THE COMPARATIVE PERFORMANCE ANALYSIS OF LATTICE BASED NTRU CRYPTOSYSTEM WITH OTHER ASYMMETRICAL CRYPTOSYSTEMS

A Thesis Submitted to
the Graduate School of Engineering and Science of
İzmir Institute of Technology
in Partial Fullfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Software

by
Ali MERSİN

September 2007
İZMİR

We approve the thesis of **Ali MERSİN**

Date of Signature

..... 10 September 2007
Assoc. Prof. Dr. Ahmet KOLTUKSUZ
Supervisor
Department of Computer Engineering
İzmir Institute of Technology

..... 10 September 2007
Prof. Dr. Şaban EREN
Department of Computer Engineering
Maltepe University

..... 10 September 2007
Dr. Serap ATAY
Department of Computer Engineering
İzmir Institute of Technology

..... 10 September 2007
Prof. Dr. Sıtkı AYTAÇ
Head of Department
Department of Computer Engineering
İzmir Institute of Technology

.....
Prof. Dr. M. Barış ÖZERDEM
Head of the Graduate School

ACKNOWLEDGEMENT

Foremost, I would like to thank my advisor, Assoc. Prof. Dr. Ahmet Koltuksuz, for his guidance, patience, and encouragement.

Furthermore, I would like to thank Dr. Serap Atay for lending her source code to measure ECC timings; Hüseyin Hışıl for our CRYMPIX education and his contributions for my cryptographic knowledge; my room mate Selma Tekir for her support and patience.

I also would like to thank my room mate Mutlu Beyazıt for his huge support; if he had not helped, this thesis would not be completed.

I would like to thank my parents for their support throughout my education as well as in my graduate study.

Finally, I would like to thank my dear girl friend, Gönül Karahan, who always stands by me. She is the one who ensured that I completed my graduate study.

ABSTRACT

THE COMPARATIVE PERFORMANCE ANALYSIS OF LATTICE BASED NTRU CRYPTOSYSTEM WITH OTHER ASYMMETRICAL CRYPTOSYSTEMS

Current popular asymmetrical cryptosystems are based on hardness of number theoretic problems. In the future, these problems may become practically solvable with the improvements of processing power, the development of quantum computation and distributed computation. So the need for new cryptosystems which are not based on these problems has risen. Researches on hardness of lattice problems have brought a new candidate for asymmetrical cryptography: Lattice Based Cryptography. NTRU is one of these cryptosystems. For practical use, the actual performance results of NTRU with respect to current asymmetrical cryptosystems should be known. This thesis is developed based on this purpose.

ÖZET

KAFES TABANLI NTRU KRİPTOSİSTEMİNİN DİĞER ASİMETRİK KRİPTOSİSTEMLERLE KARŞILAŞTIRMALI PERFORMANS ANALİZİ

Günümüzde aktif olarak kullanılan asimetrik kriptosistemlerde güvenlik sayı teorisi problemlerinin zorluklarına dayanmaktadırlar. İşlemci gücünün gün geçtikçe artması, kuantum ve dağıtık hesaplamanın gelişmesiyle, sayı teorisi problemlerinin zorluğunu temel alan kriptosistemlerin güvenilirlikleri gelecekte tehlike altına girebilecektir. Bu nedenle asimetrik kriptografide kullanılabilecek, başka matematiksel problemleri kullanan, yeni kriptosistemler üretilmeli ve bir alternatif olarak sunulmalıdır. Bu amaçla, kafes problemlerinin zorlukları üzerine yapılan çalışmalar, "Kafes Tabanlı Kriptografi"yi ortaya çıkarmış ve kafes tabanlı NTRU kriptosistemi geliştirilmiştir. NTRU kriptosisteminin günümüz kriptosistemlerinin yerini alabilmesi için; NTRU kriptosisteminin bu kriptosistemlerle karşılaştırmalı performans analizi yapılmalıdır ve bu gereksinim tezimin hedefini oluşturur.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

The most popular public key cryptosystems like RSA and ECC are based on the complexity of number theoretic problems and their security is highly dependable to the distribution of prime numbers or based on the discrete logarithm problem on finite fields. With the development of distributed computation, grid computing and quantum computers the breaking times, even the brute force attacks, for these cryptosystems are diminished. This can be dangerous for future use of the public key cryptography. Putting all the cryptographic eggs to the same basket is a risky situation, so there should be an alternative for the future. So cryptosystems which are not based on the same problems, for example cryptosystems based on geometrical problems etc., will be, at least, our reserve for future failures of current cryptosystems.

Researches on complexity of lattice problems have raised a new candidate for public key cryptography. Based on hardness of lattice problems, several cryptosystems have been developed such as Ajtai-Dwork (Ajtai and Dwork 1997), Goldreich-Goldwasser-Halevi (Goldreich et al. 1997) and NTRU (WEB_1 2006) cryptosystems. With key complexity of $O(n)$ instead of $\Omega\left(n^2\right)$, NTRU has the best performance among the other lattice based cryptosystems (Hoffstein et al. 1998). Today, extensive researches have been going on concerning the NTRU and no crucial security issue has been found so far.

NTRU uses a special lattice called NTRU Lattice. Actually an NTRU lattice is a special version of a convolution modular lattice. If a convolution modular lattice contains a short vector, then it is called as NTRU Lattice.

There are several hard lattice problems which are;

- “shortest vector problem”,
- “closest vector problem”,
- “shortest basis problem and their variations”.

The security of NTRU cryptosystem is conjectured to be equivalent to the hardness of the shortest vector problem and the closest vector problem. The shortest

vector problem (SVP) is the problem of finding the vector, other than the zero vector, that has the smallest L^2 norm; while the closest vector problem (CVP) is the problem of finding a lattice vector which has the smallest L^2 norm of distance with a given vector. It is known that the shortest vector problem is NP-Hard under randomized reduction hypothesis (Ajtai 1998). It is also known that the closest vector problem is NP-hard and the solution for this problem is at least as hard as the solution of the shortest vector problem (Nguyen and Stern 2001).

To be a candidate for public key cryptography, the comparative performance results for NTRU should be known. So this thesis is intended to clarify the performance values of such cryptosystems with a comparative way.

We begin with the mathematical background for lattice cryptography in Chapter 2. In Chapter 3 we give the basics of the asymmetrical cryptosystems that we compared with NTRU and the history of lattice based cryptography. We explain the NTRU cryptosystem in Chapter 4 while we give the details of implementation in Chapter 5. The results for comparison of cryptosystems are given in Chapter 6 and the thesis is concluded in Chapter 7.

CHAPTER 2

MATHEMATICAL BACKGROUND

This chapter provides the basics of mathematics which will be needed for the rest of the thesis. Section 2.1 describes the fundamentals of *Lattice*s while Section 2.2 covers *Quotient Polynomial Ring*. Section 2.3 describes the *Hard Lattice Problems* on which the cryptosystems (see the related sections in Chapter 3) are based. This chapter is a summary of mathematical background will be needed throughout this thesis; for further and detailed information the reader is referred to (Micciancio and Goldwasser 2002, WEB_7 2007).

2.1. Lattice

Throughout the rest of this thesis there will be need for lattice and some special lattices -like modular lattice and NTRU lattice-. This section aims to define the lattice concept and special lattices.

