AN OVERVIEW OF THE NTRU CRYPTOGRAPHIC SYSTEM

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

in

Mathematics

---

by

Hien Ba Nguyen

Fall 2014

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Hien Ba Nguyen:

AN OVERVIEW OF THE NTRU CRYPTOGRAPHIC SYSTEM

J. C. Interlando

Carmelo Interlando, Chair
Department of Mathematics and Statistics

Robert Grone

Robert Grone
Department of Mathematics and Statistics

Carl Eckberg

Carl Eckberg
Department of Computer Science

DECEMBER 12, 2014

Approval Date

DEDICATION

Dedicated to my parents, who have raised and supported me in achieving my goals . Also, I would like to dedicate to all my grandparents, my uncles and aunts who helped and encouraged me to finish my education. Finally, I owe a special thank to my best friend Nam, who has given me mental support and motivation to complete this thesis.

ABSTRACT OF THE THESIS

AN OVERVIEW OF THE NTRU CRYPTOGRAPHIC SYSTEM
by
Hien Ba Nguyen
Master of Arts in Mathematics
San Diego State University, 2014

NTRU is one of the few public key cryptosystems which is supposedly quantum-computer resistant. Its security is based on the presumed difficulty of a lattice problem, namely, the shortest vector problem. This research paper describes the NTRU cryptosystem and its cryptanalysis. Finally, a comparison of the performance of the NTRU with other public key crytosystems such as RSA, Elliptic Curve, and McEliece, is presented.

TABLE OF CONTENTS

LIST OF TABLES

PAGE

LIST OF FIGURES

PAGE

ACKNOWLEDGMENTS

I would like to thank Dr. Robert Grone, who taught me three classes during my undergraduate years. In addition, Dr. Grone gave me very valuable advice when I was struggling during my first semester of my master's program. I also would like to thank Dr. Carl Eckberg, who taught me mathematical logic. And finally, I would like to thank Dr. Camelo Interlando, who taught me three classes in my master's program and was also my thesis advisor. Without Dr. Interlando's assistance in ideas and clear guidance, this thesis would not exist.

CHAPTER 1

INTRODUCTION

The earliest example of cryptography was the simple text of a message to deny information to the untrained eye and to increase its importance when it was revealed [5]. The history of the very word cryptography originates from the Greek words kryptos and graphein, which mean hidden and writing, respectively [19]. The first recorded use of cryptography dated back as early as 400 BC, when the Spartans employed it for secret communications between military leaders [16]. This early cipher device called the scytale comprised of a tapered baton that was spirally wrapped around with a strip containing message of leather or parchment. When unwrapped, the letters of this secret message were scrambled to form a cipher; however, when the strip was wrapped around an equal size of the original baton, the plaintext resurfaced. Cryptology luckily has evolved past the simple strips of leather and parchment paper on batons into complex cryptosystems. Modern cryptography is founded on mathematics and computer science. Cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. Well-known cryptosystems include RSA (Rivest, Shamir, and Adleman) and elliptic curve cryptosystems (ECC). In general, these cryptosystems have been effective; however, with the introduction of an adversary called quantum computers, these cryptosystems have been found to be insecure.

Quantum computers are well known to be able to break cryptosystems that are based on the integer factorization problem or some discrete logarithm problem in polynomial time. This affects RSA and ECC and also number field cryptosystems [12]. The NTRU cryptosystem is an alternative algorithm based on a completely different mathematical problem from RSA and ECC called the closest lattice vector problem. As a result, NTRU is very fast and resistant to quantum computers. The McEliece cryptosystem is one of the first public-key cryptosystems based on linear error-correcting codes and it is also presumably quantum computer resistant. The NTRU cryptosystem was developed in 1996 by J. Hoffstein, J. Pipher and J. H. Silverman. The name NTRU is an abbreviation for N-th degree Truncated Polynomial Ring, or in mathematical notation, $R = \frac{\mathbb{Z}[x]}{(x^n-1)}$, which is its underlying mathematical structure.

Besides being quantum-computer resistant (at least as of this writing), the encryption/decryption speed is faster than that of other practical cryptosystems; in addition, smaller key sizes make the system attractive. Therefore, any application that requires the

aspects of fast performance and/or high-levels of security for the next 10 years would benefit from the NTRU solution [4]. Currently, NTRU satisfies these crucial aspects of security that apply to payment systems, secure messaging and email, mobile eCommerce, vehicle communications(V2V, V2I), military/aerospace, web browsers and servers, remote backup solutions, online presentations/virtual classrooms, utility meters and cloud providers/datacenters [4].

This thesis is structured as follows:

- In Chapter 2 we will review the basics of lattice and ring theory that are needed to understand the NTRU cryptosystem. This includes the LLL-algorithm, Gauss volume-heuristic and the convolution polynomial rings.
- In Chapter 3 we will describe the NTRU cryptosystem parameters, private key and public key, encryption and decryption. We also discuss in detail why its decryption works. In addition, we will give an overview of the theoretical operating characteristics of the NTRU cryptosystem. Lastly, the encryption and decryption speeds of the NTRU cryptosystem are compared against each other.
- In Chapter 4 we will study the main attacks to the NTRU cryptosystem. We will explain the brute force, meet-in-the-middle, and multiple transmission attacks. In addition, we will also discuss lattice based attacks on NTRU.
- In Chapter 5 we will briefly review the RSA, ECC, and McEliece cryptosystems and compare the performances of those cryptosystems with that of NTRU.
- In Chapter 6 we will provide a summary and future research problems.

CHAPTER 2

MATHEMATICAL BACKGROUND

The security of the NTRU cryptosystem is related to the difficulty of finding a short vector of a given lattice. Hence, in this chapter we will review lattices and their properties. In addition, the LLL-algorithm, Gauss Volume-Heuristic and the Convolution Polynomial Ring are also reviewed.

2.1 LATTICES

In this section we review the concepts from the theory of lattices that are needed to describe the NTRU cryptosystem [11].

Definition 2.1. *Let $b_1, b_2, b_3, \ldots, b_k$ be a set of linear independent vectors in $\mathbb{R}^N$. The lattice $\mathcal{L}$ generated by $b_1, b_2, b_3, \ldots, b_k$ is the set of linear combinations of $b_1, b_2, b_3, \ldots, b_k$ with coefficients in $\mathbb{Z}$.*

$$\mathcal{L} = \{a_1 b_1 + a_2 b_2 + a_3 b_3 + \cdots + a_k b_k \mid a_1, a_2, \ldots, a_k \in \mathbb{Z}\}$$

The integers k and N are respectively called the rank and dimension of $\mathcal{L}$. Lattice $\mathcal{L}$ is called full rank when $k = N$ and we assume full rank from here on. We arrange the vectors $b_1, b_2, b_3, \ldots, b_k$ as rows of a matrix B, which is then called the generator matrix of the lattice. The same lattice can have different bases. Assume $b_1, b_2, b_3, \ldots, b_k$ is a basis for a lattice $\mathcal{L}$ and another collection of vectors is $w_1, w_2, w_3, \ldots, w_k \in \mathcal{L}$. Then we can write the w_j as a linear combination of $b_1, b_2, \ldots, b_k$ as follows:

$$\begin{aligned} w_1 &= a_{11} b_1 + a_{12} b_2 + \cdots + a_{1k} b_k, \\ w_2 &= a_{21} b_1 + a_{21} b_2 + \cdots + a_{2k} b_k, \\ &\cdots \\ w_k &= a_{k1} b_1 + a_{k2} b_2 + \cdots + a_{kk} b_k, \end{aligned}$$

Note that all of the a_{ij} coefficients are integers.

Proposition 2.1. *Any two bases for a lattice $\mathcal{L}$ are related by a matrix having integer coefficients and whose determinant equals $+1$ and -1.*

Definition 2.2. *Let $\mathcal{L}$ be a lattice of dimension k and let $b_1, b_2, b_3, \ldots, b_k$ be a basis for $\mathcal{L}$. Then:*

(a) The fundamental region for $\mathcal{L}$ corresponding to this basis is the set

$$F(b_1, b_2, b_3, \ldots, b_k) = \{t_1 b_1 + t_2 b_2 + t_3 b_3 + \cdots + t_k b_k \mid 0 \leq t_i < 1 \; \forall \, i\};$$

(b) The volume of F is called the volume or determinant of $\mathcal{L}$ and is equal to $det(\mathcal{B})$. We denote it by $\det(\mathcal{L})$. This quantity does not depend on the particular choice of the basis for $\mathcal{L}$.

2.2 THE SHORTEST AND CLOSEST VECTOR PROBLEMS

In this section we define and discuss two fundamental computational problems related to lattices [11].

Definition 2.3. Shortest Vector Problem (SVP)*: Given a lattice $\mathcal{L}$, find a shortest nonzero vector in it, i.e, find a nonzero vector $b \in \mathcal{L}$ such that the Euclidean norm $||b||$ is minimum.*

Definition 2.4. Closest Vector Problem (CVP)*: Given a lattice and a vector $v \in \mathcal{R}^n$ that is not in $\mathcal{L}$, find a vector $b \in \mathcal{L}$ that is closest to v, i.e, find a vector $b \in \mathcal{L}$ that minimizes the Euclidean norm $||v - b||$.*

Note that there may be more than one closest vector in a lattice and similarly more than one shortest vector in a lattice. As a result, we look for "a" closest vector, not "the" closest vector. For example, in $\mathbb{Z}^2$, vectors $(0, \pm 1)$ and $(\pm 1, 0)$ are the shortest vectors. Both the SVP and CVP are difficult problems[11]. Moreover, CVP is NP-hard and SVP is NP-hard under a certain randomized reduction hypothesis. The variant of the shortest vector problem and the closest vector problem are the shortest basis problem, approximate shortest vector problem and approximate closest vector problem.

Definition 2.5. Shortest Basis Problem*: Find a basis $b_1, b_2, b_3, \ldots, b_k$ for a lattice that is shortest. For example, we require that max $||b_i||$ or the $\sum_{i=1}^{N} ||b_i||^2$ should be minimized.*

Definition 2.6. Approximate Shortest Vector Problem (apprSVP)*: Let $\theta(k)$ be a function of k. Given a lattice $\mathcal{L}$ of dimension k, find a nonzero vector that is less than or equal to $\theta(k)$ multiplies by a shortest vector, that means if $b_{shortest}$ is a shortest vector in $\mathcal{L}$, find a nonzero vector $b \in \mathcal{L}$ such that*

$$||b|| \leq \theta(k)||b_{shortest}||.$$

Note that each choice of function $\theta(k)$ give us a different apprSVP. It is obvious that appSVP is SVP if $\theta(k) = 1$.

The LLL-Algorithm was invented by A. Lenstra, H. Lenstra and Lovasz.

Let us review this algorithm [11].

Gram-Schmidt orthogonalization process

For any ordered lattice basis $\mathcal{B} = \{b_1, b_2, b_3, \ldots, b_N\} \in \mathbb{R}^N$ the set of corresponding Gram-Schmidt orthogonalized vectors $\mathcal{B}^* = b_1^*, b_2^*, b_3^*, \ldots, b_N^* \subseteq \mathbb{R}^N$ is defined by

$$b_i^* := b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \; for \; 1 \leq i \leq N \tag{2.1}$$

where $\mu_{ij} := \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$ for $1 \leq i \leq N$. If we want to solve the SVP from a given lattice and basis $\mathcal{B}$, we are drawn to use the Gram-Schmidt orthogonalization process above to find an orthogonal basis $\mathbb{B}'$. Given an orthogonal basis, then the shortest non-zero vector is a basis vector.

Note that in the orthogonalization process, we multiply the basis $\mathcal{B}$ by a matrix that is invertible but not all entries are integers. Thus, the lattice $\mathcal{L}'$ generated by $\mathcal{B}'$ is not the same as the original given lattice generated by $\mathcal{B}$.

Remark: Not every lattice has an orthogonal basis; as a result, we have to find a special method that works for these lattices. In 1982, A. Lenstra, H. Lenstra and Lovasz in [1] created the LLL-Algorithm for this purpose.

In order to apply the LLL-Algorithm, we need to know the LLL-reduced and LLL-reduced basis vectors. The following definition and propositions are in [1] of A. Lenstra, H. Lenstra and Lovasz.

Definition 2.7. *Let $\mathcal{L}(b_1, b_2 \ldots, b_N) \in \mathbb{R}^N$ then the basis vectors $b_1, b_2, \ldots, b_N$ are called LLL-reduced with $\frac{1}{4} < \delta < 1$, if $|\mu_{ij}| \leq \frac{1}{2}$ where $1 \leq j \leq N$,*

$$\| b_{i-1}^* \|_2^2 \leq \| b_i^* + \mu_{ii-1} b_{i-1}^* \|_2^2 \; where \; 1 < i \leq N$$

Proposition 2.2. *Let $b_1, b_2, \ldots, b_N$ be a LLL-reduced basis for the lattice $\mathcal{L}$ with $\frac{1}{4} < \delta < 1$. Then*

$$\| b_j \|_2^2 \leq \left(\frac{4}{4\delta - 1} \right)^{i-1} \cdot \| b_i^* \|_2^2 \; where \; 1 \leq j \leq i \leq N$$

$$\det(\mathcal{L}) \leq \prod_{i=1}^{N} \| b_i \|_2 \leq \left(\frac{4}{4\delta - 1} \right)^{N(N-1)/4} \cdot \det(L)$$

$$\| b_i \|_2 \leq \left(\frac{4}{4\delta - 1} \right)^{(N-1)/4} \cdot \det(L)^{1/n}$$

Proposition 2.3. *Let $\mathcal{L}(b_1, b_2, \ldots, b_n)$ be a lattice with LLL-reduced basis vectors $b_1, b_2, \ldots, b_n$. Then*

$$\| b_1 \|_2^2 \leq \left(\frac{4}{4\delta - 1} \right)^{N-1} \cdot \| v \|_2^2 \;\; for \; all \; v \in L \; where \; v \neq (0, 0, \ldots, 0)$$

Note that LLL-Algorithm calculates a LLL-reduced basis of a lattice and it will terminate after finitely many steps.

