldsymbol{f}, p\boldsymbol{g})$. In particular, the norm value of the target vector of the above defined system is 10.908712.

We form an NTRU-like lattice for the NTRU system defined above. The corresponding matrix L is an 82 x 82 matrix of integer coefficients. The following table is a

summary of the algorithm GAME applied to matrix L. Parameter r (number of processors used in parallel) was chosen to be 20

Iteration	wt (L)	wt(L) after r parallel randomized L^3	wt(L) after BIROT	Time	Cumulative Time
0	262.76223	114.26332	113.67514	13	13
1	113.67514	90.573651	90.066526	7	20
2	90.066526	87.45698	87.043456	3	23
3	87.043456	85.096357	84.674356	3	26
4	84.674356	71.530197	71.212539	6	32
5	71.212539	71.018393	70.638393	4	36
6	70.638393	62.844481	N/A	6	42

Table 5.1: Performance of GAME algorithm when parameter N=41

and resolution occurred after only 6 iterations. We performed the sequential version of GAME; however, we constructed the table and measurements as if the algorithm GAME was parallelized. The following is a list of rounded ratios of the norms of vectors in L with the norm of a target vector τ:

0 1 1 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7.

Algorithm GAME found a basis of weight 62.844481 which contains the two birotations of a target vector. Thus, after a running time of 42 seconds (the time measurements in the table above are in seconds), we broke the given NTRU cryptosystem for $N = 41$.

On the other hand, the following table is constructed to illustrate the performance of the BKZ-L^3 algorithm with larger blocksizes:

Blocksize	wt (L) after L^3	Time
2	114.26332	13
3	18.101105	48

Table 5.2: Performance of BKZ-L^3 algorithm when parameter N=41

In this case, we got resolution for blocksize 3 but the running time was 48 seconds. Hence, in this example, algorithm GAME performed slightly better (as far as breaking the system).

A second example was run on an Intel Pentium® processor with a clock-speed of 966MHz and the results are even more impressive:

Example 5.0.2. Let us choose the NTRU cryptosystem with the following set of parameters:

$N = 43$,

$p = 3$,

$q = 128$,

$d_f = 12$,

$d_g = 12$,

$d_r = 9$,

$\boldsymbol{f} = (-1, 1, -1, -1, 0, 0, 1, 1, -1, 0, -1, 1, -1, 0, 0, 0, 1, 0, 0, 1, 0, -1, 0, 0, -1, 0, 0, -1, 0,$

$1, -1, -1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0)$,

$\boldsymbol{g} = (0, -1, -1, -1, -1, 1, 1, -1, -1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, -1, 0, 1, 0, 1, 0, 1, -1, -1, 0, 0, 0, 1, 1, -1, 0, -1, 1, -1, 1, 0, 0)$,

$\boldsymbol{h} = (112, 110, 53, 26, 98, 111, 80, 17, 87, 41, 115, 70, 50, 68, 84, 82, 21, 43, 63, 60, 94, 113, 101, 5, 78, 15, 59, 101, 46, 80, 115, 51, 101, 29, 118, 19, 124, 117, 30, 3, 17, 37, 100)$,

$\tau = (\boldsymbol{f}, p\boldsymbol{g}) = (-1, 1, -1, -1, 0, 0, 1, 1, -1, 0, -1, 1, -1, 0, 0, 0, 1, 0, 0, 1, 0, -1, 0, 0, -1, 0, 0, -1, 0, 1, -1, -1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, -3, -3, -3, -3, 3, 3, -3, -3, 0, 0, 0, 3, 0, 3, 0, 0, 0, 0, 3, 0, 0, -3, 0, 3, 0, 3, 0, 3, -3, -3, 0, 0, 0, 3, 3, -3, 0, -3, 3, -3, 3, 0)$.

We represent the summary of the attack using algorithm GAME in the following table:

Iteration	wt (L)	wt(L) after r parallel randomized L^3	wt(L) after BIROT	Time	Cumulative Time
0	254.98346	75.444337	75.011991	16	16
1	75.011991	60.483632	60.215711	5	21
2	60.215711	55.208116	N/A	4	25

Table 5.3: Performance of GAME algorithm when parameter N=43

After only 2 iterations and the running time of 25 seconds, algorithm GAME found a birotation of the target vector in the lattice of weight 55.208116. The following is a list of rounded ratios of the norms of vectors in L with the norm of a target vector τ:

0 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4.