2.1.1. Lattice

A lattice is the set of all integral combination of *n* linearly independent vectors, $\mathbf{b}_1,\ldots,\mathbf{b}_n$, in *m*-dimensional Euclidean space, $\mathbb{R}^m$. Let $\mathbb{Z}$ be the set of integers, a lattice is denoted by

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

The vectors, $B=\{\mathbf{b}_1,\ldots,\mathbf{b}_n\}$, represent the *basis (or base)* of the lattice. The integers, *n* and *m* are called as the *rank* and *dimension* of the lattice, respectively. When the rank and the dimension are equal, the lattice is called as *full rank*. Throughout the rest of this

thesis, all the lattices are full rank and all the coefficients of the base vector are integers; unless stated otherwise.

The L^2 norm and *centered* L^2 norm of a vector $\mathbf{v} = [v_0 \quad v_1 \quad \dots \quad v_{n-1}]$ is respectively defined as follows:

$$|\mathbf{v}| = \sqrt[2]{\sum_{i=0}^{n-1} v_i^{\,2}}$$

$$\|\mathbf{v}\| = \sqrt[2]{\sum_{i=0}^{n-1} (v_i - \overline{v})^2} \ , \ \overline{v} = \frac{1}{n}\sum_{i=0}^{n-1} v_i \ .$$

2.1.2. Modular Lattice

A *modular lattice* is the lattice that is spanned by the rows of the following matrix

$$\mathbf{L} = \begin{bmatrix} b\mathbf{I} & \mathbf{H} \\ \mathbf{0} & q\mathbf{I} \end{bmatrix}$$

and denoted by L_{ML} = *rowspan*(**L**). Here $b, q \in \mathbb{Z}$ and the elements of matrix 2n-dimensional **L,** are *n x n* matrices. **I** and **0** represents identity and 0 (zero) matrix, respectively. All elements of matrix **H** are reduced modular *q*.

Let **H** be defined as follow

$$\mathbf{H} = \begin{bmatrix} h_0 & h_1 & \dots & h_{n-1} \\ h_{n-1} & h_0 & \dots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \dots & h_0 \end{bmatrix},$$

in this case L_{ML} is called as convolution or circulant modular lattice and denoted by L_{CML}. **H** is the circulant matrix of vector h, $\mathbf{h} = [h_0 \quad h_1 \quad \dots \quad h_{n-1}]$ and denoted by $\mathbf{M_h}$**.** It is obvious that $\mathbf{M_h}$ is obtained by shifting **h** cyclically. Furthermore, if the vector $[a_0 \quad a_1 \quad \dots \quad a_{n-1} \quad b_0 \quad b_1 \quad \dots \quad b_{n-1}]$ is an element of convolution modular lattice than any vector of form $[a_k \quad a_{k+1} \quad \dots \quad a_{k-1} \quad b_k \quad b_{k+1} \quad \dots \quad b_{k-1}]$, $k = 1, \dots, n-1$ is also an element of that convolution modular lattice.

2.1.3. NTRU Lattice

Consider the convolution modular lattice

$$L_{CML} = rowspan\left(\mathbf{L} = \begin{bmatrix} b\mathbf{I} & \mathbf{M_h} \\ \mathbf{0} & q\mathbf{I} \end{bmatrix}\right),$$

If this lattice also contains a short vector like

$$[\mathbf{f} \quad \mathbf{g}] = [f_0 \quad f_1 \quad \cdots \quad f_{n-1} \quad g_0 \quad g_1 \quad \cdots \quad g_{n-1}]$$

then, this lattice is called as *NTRU Lattice* and denoted as L_{NTRU}. Here b is called the *balancing constant*.

2.2. Quotient Polynomial Ring

Throughout this thesis, the ring $R = \mathbb{Z}[X]/(x^n - 1)$ will be used. An element of this ring f can be written as a polynomial or a vector as follow

$$f = \sum_{i=0}^{n-1} f_i x^i = [f_0, f_1, ..., f_{n-1}].$$

2.2.1. Addition on Quotient Polynomial Ring

Addition operation on quotient polynomial is traditional polynomial addition. When one tries to add two ring elements, he/she has to add just the coefficients respectively.

2.2.2. Multiplication on Quotient Polynomial Ring

Multiplication operation on quotient polynomial ring is called as *convolution product* and also called as *star multiplication*. Throughout this thesis star multiplication will be denoted as '*'. The star multiplication is defined as

$$f * g = h \text{ with } h_l = \sum_{i=0}^{l} f_i g_{k-i} + \sum_{i=l-1}^{n-1} f_i g_{n+l-i} = \sum_{i+j \equiv l \,(\mathrm{mod}\, n)} f_i g_j \,.$$

In ring R when a multiplication modulo q is done, it means that the coefficients of resultant polynomial are reduced modulo q. The result lies in $\mathbb{Z}[X]/(q, x^n - 1)$.

2.2.3. Modular Lattices and Quotient Polynomial Ring

Let $M(X) \in \mathbb{Z}_q[X]$ be a monic polynomial of degree N. Then each element of the quotient ring $\mathbb{Z}_q[X]/(M(X))$, say $h(X)$, can generate a modular lattice L_h by the following equation:

$$L_h = \{[F, G] : F(X) * h(X) \quad in \quad Z_q[X]/(M(X))\}.$$

In other words, the lattice L_h is formed from all polynomials $F(X), G(X) \in \mathbb{Z}[X]$ satisfying $F(X) * h(X) = G(X) \quad (modulo \quad q \quad and \quad M(X))$.

If $h(X)$ is chosen properly, the generated lattice will contain a pre-selected vector $[f, g]$. In order to do this $h(X)$ must be of form:

$h(X) = f(X)^{-1} * g(X) \quad (modulo \quad q \quad and \quad M(X))$. (It is assumed that $f(X)$ is invertible in the ring $Z_q[X]/(M(X))$).

2.3. Hard Lattice Problems

In order to propose a cryptosystem one needs a ring and hard problem(s) on that ring. Previous section has defined the ring and the next chapter will introduce several hard problems on lattices.

2.3.1. The Shortest Vector Problem

The shortest vector problem (SVP) is the problem of finding the vector –other than the zero vector- that has the smallest L^2 norm. A variation of the shortest vector problem –a simpler one- is called as *approximate short vector problem* (apprSVP) which is to search a lattice vector $v \neq 0$ that satisfies $|v| \leq \gamma |u|$ for all lattice vectors $u \neq 0$. It is obvious that when $\gamma = 1$ apprSVP is SVP. It is known that the shortest vector problem is NP-hard under randomized reduction hypothesis. It is also showed that approximation of SVP within some constant like $\gamma < \sqrt{2}$ is as hard as SVP in (Micciancio 1998).

2.3.2. The Closest Vector Problem

The closest vector problem (CVP) is the problem of finding a lattice vector which has the smallest L^2 norm of distance with a given vector. In other words, let t be a target vector –not necessarily a lattice vector-. One has to find a lattice vector v which satisfies $|v - t| \leq |u - t|$ for all lattice vectors u. A variation of the closest vector problem –a simpler one- is called as *approximate close vector problem* (apprCVP). In this problem it is enough that $|v - t| \leq \gamma |u - t|$ is satisfied. It is known that the closest vector problem is NP-hard. Also it is shown that the solution of the CVP is as hard as SVP in (Nguyen and Stern 2001).