2.3 GAUSS VOLUME-HEURISTIC

In this section, we will review the Gauss Volume-Heuristic [11].
Given a lattice $\mathcal{L}$ and $S \subseteq span(\mathcal{L})$ a measurable set, the expected value is

$E[\#(S \cap \mathcal{L})] = \frac{\text{vol}(S)}{\det \mathcal{L}}$

Note that for each lattice vector there exists some fundamental region. Thus, the expected number of lattice points in S is equal to the volume of the fundamental region $vol(\mathcal{L})$ divide k the volume of S.

Theorem 2.1. *Let $\mathbb{B}_R$ be a ball with radius R and center 0 in $\mathbb{R}^N$. Then the volume of $\mathbb{B}_R$ is*

$$vol(\mathbb{B}_R) = \frac{\pi^{N/2} R^N}{\Gamma\left(1 + \frac{N}{2}\right)}$$

where Γ denoted the volume of the ball $\mathbb{B}_R$ in $\mathbb{R}^N$ is approximation given by

$$vol(\mathbb{B}_R)^{\frac{1}{N}} \approx \sqrt{\frac{2\pi e}{N}} R$$

Definition 2.8. *Let $\mathcal{L}$ be a lattice in $\mathcal{R}^N$. Then the Gaussian expected shortest length is*

$$s \approx \sqrt{\frac{N}{2\pi e}} (\det \mathcal{L})^{\frac{1}{N}}$$

This Gaussian heuristic states that a shortest nonzero vector in a random lattice will satisfy

$$\| \, b_{shortest} \, \| \approx s$$

2.4 Convolution Polynomial Rings

The Convolution Polynomial Ring is a special ring that is used by the NTRU cryptosystem. As a result, we will review them now [3].

Definition 2.9. *For a positive integer n, the ring of convolution polynomial is the quotient ring*

$$\mathcal{R} = \frac{\mathbb{Z}[x]}{(x^n - 1)}$$

Similarly, the ring of convolution polynomials (modulo q) is also the quotient ring

$$\mathcal{R}_q = \frac{(\mathbb{Z}/q\mathbb{Z})[x]}{(x^n - 1)}$$

Every element of $\mathcal{R}$ or $\mathcal{R}_q$ has a unique representation of the form $\mathbf{f} = f_0 + f_1 x + \ldots + f_{n-1} x^{n-1}$ with the coefficients in $\mathbb{Z}$ or $\mathbb{Z}/q\mathbb{Z}$ respectively. Since in the ring $\mathcal{R}$ we know that $x^n \equiv 1$, the multiplication of two elements $f, g \in \mathcal{R}$ is given by $h = f \star g = f \cdot g \, (mod \; x^n - 1)$. Note that it is a lot easier to compute in the ring $\mathcal{R}$ and $\mathcal{R}_q$ then it is in the general polynomial quotient rings because the polynomial $x^n - 1$ has a simple form. This explanation makes sense because whenever we mod out by $x^n - 1$, we simply

require x^n to equal 1. A coefficient h_k of the product is calculated as

$$h_k = \sum_{i+j=k \mod n} f_i \star g_i \tag{2.2}$$

$$= \sum_{i=0}^{k} f_i \cdot g_{k-i} + \sum_{i=k+1}^{n-1} f_i \cdot g_{n+k-i} \tag{2.3}$$

Note that a vector $\mathbf{f} = (f_0, f_1, \ldots, f_{n-1})$ represents the polynomial
$f = f_0 + f_1 x + \ldots, + f_{n-1} x^{n-1}$ in $\mathcal{R}$.
For example:

$$(x^5 + 2x^2 + 1) \star (3x^4 + x^2) = 3x^9 + x^7 + 6x^6 + 5x^4 + x^2 \tag{2.4}$$

$$= 3x^2 + 1 + 6x^6 + 5x^4 + x^2 \tag{2.5}$$

$$= 6x^6 + 5x^4 + 4x^2 + 1 \tag{2.6}$$

in the polynomial ring $\mathbb{Z}[x]/(x^7 - 1)$.

Since we work in the ring, let us review the basis definition of the width and norm in [3].

Definition 2.10. *The width of an element $f = f_0 + f_1 x + \ldots, f_{n-1} x^{n-1}$ in $\mathcal{R}$ is defined by*

$$|f_\infty| = \max_{1 \le i \le n} \{f_i\} - \min_{1 \le i \le n} \{f_i\}$$

Definition 2.11. *The centered norm of an element $f \in \mathcal{R}$ is defined by*

$$\| f \| \perp = \left(\sum_{i=0}^{n} (f_i - \bar{f})^2 \right)^{\frac{1}{2}} \textit{ where } \bar{f} := \frac{1}{n} \sum_{i=0}^{n-1} f_i$$

Note that $||f||_\perp / \sqrt{n}$ is the standard deviation of the coefficients of f.

Proposition 2.4. *For $\epsilon > 0$ there are constants $\gamma_1, \gamma_2 > 0$, depending on ϵ and n, such that for any randomly chosen polynomials $f, g \in \mathcal{R}$, the probability that they satisfy*

$$\gamma_1 \| f \|_2 \| g \|_2 \le |f \cdot g| \le \gamma_2 \| f \|_2 \| g \|_2 \textit{ is greater than } 1 - \epsilon$$

The proposition above will be ineffective if the ratio γ_2/γ_1 is very large for a small ϵ.

CHAPTER 3

NTRU CRYPTOSYSTEM

In this chapter, we will define the parameters used in the cryptosystem. Additionally, we also examine the encryption and decryption of the NTRU cryptosystem. Finally, we will compare the encryption and decryption time of this cryptosystem.

3.1 PARAMETERS

The basis parameters that we use in the NTRU cryptosystem are n, p and q. The parameter n is the degree of the polynomials used in the convolution polynomial rings. The number n is preferred to be prime because it will give the NTRU cryptosystem improved security. The modulus p and the modulus q are relatively prime in $\mathcal{R}$, the modulus p has to be smaller than the modulus q, and the modulus q is usually smaller than n.

Notation:

- $\mathcal{L}_m$: Set of polynomials in $\mathcal{R}_p$ whose coefficients lie between $-\frac{1}{2}(p-1)$ and $\frac{1}{2}(p-1)$.
- $\mathcal{L}_f$: Set of polynomials in $\mathcal{R}$ having $(d+1)$ 1s and d -1s.
- $\mathcal{L}_g$: Set of polynomials in $\mathcal{R}$ having d 1s and d -1s.
- $\mathcal{L}_r$: Set of polynomials in $\mathcal{R}$ having d 1s and d -1s.

We know that $f \in \mathcal{L}_f$, $g \in \mathcal{L}_g$, and $r \in \mathcal{L}_r$ have $\mathcal{L}^2$ norms $|f|_2 = \sqrt{2d_f - 1 - n^{-1}}$, $|g|_2 = \sqrt{2d_g}$ and $|r|_2 = \sqrt{2d_r}$

Table 3.1 shows us the recommended parameters in NTRU [17].

Table 3.1. NTRU parameters

$k(bit\ security)$	n	p	q	$d(\#\ 1s\ in\ f\ and\ r)$	$d_g(\#\ 1s\ in\ g)$	$d_{mo}(\#\ 1s\ in\ m)$
80	251	2	197	48	125	70
112	347	2	269	66	173	108
128	397	2	307	74	198	128
160	491	2	367	91	245	167
192	587	2	439	108	293	208
256	787	2	587	140	393	294

Kevin [3] shows us how to define the private key and the public key.

3.2 PRIVATE KEY

To create the private key, Bob starts by randomly choosing $f \in \mathcal{L}_f$ such that f is invertible modulo q and p, which means the inverse in $\mathcal{R}_q$ and $\mathcal{R}_p$ exist. The private key is consists of either the polynomial f or the pair (f, f_p^{-1}). Bob needs to double check that f is invertible modulo p; afterward, he computes and stores it. The inverse of $f(mod\ p)$ is defined by $f_p^{-1} \star f \equiv 1\ mod\ p$. Similarly, the inverse of $f(mod\ q)$ is defined by $f_q^{-1} \star f \equiv 1\ mod\ q$. Therefore, the private key is: f or (f, f_p^{-1}).

3.3 PUBLIC KEY

To create the public key, Bob starts by randomly choosing $g \in \mathcal{L}_g$. Note that g does not have to be invertible in the ring $\mathcal{R}_p$ and $\mathcal{R}_q$. The public key is the polynomial h where $h \equiv pf_q^{-1} \star g\ (mod\ q)$.

Example of private key and public key

Given public parameters $(n, p, q, d) = (7, 3, 41, 2)$

1) Bob will randomly choose $f \in \mathcal{L}_f$,

$f(x) = x^6 - x^4 + x^3 + x^2 - 1$ (private key of Bob)

and $g \in \mathcal{L}_g$, $g(x) = x^6 + x^4 - x^2 - x$

2) He computes the inverses

$f^{-1}(x) mod\ q = 8x^6 + 26x^5 + 31x^4 + 21x^3 + 40x^2 + 2x + 37 \in \mathcal{R}_q$ and

$f^{-1}(x) mod\ p = x^6 + 2x^5 + x^3 + x^2 + x + 1 \in \mathcal{R}_p$ (private key of Bob)

3) He stores $(f(x), f_p^{-1}(x))$ as his private key and computes his public key

$h(x) \equiv p \star f_q^{-1}(x)(mod\ q) \star g(mod\ q) = 19x^6 + 38x^5 + 6x^4 + 32x^3 + 24x^2 + 37x + 8 \in \mathcal{R}_q$

3.4 ENCRYPTION

If Alice desires to send the message $m \in \mathcal{L}_m$ to Bob, then she randomly chooses a polynomial $r \in \mathcal{L}_r$ and computes $e \equiv pr \star h + m\ (mod\ q)$.

Example of Encryption

- Alice decides to send Bob the message $m(x) = -x^5 + x^3 + x^2 - x + 1$
- Using the ephemeral key $r(x) = x^6 - x^5 + x - 1$
- Since $e \equiv pr \star h + m\ (mod\ q)$,
- Then $e(x) \equiv 31x^6 + 19x^5 + 4x^4 + 2x^3 + 40x^2 + 3x + 25\ (mod\ 41)$

3.5 DECRYPTION

Once Bob receives Alice's ciphertext e, he starts the decryption process by calculating $a(x) \equiv f \star e\ \pmod q$. He then centerlifts $a(x)\ modulo\ q$ to obtain $b(x)$. Finally, he reduces $b(x)\ modulo\ p$ and computes $c \equiv f_p^{-1} \star b \bmod p$ and then he centerlifts $c(x)\ modulo\ p$ to

retrieve Alice's original plaintext $m(x)$.

Example of Decryption

1) $a \equiv f \star e \ (mod\ q)$,

Bob computes $a \equiv x^6 + 10x^5 + 33x^4 + 40x^3 + 40x^2 + x + 40 \ (mod\ 41)$

2) He then centerlifts modulo q to obtain

$b \equiv a \ (mod\ q) = x^6 + 10x^5 - 8x^4 - x^3 - x^2 + x - 1 \ (mod\ 3)$

3) Finally, he reduces $b(x)\ modulo\ p$ and computes

$c \equiv f_p^{-1}(x) \star b(x) \equiv 2x^5 + x^3 + x^2 + 2x + 1 \ (mod\ 3)$ centerlifting modulo p to retrieve Alice's plaintext $m(x) = -x^5 + x^3 + x^2 - x + 1$

3.6 Why Decryption Works

During the decryption process, we use the public key h and the private key pair (f, f_p^{-1}). The encrypted message is $e \equiv pr \star h + m \ (mod\ q)$, and it is the sum of the original message m with the scaled version of the public key h. Out of the parameter, Bob only has the value of public key h, which he has to use mathematical methods to recover the message. Now, we can substitute $h \equiv pf_q^{-1} \star g \ (mod\ q)$ into the encrypted message to check what the encrypted message m will equal to. Thus, $e \equiv r \star (pf_q^{-1} \star g) + m \ (mod\ q)$.

Note that the coefficient of the polynomials r, g, m and modulus p are very small compared to the coefficient of the modulus q. Now the encrypted message multiply the private key f to eliminate the function f_q. Recall that all of the computation are performed in the convolution polynomial ring.