Next, we present the performance of the BKZ-L^3 algorithm with various blocksizes. This approach in attacking the NTRU system defined above performed poorly in comparison to the GAME attack. As one can see from the following table, BKZ-L^3 needed blocksize 4 to find a birotation of the target vector τ:

Blocksize	wt (L) after L^3	Time
2	75.444337	16
3	56.458075	75
4	20.786134	108

Table 5.4: Performance of BKZ-L^3 algorithm when parameter N=43

The time to break the system was 108 seconds (1 minute and 48 seconds). Clearly, in this example algorithm GAME performed more than 4 times faster than the BKZ-L^3 algorithm.

However, we have to stress that in some instances our algorithm GAME performed worse than BKZ-L^3 with large blocksize. This is due to the random nature of the GAME process.

Chapter 6

Conclusions and Further Research

We found new ways of exploiting the symmetry of NTRU-like lattices. The method we proposed combines deterministic and non-deterministic approaches into one algorithm. Due to the random nature of this algorithm its performance varies, but in many instances the performance of our method is better than the performance of other NTRU attacks. In general, this method can be expanded for solving any integral lattices that posses non-trivial automorphisms (assuming we change the permutation from birotation into the appropriate one). Such integral lattices appear in problems from other areas of mathematics and science. For example, there are problems which involve reducing integral lattices with non-trivial automorphisms that appear in t-designs.

The possibilities for further research are quite extensive. Whenever a process contains some randomness there is a good chance for improvement by discovering some regularities that can replace complete randomness with a somewhat constrained randomness. This is what we are currently working on: we are trying to understand which permutations of the columns of a lattice basis matrix will result in a better performance. We are also looking at how permutations of different distances applied

to the ordering of columns change the performance. This may result in an excellent *hill-climbing* method for reducing the basis of the lattice with non-trivial automorphisms.

Bibliography

[1] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring Polynomials with Rational Coefficients*, Mathematische Annalen **261** (1982), pp 515-534.

[2] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton (1997).

[3] J. Hoffstein, J. Pipher, J.H. Silverman, *NTRU: A Ring-Based Public Key Cryptosystem*, Preprint (1998).

[4] J. Hoffstein, J. Pipher, J.H. Silverman, *The NTRU Signature Scheme: Theory and Practice*, Technical Report (2001).

[5] J.H. Silverman, *The NTRU Cryptosystems Techical Report: Wraps, Gaps, and Lattice Constants*, Technical Report **011** (2001).

[6] M. Schmidmeier, *Ring Units in Fast Public Key Cryptology*, Preprint, Florida Atlantic University (2002).

[7] M. Grötschel, L. Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag **2** (1991), pp 139-156.

[8] V. Shoup, *Number Theory Library (NTL)*, `http://www.shoup.net`

[9] J.W.S. Cassels, *An Introduction to the Geometry of Numbers*, Springer-Verlag (1959).

[10] O. Goldreich, S. Goldwasser, S. Halevi, *Public-key Cryptography from the Lattice Reduction Problems*, CRYPTO'97, Lecture Notes in Computer Science **1294**, Springer-Verlag (1997), pp 112-131.

[11] C.P. Schnorr *A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms*, Theoretical Computer Science **53** (1987), pp 201-224.

[12] C.P. Schnorr *Block Korkin-Zolotarev Bases and Successive Minima*, Technical Report TR-92-063 (1992).

[13] C.P. Schnorr, M. Euchner *Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems*, Math. Programming **66** 2-A (1994), pp 181-199.

[14] H. Koy, C.P. Schnorr, *Segment and Strong Segment LLL-Reduction of Lattice Bases*, Preprint University Frankfurt, (2002).

[15] D. Coppersmith and A. Shamir, *Lattice attacks on NTRU*, Advances in Cryptology - EUROCRYPT '97, Walter Fumy (Ed.), Springer LNCS volume **1233** (1997), pp 52-61.