2.3.3. The Smallest Basis Problem

The definition of *the smallest basis problem* can vary as the small basis defined. In general the smallest basis problem is trying to choose the set of base vectors which has the longest vector or the vector with the biggest L^2 norm with the minimum length among other basis or minimizing the product of the lengths of the basis vectors.

2.3.4. Notes on Lattice Problems

One can easily compute a close vector for a given lattice point but without extra information, a short vector, the closest vector for that computed vector can not be found. This property builds a trapdoor for lattice based cryptography.

In order to balance the solutions of the shortest and closest vector problems, the value of balancing constant b can be changed properly. Even though the resultant lattice and the solutions are changed, they can be modified properly (WEB_4 2006, WEB_7 2007)[1].

All the problems mentioned before can be defined on other norms. Besides the solutions the problems are getting harder as the dimension of the lattice increases (Micciancio and Goldwasser 2002, Cai 2000)[2].

[1] WEB_7 2007 pp. 48-49.
[2] Micciancio and Goldwasser 2002 pp. 17-22.

CHAPTER 3

ASYMMETRICAL CRYPTOGRAPHY AND THE HISTORY OF LATTICE BASED CRYPTOGRAPHY

This chapter provides basic information about asymmetrical cryptosystems besides lattice based cryptosystems and a look at lattice based cryptography from a historical point of view. Section 3.1 describes the basics of asymmetrical cryptography and several asymmetrical cryptosystems. Section 3.2 will give a look at the history of lattice.

3.1. Asymmetrical Cryptosystems

This section will give basic theory and information about asymmetrical cryptography and cryptosystems. The subsection 3.1.1 will give the basic definitions and ideas of asymmetrical cryptography, while the subsection 3.1.2 will focus on asymmetrical cryptosystems like RSA and ECC.

3.1.1. Introduction to Asymmetrical Cryptography

The word *cryptography* is originated from the Greek words *kruptein* and *graphein* which mean *to hide* and *to write* respectively. Very basically one can think that the cryptography is to transform things into other things which are meaningless without some information. Formally a cryptosystem is a family **T** of transformations on **P**laintext and **C**iphertext. A "key" is the family index member which determines plaintext transforms into which ciphertext.

The process of transforming plaintext into ciphertext is called *encryption* while the process of transforming ciphertext into plaintext is called *decryption*. In symmetrical

cryptography; the family index is called *key which is* used in both encryption and decryption process. In asymmetrical cryptography the family member index that is used in encryption is called *public key* and the other one that is used in decryption is called *private key*.

The idea of public key (asymmetrical) cryptosystems is to differentiate the encryption key and the decryption key. It solves the key distribution problem. When one wants to use encryption he/she only announce his/her public key. Anyone, desiring to message with him/her, just encrypts the messages by the announced public key. The most critical property of public key cryptography is it should not be possible to retrieve the private key from the public key.

One way and trapdoor functions are used in order to construct asymmetrical cryptosystems. A one way function is a function that; it is *easy* to compute but it is *hard* to invert. Here "easy" means that the function is computable in probabilistic polynomial time, hard means that the function is easy to invert only for a negligible fraction of the inputs. A trapdoor function is actually a one way function with an additional property. The trapdoor functions are easy to invert if additional information (trapdoor) is provided.

3.1.2. Some Asymmetrical Cryptosystems

RSA cryptosystem is the one of the earliest public key cryptosystems (Rivest et al. 1978). The hard problem that the cryptosystem gets its strength is the integer factorization problem. The following steps are the key generation steps for RSA.

- Select two primes p and q;
- calculate $n = pq$;
- compute phi of n which is $\Phi(n) = (p-1)(q-1)$;
- choose an integer e where $\{ \gcd(e, \Phi(n)) = 1;\ 1 < e < \Phi(n) \}$;
- compute $d = e^{-1} \left(\mathrm{mod}\, \Phi(n)\right)$.

After this procedure the public key is the pair e, n and the private key is d.

One has to compute $c = m^e \pmod{n}$ in order to obtain the ciphertext *c* from the plaintext *m* where $m < n$. Similarly $m = c^d \pmod{n}$ is the decryption operation.

An elliptic curve *E* is defined by $y^2 = x^3 + ax^2 + bx + c$ over a Galois field F_p. p is prime and p > 3. Any pair (*x*, *y*) – *x* and *y* are modulo *p* - which is a solution for this equation, is said to be a point over the elliptic curve. There is a rule, called the "chord-and-tangent" rule, for adding two points on an elliptic curve $E(F_p)$ to give a third elliptic curve point. From this rule the following operations can be extracted (Hankerson et al. 2004)[3]:

Let $P(x_1, y_1)$, $Q(x_2, y_2)$ and $P, Q \in E(F_p)$ where $P \neq \pm Q$

- Point addition $P + Q = R(x_3, y_3)$
- Point doubling $2P = R(x_3, y_3)$

The order of elliptic curve is defined by $\#E(F_p) = n$. The *n* is the count of the points over the elliptic curve when the point counting is started any base point *P* over the curve until this point counting operation riches to the point at infinity. *n* should be a prime. Elliptic curve cryptosystem is based on the Elliptic Curve Discrete Logarithm Problem [ECDLP]. Let $P, Q \in E$, ECDLP is the problem of finding integer *k* such that $Q = kP$ where $0 < k < n - 1$.

Let *GF* be a finite Galois field, *E* be an elliptic curve defined over *GF* and *P* be a base point on *E*. Key generation is defined as: generate a random integer k where $k \in \mathbb{Z}_p$ and *p* is prime, calculate $Q = kP$. Now the public key is Q and the private key is k.

The following is Elliptic Curve Encryption System and works as follows:

- The message is divided into pairs (m_1, m_2) such that $m_1 \in GF$ and $m_2 \in GF$.
- A random integer *a* is selected and points $(x_1, y_1) = aP$ and $(x_2, y_2) = aQ$.
- (m_1, m_2) and (x_2, y_2) are combined into field element (c_1, c_2).
- The ciphertext is $m_e = (x_1, y_1, c_1, c_2)$.

[3] Hankerson et al. 2004 pp. 79-81.

In the decryption process

- $(x_2, y_2) = k(x_1, y_1)$ is calculated where k is the private key.
- The plaintext (m_1, m_2) is obtained by $(c_1, c_2) - (x_2, y_2)$.

3.2. A Historical Perspective

The history of lattice reduction goes back to the theory of quadratic forms developed by Lagrange, Gauss, Hermite, Korkine-Zolotareff and others (Lagrange 1773, Gauss 1801, Hermite 1850, Korkine and Zolotareff 1773)[4] and to Minkowski's geometry of numbers (Minkowsi 1910).

With algorithmic number theory, the subject had attention around 1980. With the works the two famous problems have occurred: the shortest vector problem and the closest vector problem.

The very famous algorithm LLL computes a so-called reduced basis of a lattice and provides a partial answer to SVP. It runs in polynomial time and approximates the shortest vector within a factor of $2^{n/2}$. The name of this algorithm comes from the initials of the authors of the paper where the algorithm is proposed (Lenstra et al. 1982). This algorithm was the result of Lentra's work on integer programming (Lenstra 1983)[5]. Schnorr proposed a refinement for LLL algorithm which improved the above factor $(1-\varepsilon)^n$ (Schnorr 1987, Schnorr 1988). Babai developed an algorithm that approximates the closest vector by a factor of $\left(1/\sqrt{2}\right)^n$ (Babai 1986).