If encrypted message is

$$e \equiv r \star h + m \ (mod\ q) \tag{3.1}$$

$$\equiv r \star (pf_q^{-1} \star g) + m \ (mod\ q) \quad (since\ h \equiv pf_q^{-1} \star g \ (mod\ q)) \tag{3.2}$$

multiply the encrypted message by the private key f

$$a \equiv f \star e \equiv f \star (r \star pf_q^{-1} \star g) + f \star m \ (mod\ q) \tag{3.3}$$

$$a \equiv f \star e \equiv pr \star g + f \star m \ (mod\ q) \quad (since\ f \star f_q^{-1} \equiv 1\ mod\ q) \tag{3.4}$$

Note that the modulus q is a lot bigger than the private key f. Thus the value of a, containing polynomials and a scalar, is very small compared to the modulus q. Suppose all the coefficients of the polynomial $pr \star g + f \star m$ are lying in the interval $(-q/2, q/2]$ which, in other words, there is no change if we reduce the polynomial modulo q; then, polynomial a is exactly $pr \star g + f \star m$. In order to recover the original message, we multiply the polynomial a by the second private key f_p^{-1}.

$$a \equiv pr \star g + f \star m \ (mod\ q)$$

$$b \equiv pr \star g + f \star m \ (mod\, p) \quad (reduce\ modulo\ p) \tag{3.5}$$
$$\equiv f \star m \ (mod\, p)(pr \star g\ is\ reduced\ to\ zero) \tag{3.6}$$

$$c \equiv f_p^{-1} \star b \ (multiply\ b\ with\ the\ second\ private\ key\ f_p^{-1}) \tag{3.7}$$
$$\equiv f_p^{-1} \star f \star m \ (mod\, q) \quad (since\ b \equiv f \star m \ (mod\, p)) \tag{3.8}$$
$$\equiv m \ (mod\, p) \quad (since f_p^{-1} \star f \equiv 1 \ (mod\, p)) \tag{3.9}$$

Note that the decryption will fail if all the coefficients of the polynomial $pr \star g + f \star m$ do not lie in the interval $(-q/2, q/2]$.

Proposition 3.1. *[11] If the NTRU parameters (n, p, q, d) are chosen to satisfy $q > (6d + 1)p$, then this inequality guarantees that the decryption process will never fail.*

For example: NTRU with parameters (n, p, q, d)=(7,3,41,2)

Then, $41 = q > (6d + 1)p = (6(2) + 1)(3) = 39$. So, proposition guarantees that decryption will work.

3.7 THEORETICAL OPERATING SPECIFICATIONS

In this section we give a summary of the theoretical operating characteristics of the NTRU PCKS [8]. There are three integer parameters (n,p,q) as described before. Table 3.2 summarizes the NTRU PCKS characteristics in terms of these parameters.

Table 3.2. Theoretical Operating Specifications

Operation Characteristics	Formula
$Plain\ Text\ Block$	$(n-k)\log_2 p\ bits$
$Encrypted\ Text\ Block$	$n\log_2 q\ bits$
$Encryption\ Speed$	$O(n^2)\ operations$
$Decryption\ Speed$	$O(n^2)\ operations$
$Message\ Expansion$	$[n/(n-k)]\log_p\ q - to - 1$
$Private\ Key\ Length$	$2n\log_2 p\ bits$
$Public\ Key\ Length$	$n\log_2 q\ bits$

3.8 ENCRYPTION AND DECRYPTION SPEEDS

In cryptography, encryption is the process of scrambling messages or data in a form that only the intended participants can decode it. Encryption does not guarantee the deterrence of hacking; however it minimizes the possibility that the hacker will be able to intercept and interpret the data that is encrypted. Decryption is the process of converting data that has been made incomprehensible through encryption returning it to its unencrypted form.

In decryption, the system obtains and converts the distorted data and changes it to text and images that are clearly comprehensive not only by the selected readers but also by the system. The NTRU encryption and decryption execution timings before modification [20] are shown in Table 3.3.

Table 3.3. Encryption and Decryption of NTRU

Text sizes (bits)	Encryption	Decryption
128	0.000019	0.000019
256	0.000018	0.049
512	0.05697	0.048
1024	0.189	0.0601
2048	0.279	0.0612
5120	0.65895	0.19586

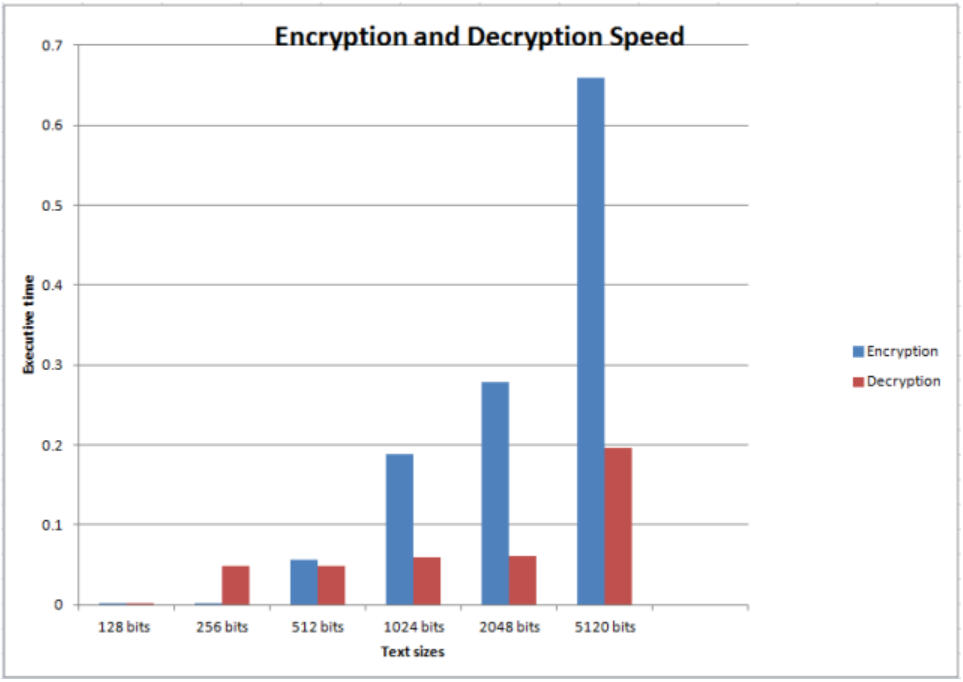

Figure 3.1. Encryption and Decryption Speeds of NTRU

From Table 3.3 and Figure 3.1, we can conclude that the execution time for NTRU decryption is a lot faster than the NTRU encryption.

CHAPTER 4

SECURITY ANALYSIS OF NTRU

In this chapter, we will introduce the alternate private keys [9]. We also examine the brute force attack [10] and an improved attack called meet-in-the-middle attack. Furthermore, we will analyze why we should not send a message multiple times with the same public key h in the multiple transmission attack. Finally, we will show the interesting attack on NTRU called the lattice attack.

4.1 ALTERNATE PRIVATE KEYS

The alternate keys f' can be used to decrypt the same message as f. Let f' be any rotation of the private key f, then $f' = x^i \star f$ for $i \in \{1, 2, ..., n-2\}$. We know that $g \equiv f \star h \; mod \; q$, thus we will have $x^i g \equiv x^i f \star h \; mod \; q$. Therefore, the polynomial f' can decrypt the same message as f. Let g' be the rotation of g, then $g' = x^i \star g$. Since $g \equiv f \star h \; mod \; q$ then $g' \equiv f' \star h \; mod \; q$. Assume we want to decrypt e, we will compute $a' = f' \star e = f' \star (pr \star h + m) = x^i \star f[pr \star (f_q^{-1} \star g) + m] = x^i \star (f \star pr \star f_q^{-1} \star g + f \star m] = x^i \star (pr \star g + f \star m) = pr \star g' + f' \star m$. Since $a' = x^i(pr \star g + f \star m) = x^i \star a$ thus, if all the coefficients of a lie in the interval $(-q/2, q/2]$ then so do the coefficients of a'. Now we compute $f_p^{-1'}(x) \star a' \equiv p f_p^{-1'}(x) \star r \star g' + f_p^{-1'}(x) \star f' \star m \equiv m \; mod \; p$. Therefore, we recovered the message m. In general, any $(f', g') \in R^2$ such that $g' \equiv f' \star h \; mod \; q$ has an inverse modulo p that can be used to decrypt the message if the coefficients of $a' = pr \star g' + f' \star m$ are in the interval $(-q/2, q/2]$. The pairs (f', g') are called alternate keys.

4.2 BRUTE FORCE ATTACKS

An attacker can recover an alternate key or the private key by trying all possible $f' \in \mathcal{L}_f$. Since $f \star h \equiv g \bmod q$, we will test to see if $f' \star h \mod q$ has small coefficients compared to q, preferred that the most coefficients are equal to 0 and the others only 1. An attacker can search for $g' \in \mathcal{L}_g$ and test to see if $g' \star h^{-1} \bmod q$ has small coefficients compared to q. Similarly, an attacker can search for $r' \in \mathcal{L}_r$ and test to see if $e - pr' \star h \bmod q$ has small coefficients. Usually $\mathcal{L}_f$ is bigger than $\mathcal{L}_g$, thus the key security is determined by $\#\mathcal{L}_g$, and an individual message security is determine by $\#\mathcal{L}_r$. A more efficient attack is shown in the next section. This attack is called a meet-in-the-middle attack.

In general, this attack will cut the search time by square root. Thus, the security level is

$$\text{key security } = \sqrt{\#\mathcal{L}_g} = \sqrt{\binom{n}{d_g}\binom{n-d_g}{d_g}} = \frac{1}{d_g!}\sqrt{\frac{n!}{(n-2d_g)!}}$$

$$\text{message security } = \sqrt{\#\mathcal{L}_r} = \sqrt{\binom{n}{d_r}\binom{n-d_r}{d_r}} = \frac{1}{d_r!}\sqrt{\frac{n!}{(n-2d_r)!}}$$

4.3 MEET-IN-THE-MIDDLE ATTACKS

This technique was originally invented by Andrew Odlyzko and was later investigated in more detail by Nick Howgrave-Graham, Joseph H. Silverman and William Whyte [14]. The idea was to search for f in the form $f = f_1 + f_2$, where both polynomials f_1 and f_2, have length $n/2$ with $d_f/2$ ones. Now let us look at the following properties below

$$f \star h \equiv g \bmod q$$

then

$$(f_1 + f_2) \star h \equiv g \; mod \; q$$

$$f_1 \star h \equiv g - f_2 \star h \; mod \; q$$

Note that g is a binary polynomial; in other words, the coefficients of $f_1 \star h$ and $f_2 \star h$ will be different by either 0 or 1 modulo q. If we can find an $f' \in \mathcal{L}_f$ where $g' \equiv f' \star h \; mod \; q$ is binary, then we have found at least one rotation of f or another key that is able to decrypt messages.

1) Enumerate the vectors f_1.

There are $\binom{n/2}{d_f/2}$ polynomials of length $n/2$ with exactly $d_f/2$ ones and the rest of its coefficients are equal to zero. We put each f_1 into a bin based on the most significant bit of the first k coordinates of the binary representation of $f_1 \star h \; mod \; q$. Each bin is then referenced by $\{0,1\}^k$ so there are a total of 2^k bins.

For example, let $n = 4$ and $q = 8$. Now, suppose $f_1 \star h \; mod \; q \equiv 7 + 2x + 3x^2 + 5x^3 + 6x^4$. We are only interested in the first four coordinates. We can write each of their coordinates in binary notation.

$$7 = (\mathbf{1}11)_2$$

$$2 = (\mathbf{0}10)_2$$

$$3 = (\mathbf{0}11)_2$$

$$5 = (\mathbf{1}01)_2$$

So, the polynomial f_1 will put into the bin referenced $[1001]$.

2) Enumerate the vector f_2.

There are also $\begin{pmatrix} n/2 \\ d_f/2 \end{pmatrix}$ steps. For each f_2 we need to check if $-f_2 \star h$ corresponds to a bin occupied by an f_1. That means the leading bit of first k coefficients of $-f_2 \star h \mod q$ are equal to $f_1 \star h \; mod \; q$. We want to find f_1 and f_2 where $(f_1 \star h)_i \equiv \{0, 1\} - (f_2 \star h)_i \; mod \; q$. As a result, we want to check the bins that correspond to $-f_2 \star h$ and also the bins that correspond to adding ones to any of the coefficients of $-f_2 \star h$.

For example, let $n = 4$ and $q = 8$, and the first four terms in $-f_2 \star h$ are $7 + 2x + 3x^2 + 5x^3$. The bin representing $-f_2 \star h$ is $[1001]$. If we add one to seven, we get eight, and the leading bit changes from one to zero. If we add one to two, we get three, and the leading bit remains zero. If we add one to three, we get four, and the leading bit change from zero to one. Finally, if we add one to five, the leading bit remains one. So, we have to check the bin $[1001]$ as well as the bins $[0001]$, $[1011]$, $[0011]$.

3) Check for matches.

Once we have found that f_2 hit an occupied bin then we compute $f = f_1 + f_2$ and need to check if the coefficients of $f \star h \; mod \; q$ are binary. If it is a binary, then we complete the algorithm because we can found a private key. If it is not a binary, then we proceed to the next f_2. If the bin is occupied by more than one f_1, then we need to check f_2 for each f_1 in the bin.

4) Running time needed for the meet-in-the-middle attack.

Let τ be the time to calculate $f_1 \star h \; mod \; q$. And the τ_l be the time either to look up or to store in the bin. Then, the expected time to run enumerate f_1 will be

$$\tau_1 = \begin{pmatrix} n/2 \\ d_f/2 \end{pmatrix} (\tau_c + \tau_l)$$

In addition, an upper bound of the expected time to run enumerate f_2 is no more than

$$\tau_2 = \#(f_2) \star (\tau_c + (expect \; different \; bins \; per \; f_2) \star \tau_l + (expect \; hits \; per \; f_2) \star \tau_c) \quad (4.1)$$

$$= \begin{pmatrix} n/2 \\ d_f/2 \end{pmatrix} \left(\tau_c + \frac{2k}{q} \tau_l + \left(\frac{\binom{n/2}{d_f/2}}{2^k} \right) \tau_c \right) \quad (4.2)$$

4.4 MULTIPLE TRANSMISSION ATTACKS

The main idea of this attack is to show us that it is not a good idea to send a message several times with the same public key h [7]. Alice sends a message m several times to Bob,

with the same public key, but she changes the random elements $r \in \mathcal{L}_r$ each time. In other words, we have n different random elements which we denote by r_i where $i = 1, 2, \ldots, n$ and then Alice calculates the different ciphertext which are given by
$e_i \equiv pr_i \star h + m \ (mod\ q)\ for\ i = 1, 2, \ldots, n$ and then she sends them to Bob. As a result, Eve intercepts these different ciphertext and gets all e_i where $i = 1, 2, \ldots, n$ and calculates

$$(e_i - e_j) \star (ph)^{-1} \ (mod\ q) \text{ where } 1 \leq i < j \leq n$$

and gets

$$r_i - r_j \ (mod\ q) \text{ where } 1 \leq i < j \leq n$$

Note that the coefficients of $r_i - r_j$ range from -2 to 2, thus Eve recovers $r_i - r_j$ with high probability. Now let us analyze the possibilities for some particular coefficient of $r_i - r_j$, say the k^{th} coefficient. If we let
θ = the k^{th} coefficient of r_i,
β = the k^{th} coefficient of r_j,
γ = the k^{th} coefficient of $r_i - r_j$,
The possibilities for $\gamma = \theta - \beta$ are listed in the following table:

Table 4.1. Coefficient of $r_i - r_j$

$\gamma = \theta - \beta$	-1	0	1
-1	0	-1	-2
0	1	0	-1
1	2	1	0

If $\gamma = 2$ (respectively $\gamma = -2$), then Eve deduces that $\beta = -1$ (respectively $\beta = 1$). Thus, Eve will be able to recover approximately $2/9$ of the coefficients of r_i. Moreover, if $\gamma = 1$ (respectively $\gamma = -1$), Eve deduces that $\beta = -1, 0$ (respectively $\beta = 0, 1$). So Eve will be able to recover approximately $4/9$ of the coefficients of r_i.
So each of the additional transmissions $e_2, e_3, ..., e_k$ will enable Eve to determine approximately $2/9$ of the coefficients of r_i, and will also give information about another $4/9$ of the coefficients. Therefore, it takes only a few transmissions for Eve to determine every coefficient of r_i, and then a brute force search on the remaining coefficients will allow Eve to recover r_i and the message m.

For example:
Let $(n, p, q) = (7, 3, 41)$.

Alice sends a message $m = -x^5 + x^3 + x^2 - x + 1$ several times to Bob, with the same public key $h(x) \equiv 30x^6 + 26x^5 + 25x^4 + 8x^3 + 18x^2 + 32x + 25$, but she changes the random elements r each time where
$r_1 = x^6 - x^5 + x - 1$,
$r_2 = x^5 - x^4 + x - 1$,
$r_3 = x^4 - x^3 + x - 1$,
$r_4 = x^4 - x^3 - x + 1$, and
$r_5 = x^6 - x^3 - x + 1$
Then Alice calculates the different ciphertexts which are given by $e_i \equiv pr_i \cdot h + m \pmod q$ for $i = 1, 2, \ldots, k$. Thus,
$e_1 \equiv 8x^6 + 11x^5 + 19x^4 + 28x^3 + 33x^2 + 8x + 17$,
$e_2 \equiv 30x^6 + 16x^5 + 5x^4 + 19x^3 + 40x^2 + 9x + 5$,
$e_3 \equiv 18x^6 + 38x^5 + 10x^4 + 5x^3 + 31x^2 + 16x + 6$,
$e_4 \equiv x^6 + 3x^5 + 30x^4 + 27x^3 + 29x^2 + 17x + 17$, and
$e_5 \equiv 22x^6 + 38x^5 + 33x^4 + 12x^3 + 16x^2 + 37x + 17$ and then she sends them to Bob.

Eve intercepts these different ciphertext and calculates

$$(e_2 - e_1) \star (ph)^{-1} \bmod q \equiv$$
$$(e_2 - e_1)(14)(12x^6 + 4x^5 + 35x^3 + 6x^2 + 6x + 8) \bmod (x^n - 1) \equiv$$
$$(r_2 - r_1) \bmod q \equiv$$
$$-x^6 + 2x^5 - x^4.$$

Note that the inverse of 3 in mod 41 is 14 and we calculate h^{-1} by using the pseudo-inverse of h technique. Similarly,
$r_3 - r_1 = -x^6 + x^5 + x^4 - x^3$,
$r_4 - r_1 = -x^6 + x^5 + x^4 - x^3 - 2x + 2$, and
$r_5 - r_1 = x^5 - x^3 - 2x + 2$

Coefficient of x^6 in $r_2 - r_1 = -1 \Rightarrow coefficient\ of\ x^6\ in\ r_1 = 0\ or\ 1$
Coefficient of x^6 in $r_5 - r_1 = 0 \Rightarrow coefficient\ of\ x^6\ in\ r_1 = -1, 0, 1$
Coefficient of x^5 in $r_2 - r_1 = 2 \Rightarrow coefficient\ of\ x^5\ in\ r_1 = -1$
Coefficient of x^4 in $r_2 - r_1 = -1 \Rightarrow coefficient\ of\ x^4\ in\ r_1 = 0\ or\ 1$
Coefficient of x^4 in $r_3 - r_1 = -1 \Rightarrow coefficient\ of\ x^4\ in\ r_1 = 0\ or\ -1$
Coefficient of x^3 in $r_2 - r_1 = 0 \Rightarrow coefficient\ of\ x^3\ in\ r_1 = -1, 0, 1$
Coefficient of x^2 in $r_2 - r_1 = 0 \Rightarrow coefficient\ of\ x^2\ in\ r_1 = -1, 0, 1$
Coefficient of x in $r_4 - r_1 = -2 \Rightarrow coefficient\ of\ x\ in\ r_1 = 1$

Coefficient of constant term in $r_4 - r_1 = 2 \Rightarrow constant\ term\ in\ r_1 = -1$

Then a brute force search on the remaining coefficients will allow Eve to recover r_i and the message m.

4.5 LATTICE ATTACKS

In this section we will analyze the known lattice attack on the public key h. We also will review the lattice reduction. The purpose of lattice reduction is to find the "small" vector in a given lattice. In general, the smallest vector can be found by a very large search; however, this is impossible if the dimensions are very large. A. Lenstra, H. Lenstra and Lovasz invented the LLL-Algorithm, including many different improvements with Schnorr and others, that will find small vectors in polynomial time. However, the LLL-Algorithm takes a long time to find the smallest vector. Note that the smallest vector is not too much smaller than the expected length of the smallest vector. Therefore, we will look at the lattice attack [10].

Consider the $2n \times 2n$ matrix that is composed of $4\ n \times n$ blocks.

Let $\mathcal{L} =$

$$\left[\begin{array}{cccc|cccc} \alpha & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{n-1} \\ 0 & \alpha & \cdots & 0 & h_{n-1} & h_0 & \cdots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right]$$

Let $\mathcal{L}$ be the lattice generated by the row of this matrix or simply

$$\mathcal{L}^{NT} = \begin{bmatrix} \alpha & H \\ 0 & qI \end{bmatrix}$$

Note that $\mathcal{L}$ is composed of $4\ n \times n$ blocks

- upper left block=α times the identity matrix
- lower left block=zero matrix
- lower right block=q times the identity matrix
- upper right block=cyclical permutations of the coefficients of $h(x)$

Then, the $\det(\mathcal{L}) = q^n \alpha^n$.

Recall the public key h is $h \equiv p f_q^{-1} \star g (mod\ q)$, so the lattice $\mathcal{L}$ will contain the vector $\tau = (\alpha f, g)$. That means the $2n$ vector consisting of the n coefficients of f, multiplied by α,

is followed by the n coefficients of g.

From the Gaussian heuristic earlier, we know that expected length of the shortest non-zero vector is approximately s where

$$s \approx \sqrt{\frac{dim(\mathcal{L})}{2\pi e}} (\det \mathcal{L})^{\frac{1}{N}}$$

For our case, $dim(\mathcal{L}) = N = 2n$ and $det(\mathcal{L}) = q^n \alpha^n$. Thus, the expected length of the shortest non-zero vector is approximately $s = \sqrt{\frac{n\alpha q}{\pi e}}$.

Recall that lattice reduction techniques will have a good chance of finding the target vector of length τ – or a vector whose length is close to τ – if τ is much shorter than the length of the expected shortest non-zero vector in the lattice. Therefore, we want to choose α to maximize the ratio $s/|\tau|_2$. Squaring this ratio, we see that we should choose α so as to maximize

$$\frac{\alpha}{\alpha^2 |f|_2^2 + |g|_2^2} = (\alpha |f|_2^2 + \alpha^{-1} |g|_2^2)^{-1} \Rightarrow \alpha = \frac{|g|_2}{|f|_2}$$

So we chose $\alpha = \frac{|g|_2}{|f|_2}$ and we define a constant c_h by setting $|\tau|_2 = c_h s$. As a result, c_h is the ratio of the length of the target vector to the length of the expected shortest vector. The smaller the value of c_h, the easier it will be to find the target vector. Now we will calculate the value of c_h,

since $\tau = (\alpha f, g), |\tau|_2^2 = \alpha^2 |f|_2^2 + |g|_2^2$. Then

$$\begin{aligned}
c_h &= \frac{|\tau|_2}{s} && (4.3)\\
&= \frac{\sqrt{\alpha^2 |f|_2^2 + |g|_2^2}}{s} && (4.4)\\
&= \frac{\sqrt{2|g|_2^2}}{s} \ (since\ \alpha = \frac{|g|_2}{|f|_2}) && (4.5)\\
&= \sqrt{2|g|_2^2} \times \sqrt{\frac{\pi e}{2n|g|_2 \times q/|f|_2}} && (4.6)\\
&= \sqrt{\frac{2\pi e |f|_2 |g|_2}{2nq}} && (4.7)\\
& && (4.8)
\end{aligned}$$

thus $c_h = \sqrt{\frac{2\pi |f|_2 |g|_2}{Nq}}$ $(since\ N = 2n)$

CHAPTER 5

COMPARISON OF NTRU WITH OTHER PUBLIC KEY CRYPTOSYSTEMS

Public-key cryptography uses two separate keys for encryption and decryption. Even though it is not the same key, the two parts of this key pair are mathematically linked. The public key is used to encrypt plaintext or to verify a digital signature which can be announced to the public. The private key used to decrypt ciphertext or to create a digital signature must be kept secret. Public key algorithm are usually based on the computational complexity of hard problems. For example, the RSA cryptosystem is based on the difficulty of the integer factorization problem, Elliptic curve cryptography (ECC) is based on the difficulty of number theoretic problems involving elliptic curves, and the McEliece cryptosystem based on error correcting codes. In this chapter, we will examine the comparison of the NTRU against RSA and ECC in terms of public key size, key generation, encryption time and decryption time. For these comparisons, we will code NTRU cryptosystem compatible with "IEEE 136.1 Draft Standard for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices" standard. Lastly, we will compare NTRU against the McEliece cryptosystem in terms of encryption time and decryption time.

5.1 RSA CRYPTOSYSTEM

RSA is one of the first public key cryptosystem. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman and was proposed in 1977. The three main steps of RSA algorithm are key generation, encryption and decryption [6].

Key Generation:

- choose two prime numbers p and q with $p \neq q$
- compute $n = pq$
- compute $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1)$ such that ϕ is Euler's totient function
- select an integer e where $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$
- compute d where $d = e^{-1}(mod\ \phi(n))$, d is the multiplicative inverse of $e(modulo\ \phi(n))$
- The public key is (n, e) and the private key is d.

Encryption:

- Alice sends the public key (n, e) to Bob.
- Bob then sends the message M to Alice.

- But he has to turns M into an integer m where $0 \leq m < n$.
- Then he calculates the ciphertext c such that $c = m^e (mod\ n)$.
- Bob then sends the ciphertext c to Alice.

Decryption:
Now Alice can recover m from the ciphertext c by using her private key exponent d via computing $m = c^d (mod\ n)$.
Example:
let $p = 11$, $q = 13$
so $n = pq = 11 \times 13 = 143$
$\phi(n) = (p-1)(q-1) = 10 \times 12 = 120$
select an integer $e\ where\ 1 < e < \phi(n)\ and\ \gcd(e, \phi(n))$, so let choose $e = 7$ then compute d where $d = e^{-1} (mod\ \phi(n))$. By using the Euclidean Algorithm then $d = 103$.
let check, $e.d = 7.103\ mod\ (120) = 1\ mod\ 120$, now pick the plaintext message m, $m = 9$
for encryption, $c = m^e (mod\ n) = 9^7 mod\ 113 = 48$
for decryption, $m = c^d (mod\ n) = 48^{103} (mod\ 143) = 9$

5.2 ELLIPTIC CURVE CRYPTOGRAPHY

In 1985 Victor Miller and Neil Koblitz invented the Elliptic Curve Cryptography [6]. Assume p is a prime and F_p is the field of integers modulo p; an elliptic curve E over F_p is given as $y^2 = x^3 + ax + b$ (1)
where $a, b \in F_p$ sastisfy $4a^3 + 27b^2 \neq 0 (mod\ p)$. Let $x, y \in F_p$, then (x, y) is a point on the curve if (x, y) satisfies equation (1). Let ∞ denote the point of infinity that is to be on the curve. $E(F_p)$ is the set of all the points on E.
Key Generation:
The cyclic subgroup of $E(F_p)$ generated by P is $< P >= \infty, P, 2P, 3P, ..., (n-1)P$. The public domain parameter includes the prime p, the point P, the equation of the Elliptic Curve E and the order n. The private key is an integer d that we have selected within the range of (1,n-1). And the public key is Q where $Q = dP$ where P is the point on the curve. The elliptic curve discrete logarithm problem (ECDLP) use to solve d and Q.
Encryption:
Consider plaintext m that has the point M on the curve E, and is then encrypted by adding it to kQ where $k \in R[1, n-1]$, and Q is the recipient's public key. The sender sends $C_1 = k \star P$ and $C_2 = M + k \star Q$ to the recipient who uses her private key d to compute $dC_1 = d(k \star P) = k(d \star P) = kQ$ and then recover $M = C_2 - k \star Q$. An eavesdropper who want to recover M needs to compute $k \star Q$.
Decryption:

We need to get back to the message $'m'$ that was send to us, $M = C_2 - d \star C_1$ where M is the original message that was sent to us.