After these works lattice reduction started to take place in cryptography. Shamir proposed a polynomial time algorithm (Shamir 1982) breaking the Merkle-Hellman public key cryptosystem (Merkle and Hellman 1978) which was by then a unique alternative to RSA. Shamir used Lenstra's integer programming then Adleman extended Shamir's work by treating the cryptographic problem as a lattice problem rather than a linear programming problem (Adleman 1983). Lattice reduction has also been applied to other cryptographic context: against a version of Blum's protocol for exchanging secrets (Frieze et al. 1988), against cryptosystems based on truncated linear congruential

[4] Lagrange 1773, pp. 265-312.
[5] The results concering LLL were used before the work is published (around 1979).

generators (Frieze et al. 1988, Stern 1987), against cryptosystems based on rational numbers (Stern and Toffin 1990) or modular knapsacks (Joux and Stern 1991, Chee et al. 1991) and RSA with exponent 3 (Coppersmith 1996).

Ajtai discovered a connection between the worst-case complexity and the average-case complexity of some well known lattice problems (Ajtai 1996). Ajtai and Dwork proposed a cryptosystem using this theorem (Ajtai and Dwork 1997). The same year Goldreich, Goldwasser and Halevi proposed another cryptosystem based on lattice problems (Goldreich et al. 1997). Nguyen and Stern have broken the Ajtai-Dwork cryptosystem (Nguyen and Stern 1998). Later Nguyen proposed the cryptanalysis of the Goldreich, Goldwasser and Halevi cryptosystems (Nguyen, 1999).

Recently NTRU, firstly proposed by Hoffstein, Pipher and Silverman (Hoffstein et al. 1998), is being discussed under IEEE's 1363-1 standard named *Draft Standard for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices* (WEB_7 2007). Researches are being continued on NTRU and no crucial security issue is found.

CHAPTER 4

NTRU CRYPTOSYSTEM

This chapter is intended to provide the theoretical background on the lattice-based public key cryptosystem NTRU[6]. Section 4.1 describes domain parameters and some definitions for NTRU while section 4.2, section 4.3 and section 4.4 cover key generation, encryption and decryption operations respectively. Section 4.5 focuses on some considirations and finally the last section covers a propsed digital envelope for NTRU.

4.1. Domain Parameters and Some Definitions

The main parameters of NTRU cryptosystem are integers n, p and q.[7] These values are used to define the following polynomial rings:

- $R = \mathbb{Z}[X]/(X^n - 1)$ which specifies the polynomials modulo $X^n - 1$ with integer coefficients.
- $R_p = (\mathbb{Z}/p\mathbb{Z})[X]/(X^n - 1)$ which specifies the polynomials modulo $X^n - 1$ whose coefficients are reduced modulo p.
- $R_q = (\mathbb{Z}/q\mathbb{Z})[X]/(X^n - 1)$ which specifies the polynomials modulo $X^n - 1$ whose coefficients are reduced modulo q.

For secure implementation of NTRU, the parameters should also satisfy $\gcd(p, q) = 1$ where $p > q$ and n should be chosen as a prime number due to the reasons discussed in (Gentry 2001).

[6] http://www.ntru.com.

[7] In fact, it is possible to choose p to be a polynomial if the parameters are properly defined. However, we shall slightly ignore this case since our forthcoming discussion makes use of p as a fixed integer value of 2.

L_f, L_g, L_m and L_r are also parameters which represent some special subsets of the polynomial ring R from which particular polynomials are chosen to be used in key generation, encryption and decryption.

Throughout this thesis, all polynomials under our consideration have integer coefficients and generally belong to the ring R. In order to perform key generation, encryption and decryption operations, we need to specify some operations on polynomials.

Let *u*, *v* be arbitrary polynomials and *m* be a positive integer, then we can define following operations:

- $[u]_m$ or $u(\bmod m)$, is reducing the coefficients of *u* to a specified interval of length *m*, generally [0, *m*). However, we may take this interval to be $[A, A+m]$, for some integer *A*, in order to properly center the polynomial in some part of the decryption process.
- $u*v \pmod m$ or equivalently $u.v \pmod{m,\ X^n-1}$ is called (cyclic) convolution product or star multiplication. Here the point [.] is the usual polynomial multiplication, and $(\bmod m,\ X^n-1)$ means reducing the polynomial modulo X^n-1 and coefficients modulo *m*.
- For $u = u_0 + u_1 x + ... + u_{n-1} x^{n-1}$, we define

  $\text{Max } u = \max\ u_i$, $\text{Min } u = \min\ u_i$ and $\text{Width } u = \text{Max } u - \text{Min } u$.
- L^2-norm and centered L^2-norm of the polynomial *u* gives idea on the smallness or the length of *u* and are defined as

  $|u| = \sqrt[2]{\sum_{i=0}^{n-1} u_i^{\,2}}$ and $\|u\|_2 = \sqrt[2]{\sum_{i=0}^{n-1} \left(u_i - \bar{u}\right)^2}$ where $\bar{u} = \frac{1}{n}\sum_{i=0}^{n-1} u_i$ (Micciancio and Goldwasser 2002), (WEB_7 2007)[8].

For detailed information on domain parameters and their effect on the cryptosystem's security and performance the reader is referred to (WEB_7 2007).

[8] Micciancio and Goldwasser 2002 p. 8, WEB_7 2007 p. 45.

4.2. Key Generation

To create an NTRU public key, one chooses two polynomials such that $f \in L_f$ and $g \in L_g$. Here, polynomials in L_f and L_g have small widths. Also, the polynomial f should have inverses modulo p and q. In other words, one should be able to calculate f_p^{-1} and f_q^{-1} such that

$$f * f_p^{-1} \equiv 1 \pmod{p} \quad \text{and} \quad f * f_q^{-1} \equiv 1 \pmod{q}.$$

Private key is composed of the polynomials f and f_p^{-1}. After choosing the polynomials appropriately, public key can be computed as

$$h \equiv p f_q^{-1} * g \pmod{q}. \qquad \text{(Eq. 4.1)}$$

The following is a numerical example for key generation operation. Domain parameters are as follows:

$N = 11 \quad q = 32 \quad p = 3$

We choose a polynomial f such that it is invertible in both modulus p and.q.Also we choose the polynomial g which will be used in public key generation.

$f = -1 + x + x^2 - x^4 + x^6 + x^9 - x^{10}$ and $g = -1 + x^2 + x^3 + x^5 - x^8 - x^{10}$

Next step is to calculate the inverses of f in modulus p and q namely f_p and f_q:

$$f_p = 1 + 2x + 2x^3 + 2x^4 + x^5 + 2x^7 + x^8 + 2x^9 \text{ and}$$

$$f_q = 5 + 9x + 6x^2 + 16x^3 + 4x^4 + 15x^5 + 16x^6 + 22x^7 + 20x^8 + 18x^9 + 30x^{10}$$

f is the private and the public key h is calculated as follows:

$$h = p f_q * g = 8 + 25x + 22x^2 + 20x^3 + 12x^4 + 24x^5 + 15x^6 + 19x^7 + 12x^8 + 19x^9 + 16x^{10} \pmod{32}$$

4.3. Encryption

In order to perform encryption, one chooses a polynomial *m* representing the message such that $m \in L_m$, and a random polynomial $r \in L_r$. Later the polynomial corresponding to the ciphertext is computed as

$$e \equiv r * h + m \pmod{q}. \qquad \text{(Eq. 4.2)}$$

As in key generation, L_r and L_m are special sets of the polynomials in *R*, having small widths.