5.3 Comparison of Key Sizes

In cryptography, key size or key length is the size measure in bits of the key used in a cryptographic algorithm. The minimum of a key length that is considered strong security is 80 bits. However, the key length of 128 bits is recommended because it offers more security. The private and public keys in RSA and ECC can be chosen from almost equal lengths. From section 3.8, the message expansion of NTRU is $[n/(n-k)] \log_p q - to - 1$. Table in [15] shows the public key sizes of NTRU, RSA and ECC.

Table 5.1. Key Size Graph Comparison of NTRU vs. ECC vs. RSA

Security Level(bits)	n	NTRU Key Sizes(bits)	RSA Key Sizes(bits)	ECC Key Sizes(bits)
80	251	2008	1024	163
112	347	3033	2048	224
128	397	3501	3072	256
160	491	4383	4096	320
192	587	5193	7680	384
256	787	7690	15360	521

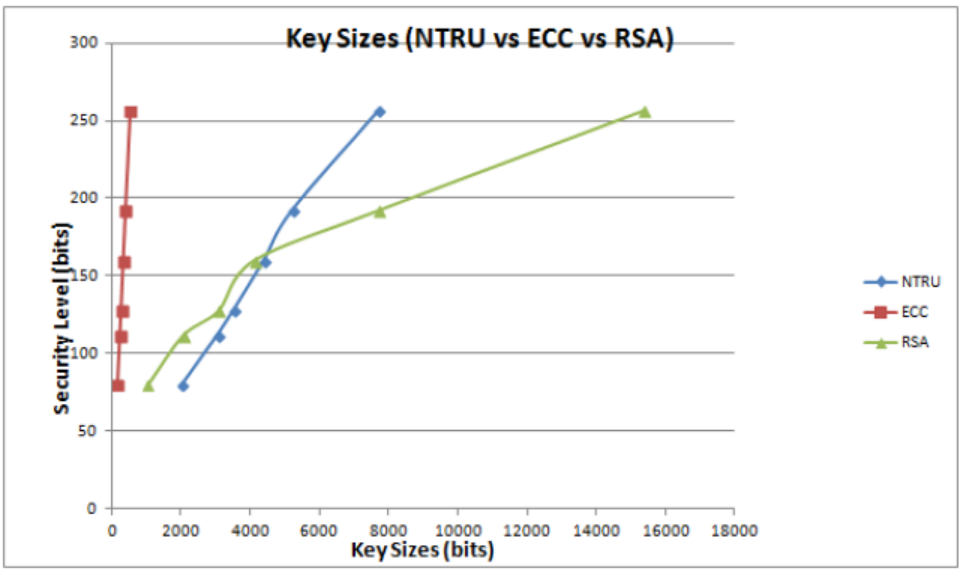

Figure 5.1. Key Sizes Graph Comparison of NTRU vs. ECC vs. RSA

From Figure 5.1 above we can conclude that ECC has the best performance in terms of key size compared to the other two cryptosystems. The NTRU has the worst performance when the security level is from 80 bits to 160 bits. However, its performance will be better than the RSA when the security starts from 192 bits to 256 bits.

5.4 Comparing key generation, encryption, and decryption in NTRU and RSA

Public key cryptosystems are mainly used to exchange either a secret key or a short message. RSA and NTRU work in units of message blocks, and in both system, a single message block is large enough to maintain a short message or a secret key at high security. Therefore, in comparison between these two systems, we will evaluate the length of time to encrypt and decrypt a single message block.

Table in [8] shows the security and timing comparisons of NTRU versus the RSA cryptosystem. Comparisons also include the time for a cryptosystem to encrypt, decrypt and create a key. Key generation is the process of generating keys for cryptography. A key is used to encode and decode any data being encoded/decoded.

Table 5.2. Comparison of NTRU cryptosystem vs. the RSA cryptosystem

System	Security(yrs)	Key Size(bits)	Create Key(msecs)	Encrypt(blks/sec)	Decrypt(blks/sec)
RSA 512	$4.00.10^5$	512	260	2441	122
NTRU 167	$2.08.10^6$	1169	4.0	5941	2818
RSA 1024	$3.00.10^{12}$	1024	1280	932	22
NTRU 263	$4.61.10^{14}$	1841	7.5	3676	1619
RSA 2048	$3.00.10^{21}$	2048	4195	310	3
RSA 4096	$2.00.10^{33}$	4096	–	–	–
NTRU 503	$3.38.10^{35}$	4024	17.3	1471	608

Notes for Table 5.2

-"NTRU n" refers to an implementation of the NTRU public key cryptosystem using the domain parameters n, that is, using polynomials of degree $n-1$ in the ring $Z[x]/(X^n-1)$.

-Security is measured in MIPS-years required to break the system. Note that all security times are estimates, based on extrapolation of the breaking time for lower security levels.

-Key size refers to public key in bits. NTRU private key are shorter than their public keys. RSA public and private keys are the same size.

-NTRU encryption, decryption and key creation performed using Tao Group's Tumbler

implementation of the NTRU algorithm programmed in C and running on a $300\ MHz\ Pentium\ II$ operating under Linux.
-RSA key creation done on a $255\ MHz\ Digital\ Alpha\ Station$.
-RSA encryption/decryption programmed in Microsoft Visual C++ 5.0 (optimized for speed, Pentium Pro code generation), and run on a $Pentium\ II\ 266\ MHz$ machine under Windows NT 4.0. RSA encryption uses exponent 17 to increase speed.

Figures 5.2, 5.3, and 5.4 present the graph of NTRU cryptosystem versus the RSA cryptosystem relating to the key security versus key generation, encryption and decryption. In addition, Figures 5.5, 5.6, and 5.7 present the graph of NTRU cryptosystem versus the RSA cryptosystem relating to the key size versus key generation, encryption and decryption.

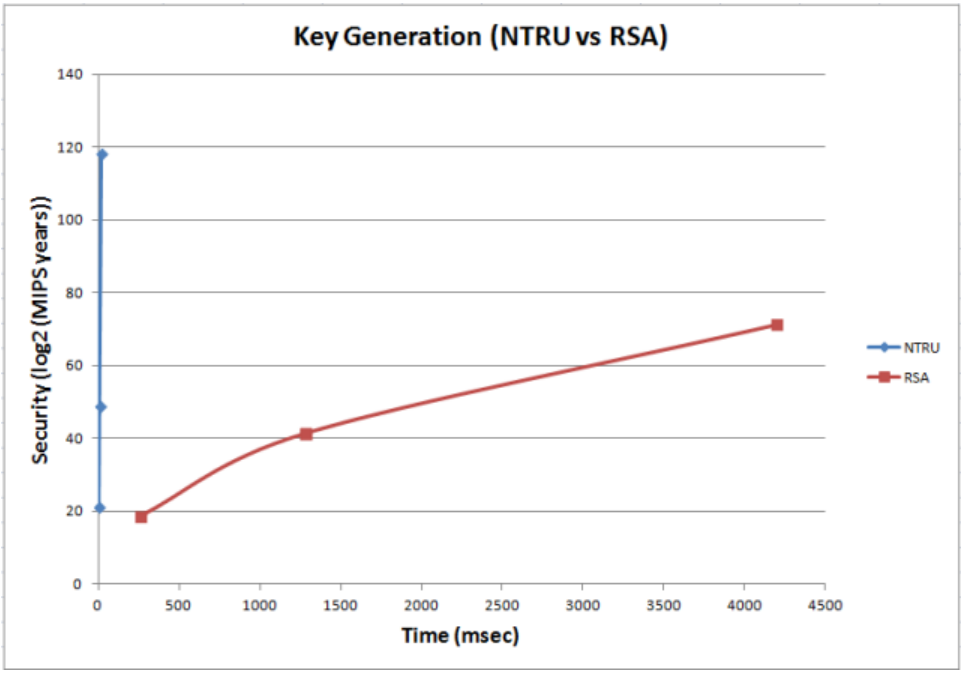

Figure 5.2. Key Generation Graph of NTRU vs. RSA

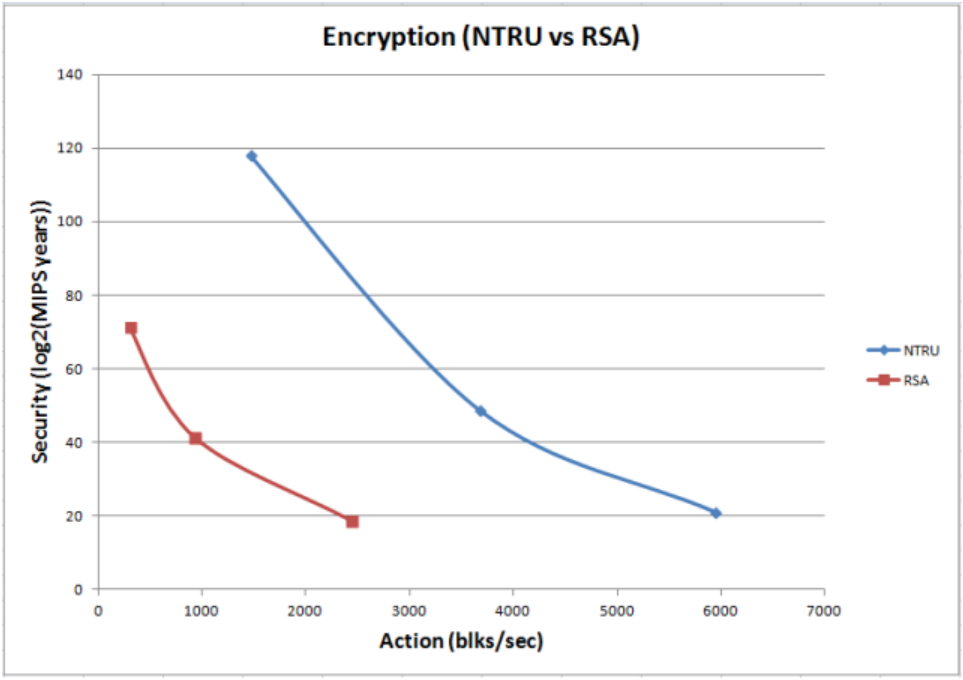

Figure 5.3. Encryption Graph of NTRU vs. RSA

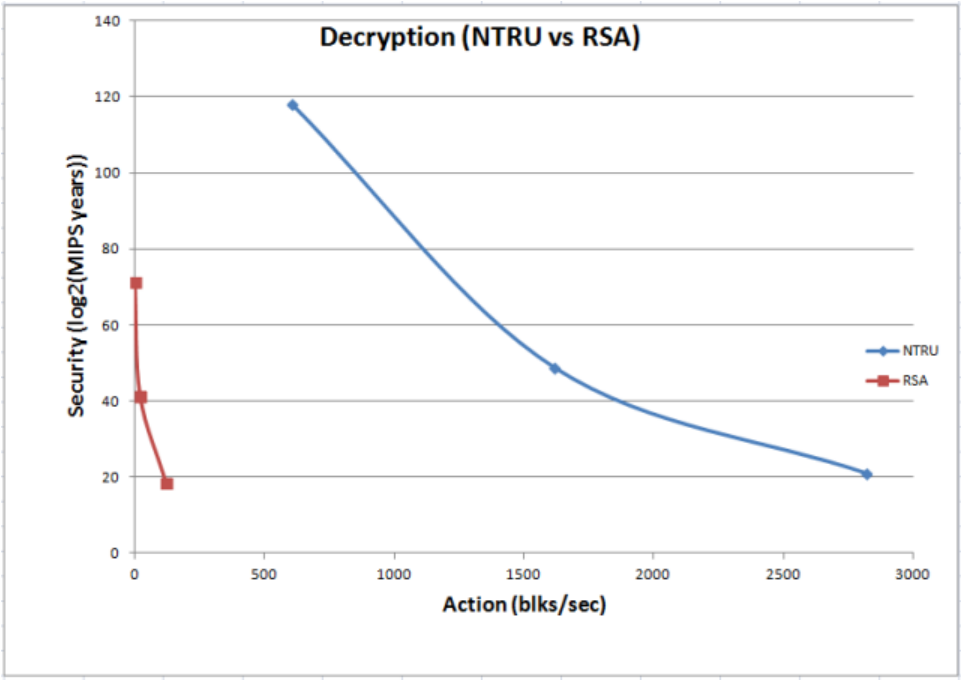

Figure 5.4. Decryption Graph of NTRU vs. RSA

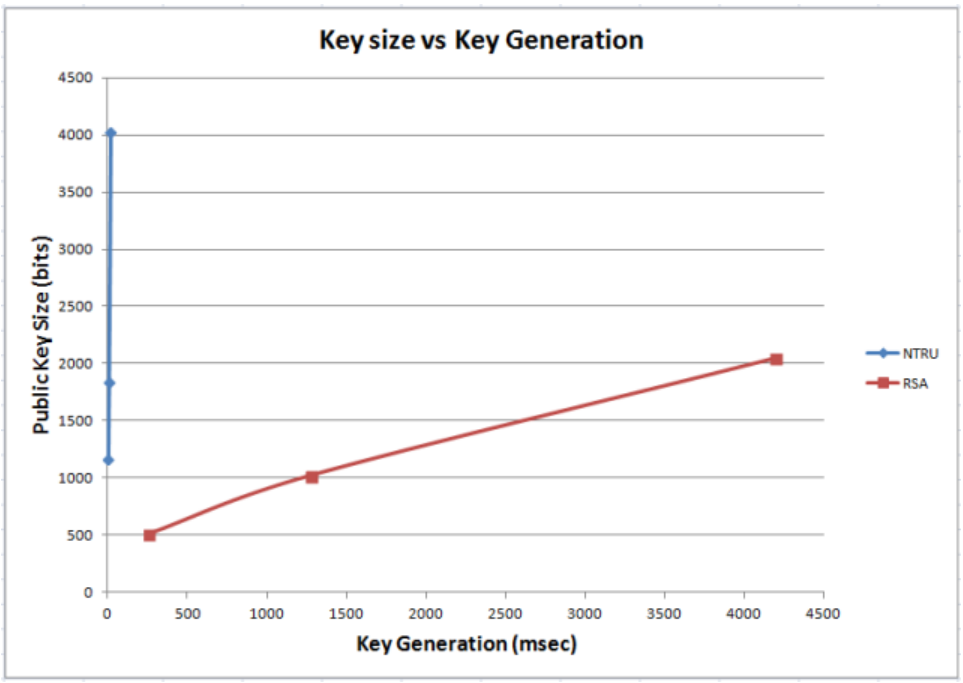

Figure 5.5. Key Size vs. Key Generation Graph of NTRU vs. RSA

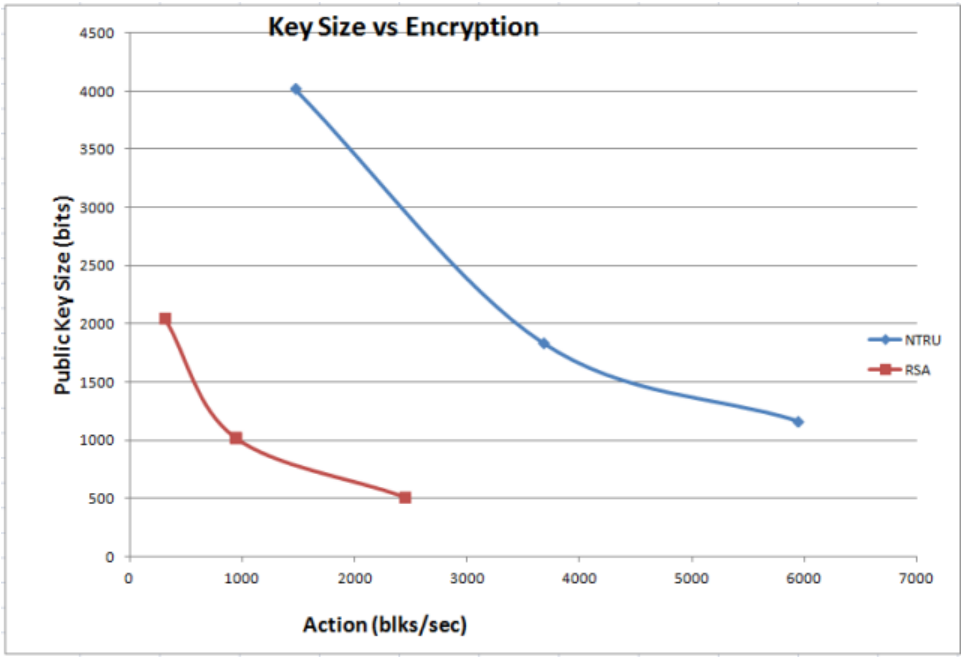

Figure 5.6. Key Size vs. Encryption Graph of NTRU vs. RSA

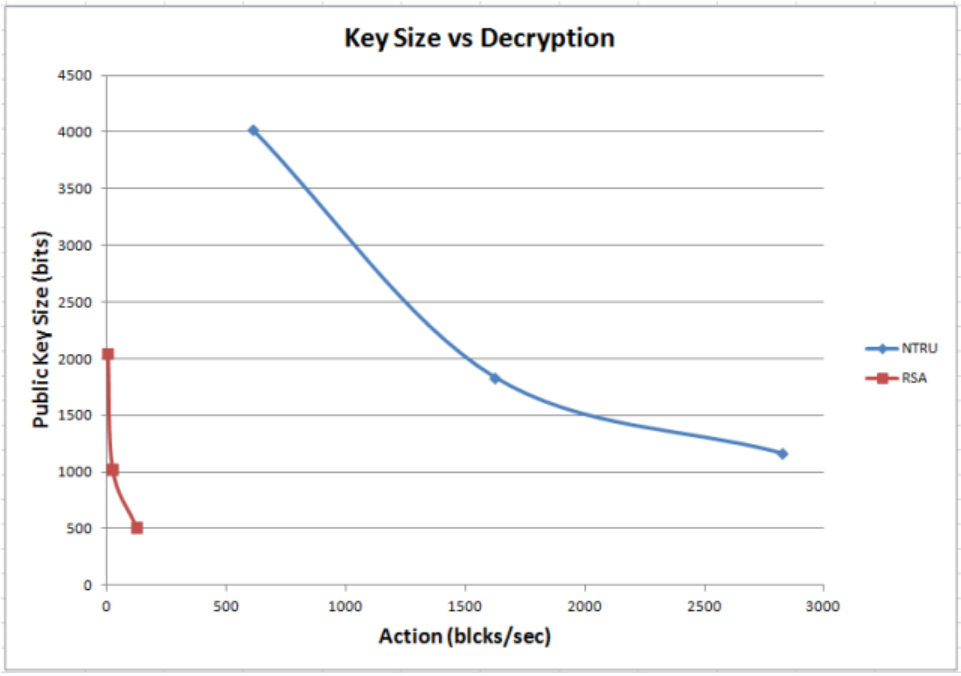

Figure 5.7. Key Size vs. Decryption Graph of NTRU vs. RSA

Clearly, we can conclude that the performance of NTRU is better than RSA in terms of key generation, encryption and decryption by examining Figures 5.2, 5.3, and 5.4. We can also see that the key generation of NTRU is faster than in RSA in Figure 5.5. Furthermore, NTRU encrypts more messages than RSA with the same key sizes in Figure 5.6. Finally, the NTRU also decrypts more messages than RSA with the same key sizes in Figure 5.7.

5.5 Comparing key generation, encryption, and decryption in NTRU and ECC

Table 5.3 [13] shows us the time requirement for key generation, encryption and decryption of NTRU and ECC cryptosystem where the code was compiled and run on a personal computer of Windows XP Professional OS, P4 2.80 GHz CPU and 1GB RAM.

Table 5.3. Comparison of NTRU cryptosystem vs. the ECC cryptosystem

Cryptosystem	Security(bits)	Key Generation(msec)[1]	Encryption(msec)[1]	Decryption(msec)[1]
$NTRU251$	80	75.65	1.68	8.22
$ECC192$	$between\ 80-112$	$57.87-152.73$	$37.81-116.39$	$19.15-57.68$
$NTRU347$	112	114.16	3.11	15.70
$ECC224$	112	$234.11-367.98$	$52.52-164.50$	$26.35-81.52$
$NTRU397$	128	188.92	3.97	20.26
$ECC256$	128	$478.22-656.63$	$68.72-223.29$	$35.00-111.16$
$NTRU491$	160	288.31	5.97	30.96
$NTRU587$	192	412.10	8.42	44.42
$ECC384$	192	$947.43-1429.11$	$182.35-586.20$	$90.61-290.94$
$NTRU787$	256	738.75	14.49	48
$ECC521$	256	$2055.04-3175.87$	$423.25-1257.56$	$211.35-626.33$

[1]ECC timings are given as minimum-maximum of the values observed over all coordinate systems.

Figures 5.8, 5.9, and 5.10 present the graph of the NTRU cryptosystem versus the graph of the minimum values of ECC cryptosystem relating to the key generation, encryption and decryption. Figures 5.11, 5.12, and 5.13 present the graph of the NTRU cryptosystem versus the graph of the minimum values of ECC cryptosystem relating to the key sizes versus the key generation, encryption and decryption. Figures 5.14, 5.15, and 5.16 present the graph of the NTRU cryptosystem versus the graph of the maximum values of ECC relating to the key generation, encryption and decryption. Figures 5.17, 5.18, and 5.19 present the graph of the NTRU cryptosystem versus the graph of the maximum values of ECC cryptosystem relating to the key sizes versus the key generation, encryption and decryption.