The following is a numerical example representing how the encryption process works.

In order to encrypt a message we need a public key *h* (we use the one which is calculated in the previous section) and a random polynomial *r* besides the message polynomial *m*. So we choose the random *r* and the message *m* as:

$$r = -1 + x^2 + x^3 + x^4 - x^5 - x^7 \text{ and } m = -1 + x^3 - x^4 - x^8 + x^9 + x^{10}$$

The following operation calculates the encrypted message (chipertext) *e*:

$$e = r * h + m = 14 + 11x + 26x^2 + 24x^3 + 14x^4 + 16x^5 + 30x^6 + 7x^7 + 25x^8 + 6x^9 + 19x^{10} \pmod{32}$$

4.4. Decryption

One can carry out the decryption by computing the polynomial

$$d \equiv f_p^{-1} * \left[f * e \right]_q \pmod{p}. \qquad \text{(Eq. 4.3)}$$

However, in some cases decryption may not be successful. The condition for successful decryption and its effects on the choice of parameters are briefly discussed in the next section.

The following is a numerical example to demonstrate the decryption operation. One can compute the temporary polynomial a as follows: (The chipertext, private key f and the inverse of private key f_p is provided from previous sections)

$$a = f * e = 3 - 7x - 10x^2 - 11x^3 + 10x^4 + 7x^5 + 6x^6 + 7x^7 + 5x^8 - 3x^9 - 7x^{10} \left(\text{mod} \quad 32\right)$$

The next step is to reduce the coefficients of a to modulo p. a results in polynomial a':

$$a' = a\left(\text{mod} \quad p\right) = -x - x^2 + x^3 + x^4 + x^5 + x^7 - x^8 - x^{10} \left(\text{mod} \quad 3\right)$$

Next we need to move the next step to calculate the plaintext:

$$c = f_p * a' = -1 + x^3 - x^4 - x^8 + x^9 + x^{10} \left(\text{mod} \quad 3\right)$$

4.5. Conditions

Consider the polynomial

$$\left[f * e\right]_q \equiv pr * g + f * m \quad (\text{mod}\, q)\,. \qquad \text{(Eq. 4.4)}$$

For different parameter sets $\left(n, p, q, L_f, L_g, L_r, L_m\right)$ it is probable that we will have the right hand side of equation 4.4 in the interval $\left[A, A+q\right)$, $A \neq 0$. Therefore, we need to center the value of $\left[f * e\right]_q$ by reducing its coefficients into the correct interval in order to satisfy equation 4.5.

$$\left[pr * g + f * m\right]_q = pr * g + f * m\,, \qquad \text{(Eq. 4.5)}$$

which guarantees the success of decryption.

Let $t = pr * g + f * m$. In some cases, the polynomial t may not be obtained easily due to the fact that it is not properly centered. This is called decryption failure. Although the probability of occurrence of decryption failure is significantly small for appropriately chosen parameter values, as discussed in (WEB_1 2006) and (WEB_13 2006), they should not be ignored (WEB_5 2006).

(WEB_12 2006) discusses different types of wrap and gap failures which are different types of decryption failures. (WEB_13 2006) calculates the probability of failures, and discusses methods in order to correctly center the polynomial *t* to eliminate the wrapping failures. However, gap failures still remains untreated. On the other hand, (Yu and He 2005) gives an algorithm to overcome all decryption failures. Furthermore, the same paper outlines an analysis relating the NTRU parameters to the decryption failures and presents the conditions for choosing the parameter values which prevents all decryption failures.

4.6. Digital Envelope

The original NTRU (Hoffstein et al. 1998), which is mainly outlined so far, considers the plaintext directly as the polynomial *m*. However, this scheme is vulnerable to some certain types of attacks and in particular, if decryption failure occurs (WEB_12 2006). For example, if the attacker is allowed to send a large number of messages and observe which ones are accepted as valid he/she can easily recover the messages. Therefore, calculation of the polynomial *m* is modified as in (WEB_1 2006) in order to improve the security of the cryptosystem.

Let $P_p(n-k)$ is the set of polynomials in R_p having degree at most $n-k-1$, and let $m' \in P_p(n-k)$ be the plaintext polynomial. Then, during the encryption, one can compute the polynomial *m* as follows:

$$m = \left[m' + G\left([r * h]_p \right) + H\left(m', [r * h]_p \right) X^{n-k} \right]_p . \qquad \text{(Eq. 4.6)}$$

Here, $G : P_p(n) \to P_p(n)$ and $H : P_p(n) \times P_p(n) \to P_p(k)$ are generating function and hashing function respectively.

In order to obtain m' in the decryption process, after computing *m*, we need to calculate the values $x = [e - m]_p$ and $y = \lfloor m - G(x) \rfloor_p$. It should be noted that in a valid decryption we expect the following equalities to hold $x = [r * h]_p$ and $y = m' + H\left(m', [r * h]_p\right) X^{n-k}$. Later, we extract two polynomials $y' \in P_p(n-k)$ and $y'' \in P_p(k)$ from *y* as

$$y = y' + y'' X^{n-k}. \qquad \text{(Eq. 4.7)}$$

If $y'' = H(y', x)$, it implies that $y' = H\left(m', [r * h]_p\right)$. Therefore, we conclude $y' = m'$ and decryption is valid.

Here, *k* is defined to be the security parameter of NTRU which provides resistance to some certain types of attacks and according to the chosen value of *k*, the probability of forging a valid ciphertext is p^{-k} (WEB_11 2006).

Lastly, similar and more secure padding schemes like the one discussed above are also designated in (WEB_2 2006, WEB_3 2006) and (WEB_5 2006) for particular chosen set of parameters.

CHAPTER 5

IMPLEMENTATION OF NTRU

This chapter aims to cover the details of how the parameters are chosen, the details of encryption scheme implemented and the details of our design of NTRU cryptosystem.

5.1. Instantiation of Cryptosystem

In this section, we outline some conditions which vitally affect the way the parameters are chosen. Also, we briefly mention the latest recommended and the alternative choices of the parameters in order to provide efficient and secure realizations of the cryptosystem. However, we do not cover any of these in full detail. For a complete discussion, one should refer to (WEB_5 2006), NAEP encryption scheme, and (WEB_6 2006), SVES-3 an instantiation of NAEP.

5.1.1. Choosing Parameters

Since NTRU is first proposed, the recommended parameter values have been subject to changes. Many different parameter choices are discussed in the literature in order to provide different levels of efficiency and security, and in general, for each proposed set of parameters and defined security levels, the parameter p is fixed to be a small integer or polynomial value.

In order to realize efficient implementation of NTRU at least one polynomial in the convolution product should be binary, whose coefficients are in the set $\{0,1\}$, or trinary, whose coefficients are in $\{-1,0,1\}$. Therefore, in the rest of our discussion we

shall define d_z to be the number of coefficients in the polynomial binary or trinary polynomial z which are equal to 1.