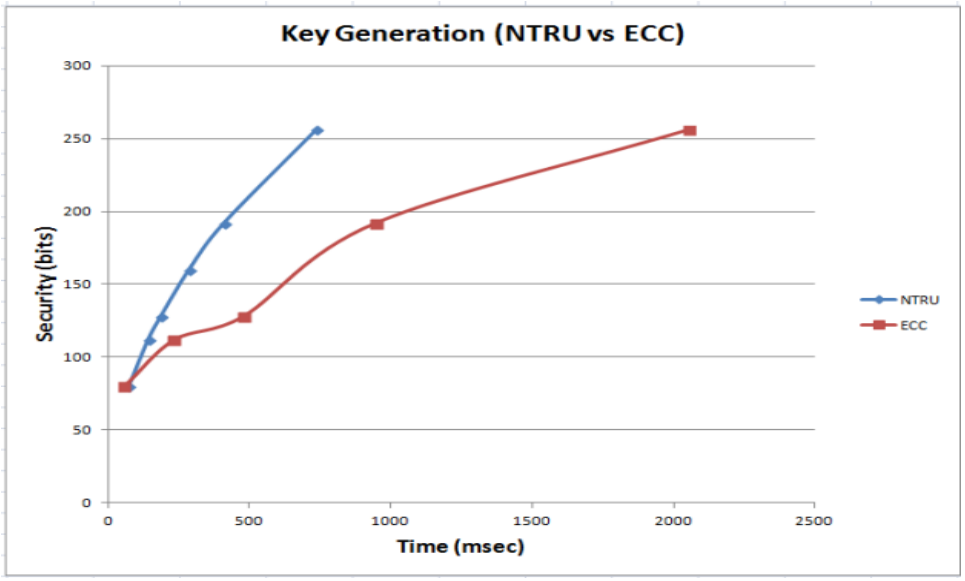

Figure 5.8. Key Generation Graph of NTRU vs. ECC-min

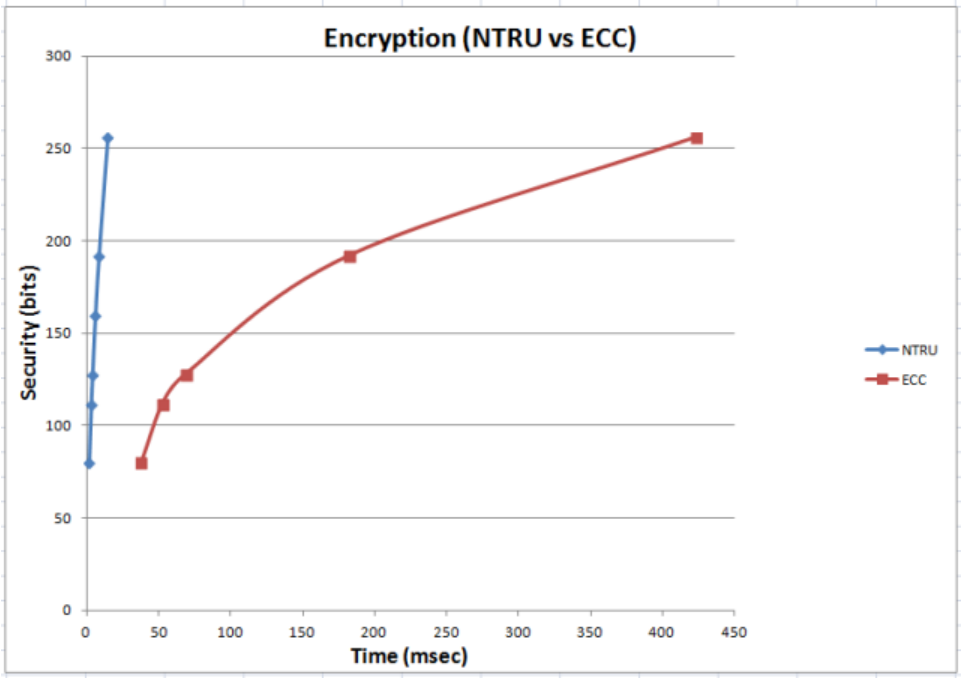

Figure 5.9. Encryption Graph of NTRU vs. ECC-min

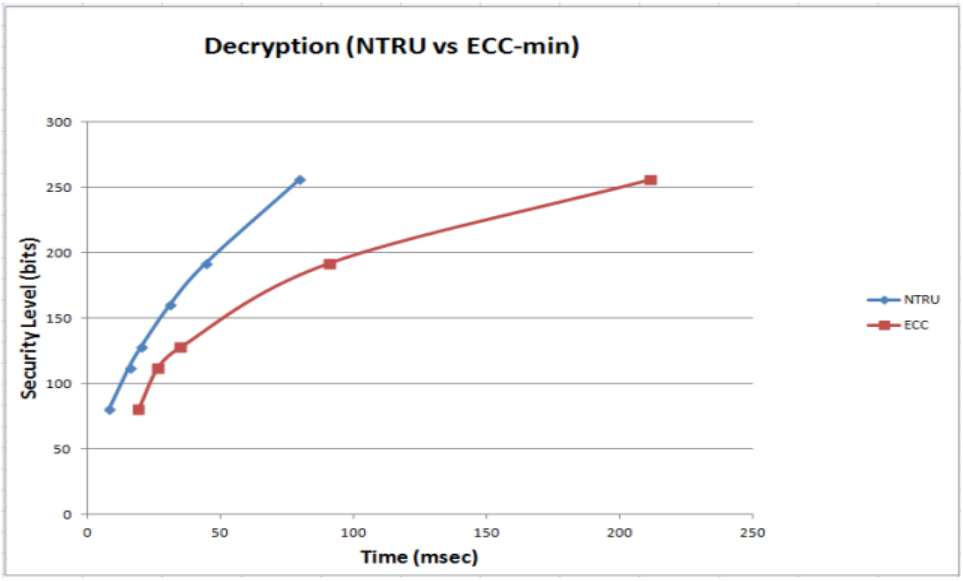

Figure 5.10. Decryption Graph of NTRU vs. ECC-min

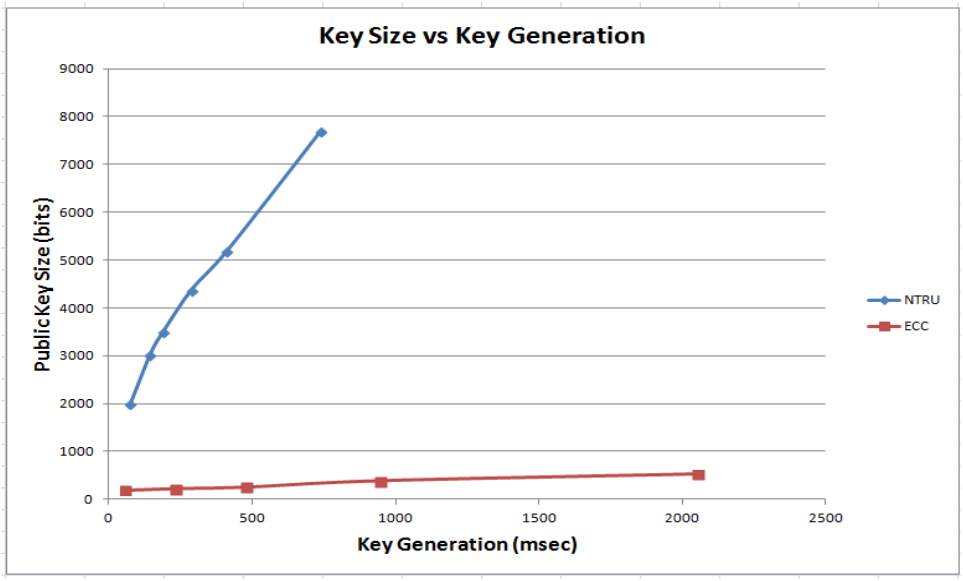

Figure 5.11. Key Size vs. Key Generation Graph of NTRU vs. ECC-min

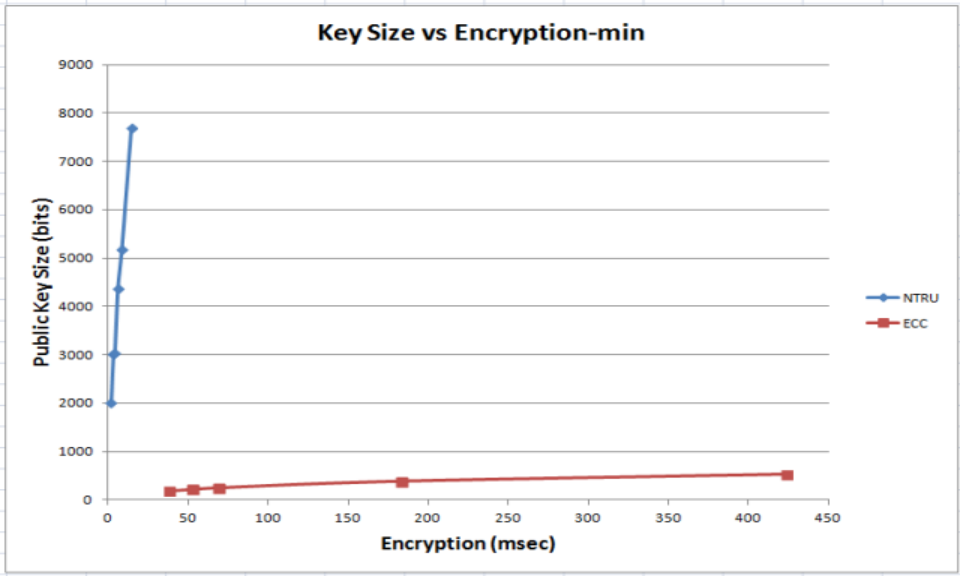

Figure 5.12. Key Size vs. Encryption-min Graph of NTRU vs. ECC-min

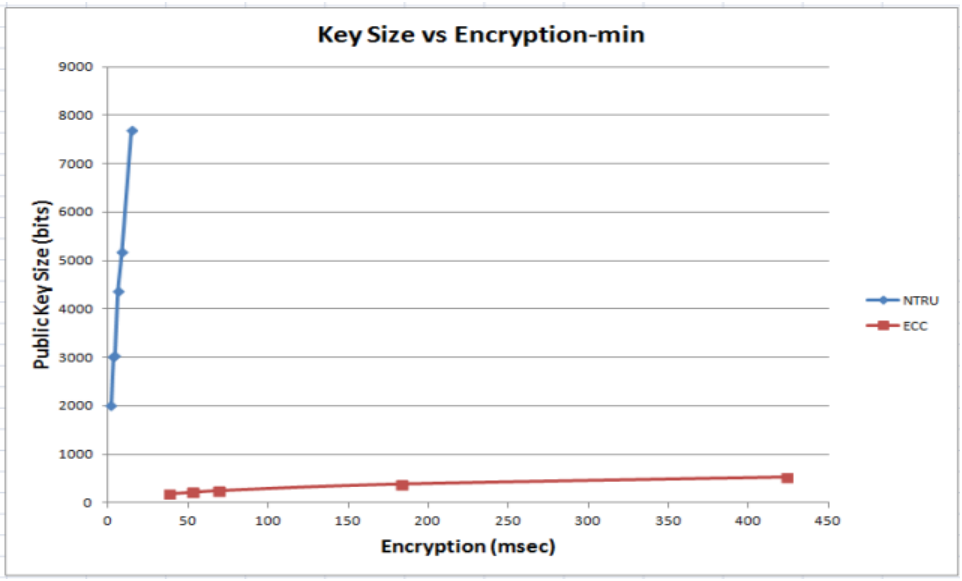

Figure 5.13. Key Size vs. Decryption-min Graph of NTRU vs. ECC-min

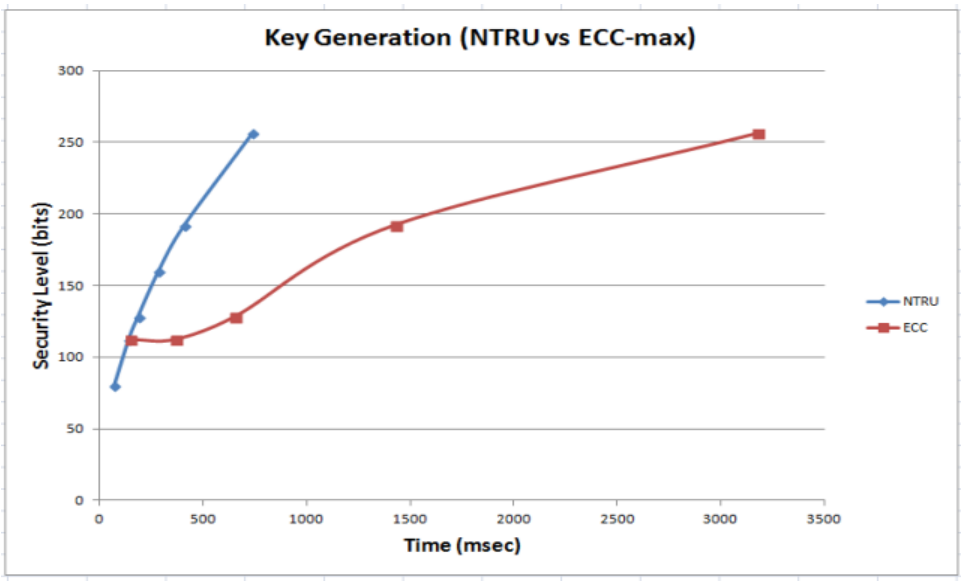

Figure 5.14. Key Generation Graph of NTRU vs. ECC-max

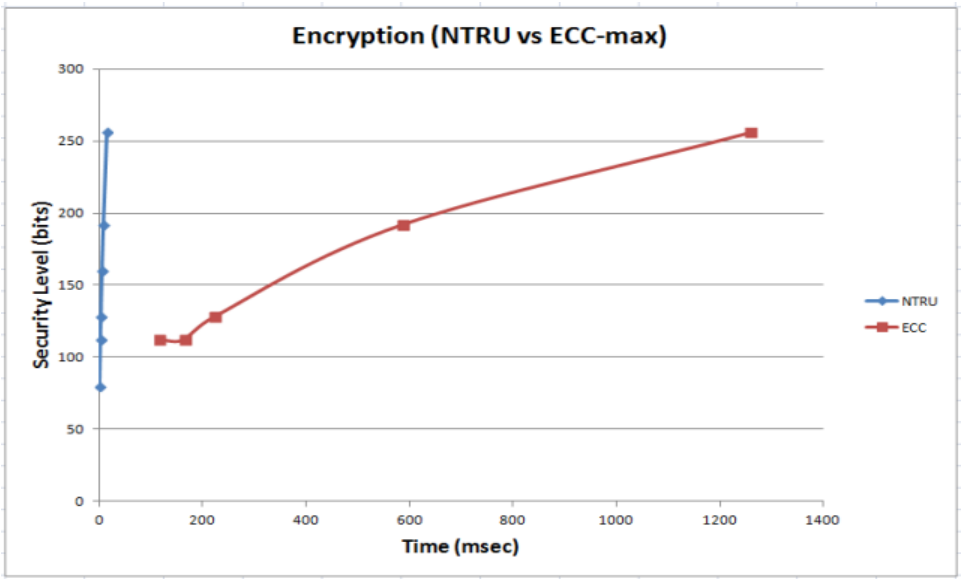

Figure 5.15. Encryption Graph of NTRU vs. ECC-max

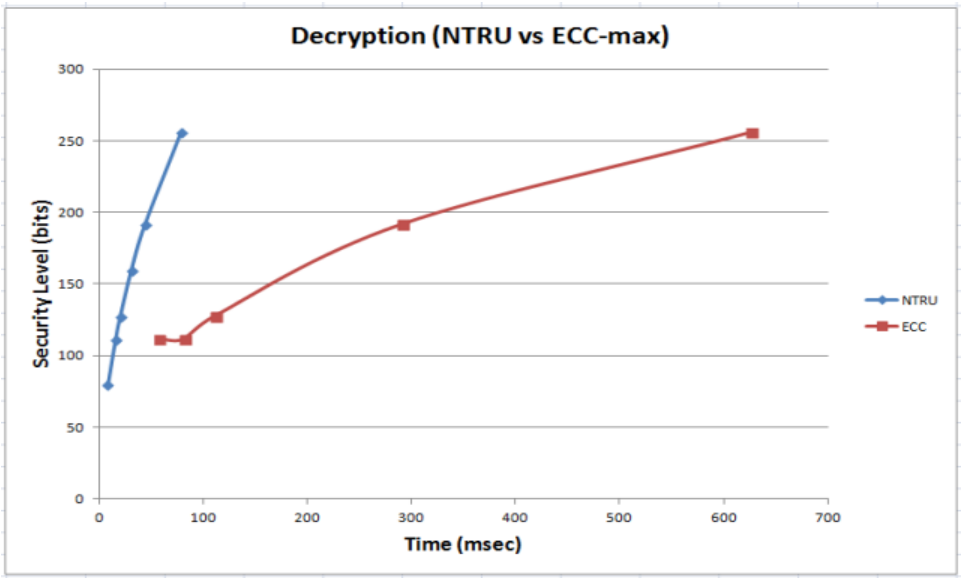

Figure 5.16. Decryption Graph of NTRU vs. ECC-max

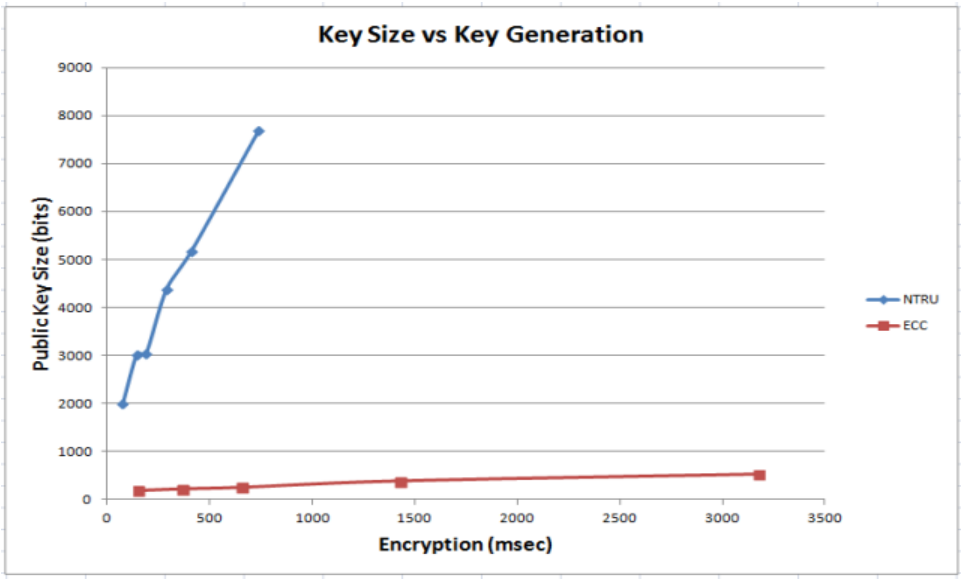

Figure 5.17. Key Size vs. Key Generation Graph of NTRU vs. ECC-max

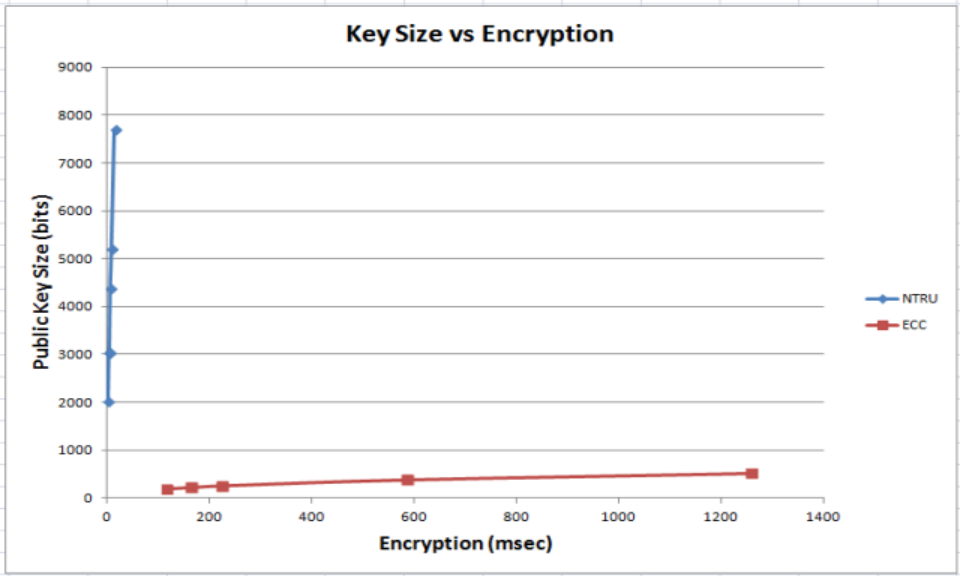

Figure 5.18. Key Size vs. Encryption Graph of NTRU vs. ECC-max

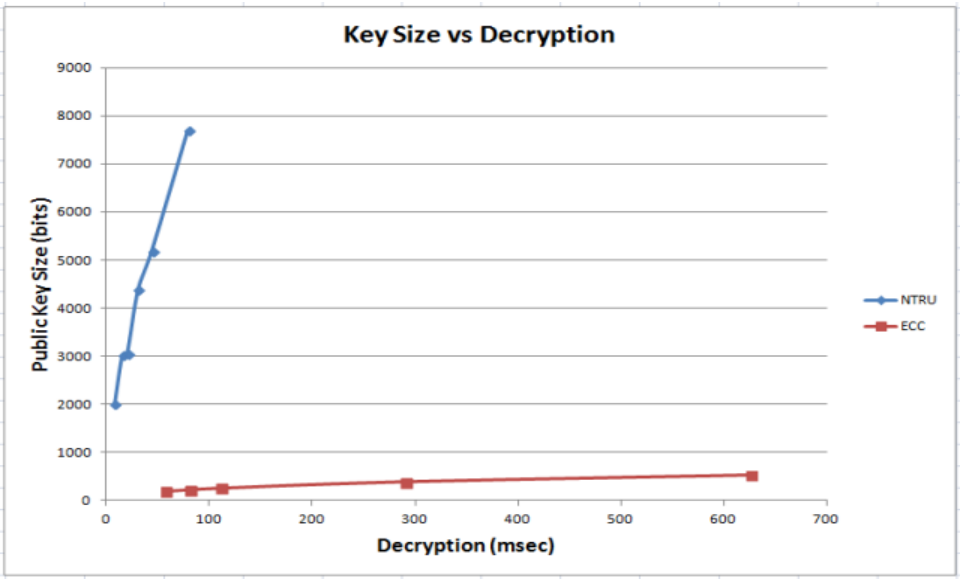

Figure 5.19. Key Size vs. Decryption Graph of NTRU vs. ECC-max

From Table 5.3 above, we discovered that the NTRU is much faster than the ECC with all levels of security. From Figures 5.8, 5.9 and 5.10, we concluded that the performance of NTRU is superior in comparison to the performance of minimum values of ECC timings.

Similarly, from Figures 5.14 to 5.19, we also concluded that the performance of NTRU is superior in comparison to the performance of maximum values of ECC timings.

5.6 McEliece Cryptosystem

McEliece cryptosystem is one of the oldest public key cryptosystem that is based on coding theory. It was developed in 1978 by Robert McEliece. The advantage of the McEliece cryptosystem is that its speed of encryption and decryption is very fast compared with other public key cryptosystems. Furthermore, it is very secure because it is quantum computer resistant. The disadvantage is the large sizes of the private and public key which make it very difficult to use in many practical situations.

The three main steps of the McEliece algorithm are key generation, encryption, and decryption [18].

Key Generation:

- Alice chooses a binary (n, k) linear code C capable of correcting t errors. This code must posses an efficient decoding algorithm and generates a k x n generator matrix G for the code C.
- She randomly picks a k x k invertible matrix S.
- She randomly picks a n x n permutation matrix P.
- She calculates the k x n matrix $\widehat{G} = SGP$
- Her public key:=$(\widehat{G}, t)$ and her private key:=(S, G, P)

Encryption:

- Assume Bob wants to send a message m to Alice whose public key:=$(\widehat{G}, t)$
- He encodes the message m as a binary string of length k.
- He calculates the vector $c' = m\widehat{G}$.
- He generates a random n-bits vector z containing exactly t ones.
- He calculates the ciphertext $c = c' + z$.

Decryption:

- After receipt of c, Alice performs the following steps:
- She calculates P^{-1}
- She calculates $\widehat{c} = cP^{-1}$
- She uses the decoding algorithm for the code C to decode $\widehat{c}$ to $\widehat{m}$
- She calculates $m = \widehat{m}S^{-1}$

Let take a look at the Table 5.4 in [2] where the simulations were performed on a machine featuring an Intel Core 2 processor with dual core. It ran on a 32 bits operating

system and a single core. The C program was compiled with icc Intel compiler with the options-g-static-O-ipo-xP.

Table 5.4. McEliece selected parameters at a glance and NTRU

(m, t)	Encrypt(cycles/byte)	Decrypt(cycles/byte)	Key Size(kB)	Security
$(10, 50)$	243	7938	32	60
$(11, 32)$	178	1848	73	88
$(11, 40)$	223	2577	86	96
$(12, 21)$	126	573	118	88
$(12, 41)$	164	1412	212	130
$(13, 18)$	119	312	227	93
$(13, 29)$	149	535	360	129
$(14, 15)$	132	229	415	91
$(15, 13)$	132	186	775	90
$(16, 12)$	132	166	1532	91
$NTRU$[1]	4753	8445	$-$	$-$

[1]ntru-enc 1 $ee787ep1$ NTRU encryption with $N = 787$ and $q = 587$. Software written by Mark Etzel(NTRU Cryptosystem).

From the Table 5.4 above, we concluded that the speed of the McEliece cryptosystem is faster than the NTRU cryptosystem in terms of encryption time and decryption time.

CHAPTER 6

CONCLUSION AND FUTURE WORK

NTRU is very fast and secure compared to other public key cryptosystems such as RSA and ECC. As a result, in this study we analyzed the NTRU cryptosystem and compared its performance against other cryptosystems. The first part of this thesis introduced a few mathematical backgrounds such as lattices, the shortest and closest vector problems, LLL-Algorithm, Gauss Volume-Heuristic and Convolution Polynomial Ring. Also it described the NTRU cryptosystem parameters, private key, public key, encryption and decryption. Next, some of the general attacks against the NTRU cryptosystem were discussed such as alternate private keys, brute force, meet-in-the-middle, multiple transmission and lattice attacks. First, in alternate private key attack, the alternate key f' is used to decrypt the same message as f. Second, we looked at the brute force attack and an improvement, which we called meet-in-the-middle attack. Third, in the multiple transmission attack, it shows that sending a message multiple times with the same public key h should be avoided. Lastly, we also showed the lattice base attacks on NTRU. Following in this thesis, it introduced other public key cryptosystems such as RSA, ECC and McEliece. The performance from these three cryptosystems was compared to NTRU. In this comparison, it was concluded that ECC has the best key size overall. NTRU was better than RSA if the security level starts from 192 bits to 256 bits. However, in terms of key generation, encryption, decryption and so on, NTRU out performed RSA and ECC. Moreover, when comparing only the speed encryption and decryption of McEliece to NTRU, McEliece has better performance but difficult to use in many practical situations due to the large sizes of the private and public key. In general, this research was solely based on RSA, ECC, McEliece and NTRU. The limitations are that more research is needed to include Ajtai-Dwork cryptosystem and ElGamal cryptosystems. Therefore, suggested future work should compare these two cryptosystems concerning their speed and security with NTRU.

BIBLIOGRAPHY

[1] H. W. L. A. K. LENSTRA AND L. LOVASZ, *Factoring polynomials with rational coefficients*, Mathematische Annalen, 261 (1982), pp. 515–534.

[2] B. BISWASS AND N. SENDRIER., *McEliece cryptosystem implementation: Theory and practice*, in Post-Quantum Cryptography, J. Buchman and J. Ding., eds., Cincinnati, OH, 2008, pp. 47–62.

[3] K. BRAND, *NTRU cryptography a lattice-based cryptosystem and attacks against it*, Master's thesis, University of Zurich, Zurich, Switzerland, 2013.

[4] *Security innovation NTRU FAQs the application security company.* https://www.securityinnovation.com/products/encryption-libraries/ntru-crypto/Security Innovation NTRU FAQs The Application Security Company, accessed October 2014, n.d.

[5] T. M. DAMICO, *A brief history of cryptography.* http://www.studentpulse.com/articles/41/a-brief-history-of-cryptography, accessed April 2014, n.d.

[6] A. M. DARREL HANKERSON AND S. VANSTONE, *Guide to Elliptic Curve Cryptography*, Springer, New York, 2004.

[7] J. HOFFSTEIN AND J. H. SILVERMAN, *Implementation notes for NTRU PKCS multiple transmissions*, Technical Report 6, NTRU Cryptosystem Inc., Burlington, 1998.

[8] T. HOFFSTEIN, D. LIEMAN, J. PIPHER, AND J. SILVERMAN, *NTRU: A Public Key Cryptosystem*. grouper.ieee.org/groups/1363/latt/PK/submissions/ntru.pdf, accessed January 2014, n.d.

[9] K. JARVIS, *NTRU over the eisenstein integers*, Master's thesis, University of Ottawa, Ottawa, Canada, 2011.

[10] J. JEFFREY HOFFSTEIN, JILL PIPHER AND J. H. SILVERMAN., *NTRU: A Ring-Based Public Key Cryptosystem*. http://math.boisestate.edu/ liljanab/MATH508/ntru.pdf, accessed March 2014, n.d.

[11] J. P. A. J. H. S. JEFFREY HOFFSTEIN, *An Introduction to Mathematical Cryptography*, Springer, New York, 2008.

[12] C. LUDWIG, *A faster lattice reduction method using quantum search*, in Algorithms and Computation, H. O. Toshihide Ibaraki, Naoki Katoh, ed., vol. 14, Kyoto, Japan, 2003, pp. 199–208.

[13] A. MERSIN AND M. BEYAZIT, *On NTRU and its performance*, in Security of Information and Networks: Proceedings of the First International Conference on Security of Information and Networks (SIN 2007), S. B. O. Atilla Elci and B. Preneel., eds., Bloomington, IN, 2008, pp. 278–284.

[14] W. W. NICK HOWGRAVE-GRAHAM, JOSEPH H. SILVERMAN, *A meet-in-the-middle attack on a NTRU private key*, Technical Report 4, NTRU Cryptosystem Inc., Burlington, 2003.

[15] W. W. NICK HOWGRAVE-GRAHAM, JOSEPH H. SILVERMAN, *Choosing parameter sets for NTRUEncrypt with NAEP and SVES-3*. http://eprint.iacr.org/2005/045.pdf, accessed May 2014, 2005.

[16] T. E. OF ENCYCLOPAEDIA BRITANNICA, *History of cryptology.* http://www.britannica.com/EBchecked/topic/530355/scytale, accessed March 2014, n.d.

[17] I. P1363.1, *Draft standard for public key cryptographic techniques based on hard problems over lattices.* http://grouper.ieee.org/groups/1363/lattPK/draft.html, accessed April 2014, n.d.

[18] O. PAUSTJAN, *Post quantum cryptography on embedded devices: An efficient implementation of the McEliece public key scheme based on Quasi-Dyadic Goppa Codes*, Master's thesis, Ruhr-University Buchum, Buchum, Germany, 2010.

[19] M. PAWLAN, *Cryptography: the ancient art of secret messages.* http://www.pawlan.com/Monica/crypto/Cryptography: the ancient art of secret messages, accessed May 2014, 1998.

[20] S. K. RANJEET RANJAN, DR. A. S. BAGHEL, *Improvement of NTRU cryptosystem*, International Journal of Advanced Research in Computer Science and Software Engineering, 2 (2012), pp. 79–84.