5.1.2. Choosing n

If the message is binary, n is the number of bits that can be transported. In order to provide k bits of security[9] and prevent some particular (birthday-like) attacks, $2k$ bits can be transported. In addition, SVES-3 uses k bits of random padding to gain security against enumeration attacks in case some low-entropy messages are transported. Therefore, we set n to be the first prime number greater than $3k$. It should be noted that n might need being changed if one cannot find appropriate values of the remaining parameters.

5.1.3. Choosing *f*, *g*, *r* and *m*

Let F, g and r to be binary polynomials with d_f, d_g and d_r number of 1s respectively. We take $f = 1 + pF$ so that the second convolution product in the decryption can be eliminated since f_p^{-1}. Furthermore, since security increases when h is invertible, we also take g to be invertible and set $d_g = n/2$ to obtain the best lattice security, and choose smallest d_F, d_r and d_m such that

$$\frac{1}{\sqrt{n}}\left(\frac{n/2}{d'/2}\right) \geq 2^k$$

where $d' \in \{d_F, d_r, d_m\}$. Here, we can take $d_F = d_r = d$ in order to equalize the combinatorial security levels of F and r. Moreover, the message representative polynomial m is chosen in such a way that it does not contain very few 1s or very few 0s. Also, $\|m\|_2$ should be sufficiently large to provide resistance against attacks which stems from information leakage from the encrypted message, and we should have the

[9] Bits of Security (also known as Security Strength): Number of operations to break a cryptosystem.

probability of being rejected due to having insufficient security, P_{reject}, very small, for instance less than 2^{-40}.

5.1.4. Choosing *p* and *q*

It is already noted that p and q should be relatively prime. p is fixed to be the integer value of 2 so that we can work with binary polynomials. Also, q must have a higher order modulo n, i.e. the order of divisors of X^{n-1} modulo q should be high, for example $(n-1)$ or $(n-1)/2$. In addition, to achieve better lattice security we must keep f and g as large as possible relative to q. Though, for combinatorial security, it is better to increase p, it causes an increase in q and so decreases lattice security. As a result, we can select q as a prime number such that

$$q \le p.\min\left(d_r, d_g\right)+1+ p.\min\left(d_f, n/2\right)$$

and

$$\text{order of q modulo } n \ge (n-1)/2.$$

This choice gives us the best lattice security and zero probability of decryption failure.

5.1.5. Alternatives

It is possible to choose f not to be of form $1+pF$ in order to decrease q. On the other hand, F and r can be chosen in the product form $f_1 * f_2 + f_3$ in order to obtain further performance benefits and slightly increased bandwidth.

We can also choose *p* and *q* values differently. Let *s* be the first power of 2 such that

$$s \ge p(1).\min\left(d_r, d_g\right)+1+ p(1).\min\left(d_f, n/2\right)$$

Then for a small integer or polynomial value of $p = 2 + X$ or $p = 3$, in which cases we work with binary or trinary polynomials respectively, one can choose q = s.

This speeds up the reductions modulo q. However, with the larger values of p lattice security worsens due to the fact that q gets larger. In addition, we can also speed up these reductions by choosing q to be the largest prime such that $q \le s$ for $p = 2$ at the expense of lattice security.

As a last note, allowing the probability of decryption failures to be greater than 0 reduces q, thus improves the lattice security and the bandwidth.

5.1.6. NAEP Encryption Scheme

Let B_n be the set of binary polynomials whose degree is less than *n*, and $B_n(d)$ be the subset of B_n with polynomials having *d* number of 1s. Furthermore, let *G* and *H* be two hashing functions such that

$$G : B_{n-k} \times B_k \to B_n(d_r)$$

$$H : B_n \to B_n.$$

These functions should be chosen such that each of them has a very small probability of variation in running time, since the running time variations may cause leakage of information about the private key (WEB_14 2007).

5.1.6.1. Encryption

During encryption we choose a random polynomial $b \in B_k$, and then we calculate the polynomial $r \in G(m', b)$, where m' is the plaintext polynomial. Message representative polynomial *m* is given

$$m \equiv \left(m' + bX^{n-k}\right) + H\left(\left[\left[r * h\right]_q\right]_p\right) \pmod{p}. \qquad \text{(Eq. 5.1.)}$$

At this point, one should check whether *m* has the expected level of combinatorial security. If not the operation should be performed with a different and randomly chosen *b*.

For properly computed *m*, encryption is performed as defined before:

$$e \equiv r * h + m \pmod{q}.$$

5.1.6.2. Decryption

In decryption, one, first, calculates the polynomial m as described before:

$$m = \left[f_p^{-1} * \left[f * e \right]_q \right]_p .$$

Of course, the polynomial $\left[f * e \right]_q$ should be centered if decryption failure occurs.

In order to obtain m', we need to calculate the values

$$x = e - m \text{ and } y = \left[m - H\left(\left[\left[x \right]_q \right]_p \right) \right]_p .$$

Later, we extract two polynomials $y' \in B_{n-k}$ and $y'' \in B_k$ from *y* as

$$y = y' + y'' X^{n-k} .$$

If the conditions

$$x = \left[G\left(y', y'' \right) * h \right]_q \text{ and } y' \in B_{n-k}\left(d_m \right)$$

are satisfied, the ciphertext is valid and $y' = m'$ is the plaintext.

5.2. Implementation Details

The source code for this thesis is developed compatible with IEEE's 1363.1 standard namely "*Draft Standard for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices*" (WEB_7 2007). Eclipse with CDT[10] plugin is used as the development environment for this project.

[10] http://www.eclipse.org and http://www.eclipse.org/cdt.

5.2.1. Programming Language

ANSI C is selected as development language. The practical performance for the project is very important and it is obvious that the more the programming language is low level the more performance we gain. Pointer arithmetic, structural features and portability of ANSI C were the main factors for us to choose ANSI C as our programming language. Also the fact that CRYMPIX (Hışıl 2005) is developed using ANSI C, played a very huge role for us to choose ANSI C for compatibility reasons.

5.2.2. Representation of Polynomials

In the polynomial structure the number of coefficients, the degree of the polynomial and the array that containing the actual coefficients are stored.

5.2.3. Memory Management

Allocation and deallocation of objects bring a huge overhead to software in terms of runtime performance. So one has to get rid of this overhead to provide better runtime performance. To eliminate these types of performance losses a memory management model called *Static Memory Management* (Mersin and Beyazıt 2007) for this implementation is used. In this model, a defined number of memory slots, used as polynomials for this project, are allocated before the application begins and stored in a memory slot stack. When ever one needs a polynomial, the initialization process starts and an available memory slot is provided to the user (developer) (This can be imagined as a pop stack operation). In deallocation of a polynomial, one return the polynomial to the software, this memory slot is put back to the stack; from now on this particular memory slot is available for further use. (This can be imagined as a push stack operation) Static Memory Model is used for this implementation because, for this thesis,

the main concern is the performance. The assumption here is that we do not have memory restrictions.

5.2.4. Details of Some Instructions

The following Figure 5.1 and Figure 5.2 illustrate the two core operation of the NTRU cryptosystem. Those are addition and convolution product operations.

```
input: a, b, m
output: Returns c = a + b (coefficient mod m)

if a→n > b→n then
   n = a→n, sn = b→n.
   for i = sn to n do
      c→coef[i] = a→coef[i].
   end
else
   n = b→n, sn = a→n.
   for i = sn to n do
      c→coef[i] = b→coef[i].
   end
for i = sn tao n do
   c→coef[i] = a→coef[i] + b→coef[i] (mod m).
end

return c.
```

Figure 5.1. Add Operation.

```
input: a, b, n, m
output: Returns c = a * b (coefficient mod m) (degree mod n)

for i = 0 to a→n do
   for j = 0 to b→n do
      cur_i = i + j (mod n).
      co_a = a→coef[i].
      co_b = b→coef[i].
      c→coef[cur_i] =c→coef[cur_i] + (co_a * co_b (mod m)) (mod m).
   end
end

return c.
```

Figure 5.2. Convolution Product Operation.

CHAPTER 6

PERFORMANCE ANALYSIS OF NTRU

The underlying theory implies that NTRU can be yet another popular public key cryptosystem residing with ECC[11], RSA[12] and the likes. Nevertheless, it is important to make detailed discussion of these cryptosystems in order to better comprehend how NTRU performs. This section intends to show the comparative performance analysis of lattice based NTRU cryptosystem with respect to popular public key cryptosystems; like RSA and ECC. Next subsections will give the results of this analysis in terms of key size and key generation, encryption and decryption timings.

6.1. Parameters of NTRU Used for Comparison

Using the conditions for recommended parameters in Chapter 5, one can obtain the following sets for the parameter values in Table 6.1.

Table 6.1. NTRU Parameter Sets.

k (bit security)	$n(\geq 3k)$ (degree)	p (small modulus)	q (large modulus)	$d\left(=d_f=d_r\right)$ (number of 1s in f and r)	$d_g\left(=\lfloor n/2 \rfloor\right)$ (number of 1s in g)	$d_{m_0}\left(\leq d_m\right)$ (number of 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

[11] http://www.certicom.com.
[12] http://www.rsa.com.

It should be noted that, besides complying the discussion we made so far, the parameter values in Table 6.1. are also recommended as SVES parameter choices in (WEB_7 2007), the latest IEEE draft standard (currently draft 9) for public key cryptosystems based on hard problems over lattices.

In the rest of our study, while making comparison of NTRU with other public key cryptosystems, we shall refer to NTRU instantiated with these sets of parameters.

6.2. Performance Comparison of NTRU with ECC and RSA

6.2.1. Comparison of Key Sizes

In ECC and RSA, public and private keys can be chosen of approximately equal lengths, whereas NTRU public key size differs from private key size with a ratio of $\frac{n}{n-k}\log_p q - to - 1$. The public key size of a cryptosystem gives useful insight on the bandwidth usage if the cryptosystem is intended to be used in key exchange schemes. Table 6.2. gives corresponding NTRU, ECC and RSA keys sizes for equivalent security levels (k) of 80 bits, 112 bits and 128 bits etc (WEB_9 2007, WEB_6 2006).

Table 6.2. Public Key Sizes (in bits).

Security Level (bits)	Public Key Sizes (bits)		
	NTRU	ECC	RSA
80	2008	160	1024
112	3033	224	2048
128	3501	256	3072
160	4383	320	4096
192	5193	384	7680
256	7690	521	15360

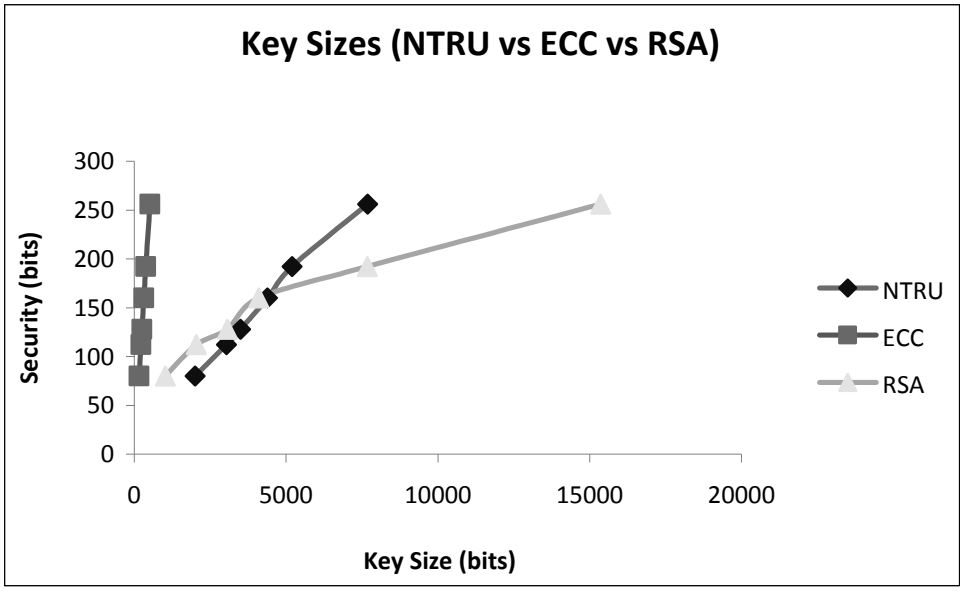

Figure 6.1. Key Size Comparison Graph.

From Table 6.2. and Figure 6.1., one can observe that ECC, among the three, makes the best use of bandwidth and NTRU's bandwidth usage becomes more efficient with respect to RSA as the security level increases.

6.2.2. Comparison of Key Generation, Encryption and Decryption Performance

Though RSA is the most studied, tested and scrutinized cryptosystem (among the three), the latest debates, such as in (WEB_9 2007), point out that ECC gained significant trust over time, and now, many security vendors are including ECC modules in their own products.

Accordingly, we find it useful to give timing comparisons with ECC. On the other hand, preliminary timing comparisons with RSA can be found in Table 6.3 (WEB_1 2006). Figure 6.2., Figure 6.3. and Figure 6.4. give the graphical representation of key generation, decryption and encryption performance comparisons of NTRU and RSA respectively.

Table 6.3. Comparison of NTRU and RSA.[13]

System	Security (MIPS years)	Public Key Size (bits)	Create Key (msec)	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 6.3.[14]

- Security is measured in MIPS-years required to break the system.
- 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 is done on a 255 MHz Digital AlphaStation.
- RSA encryption/decryption programmed in Microsoft Visual C++ 5.0 (optimized for speed, Pentium Pro code generation), and run on a Pentium II 266MHz machine under Windows NT 4.0. RSA encryption uses exponent 17 to increase speed.

Figure 6.2., Figure 6.3. and Figure 6.4. give information about how NTRU acts against RSA. For another point of view, Figure 6.5., Figure 6.6. and Figure 6.7. show the key size versus performance graphs.

[13] WEB_1 2006.
[14] The reader is referred to WEB_1 2006 p. 10 for further details on Table 6.3.

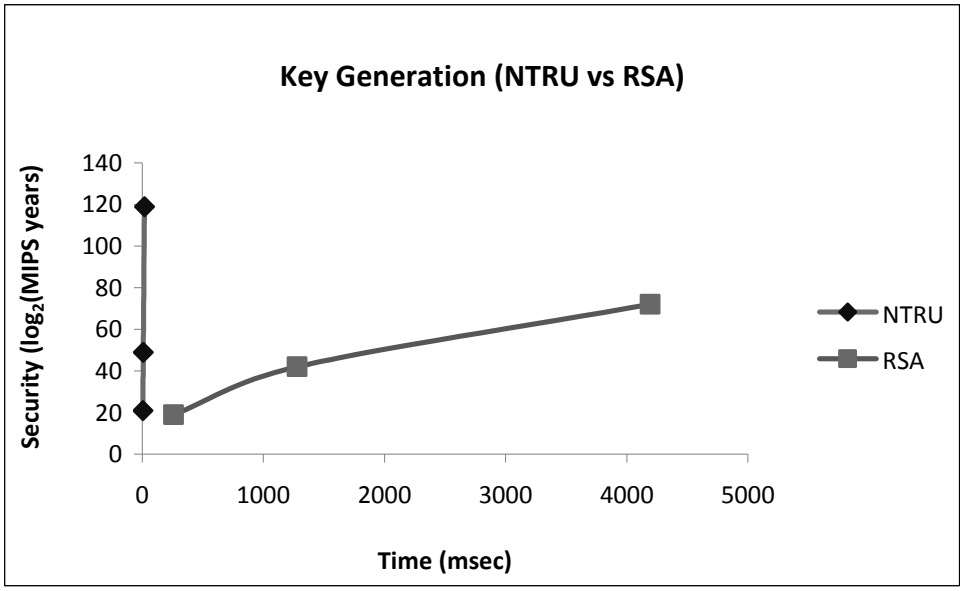

Figure 6.2. Key Generation Performance Graph (NTRU and RSA).

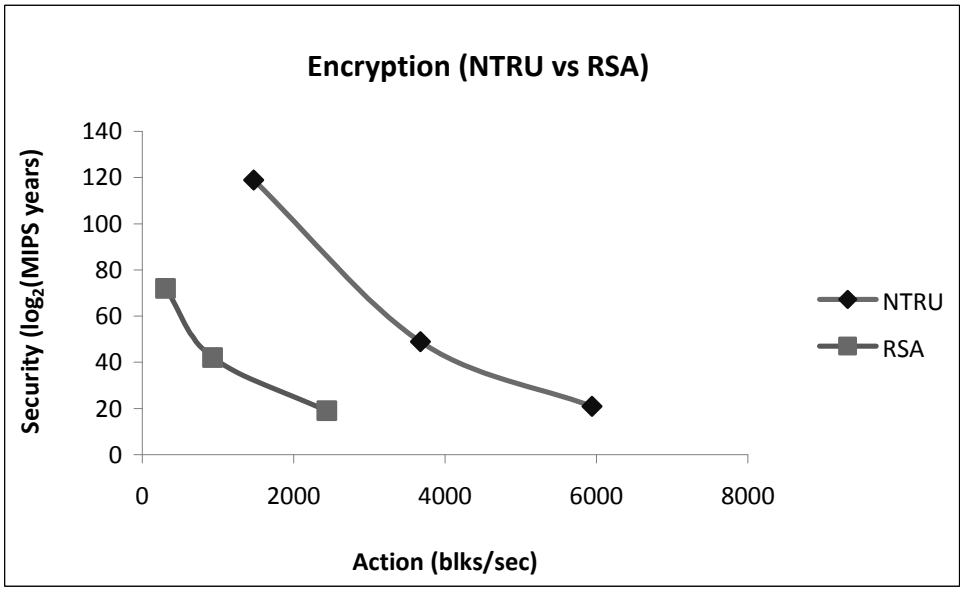

Figure 6.3. Encryption Performance Graph (NTRU and RSA).

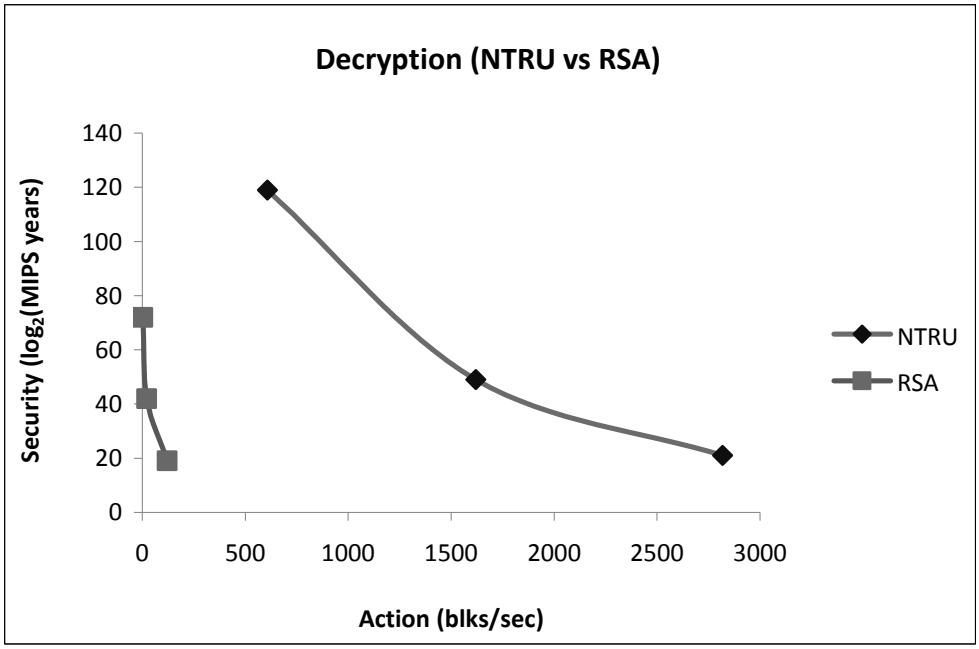

Figure 6.4. Decryption Performance Graph (NTRU and RSA).

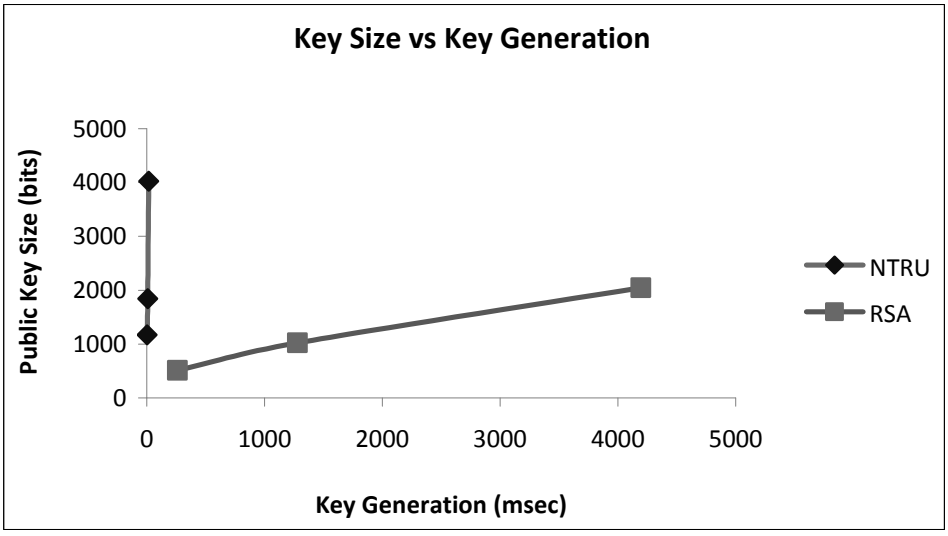

Figure 6.5. Key Size vs Key Generation Performance Graph (NTRU vs RSA).

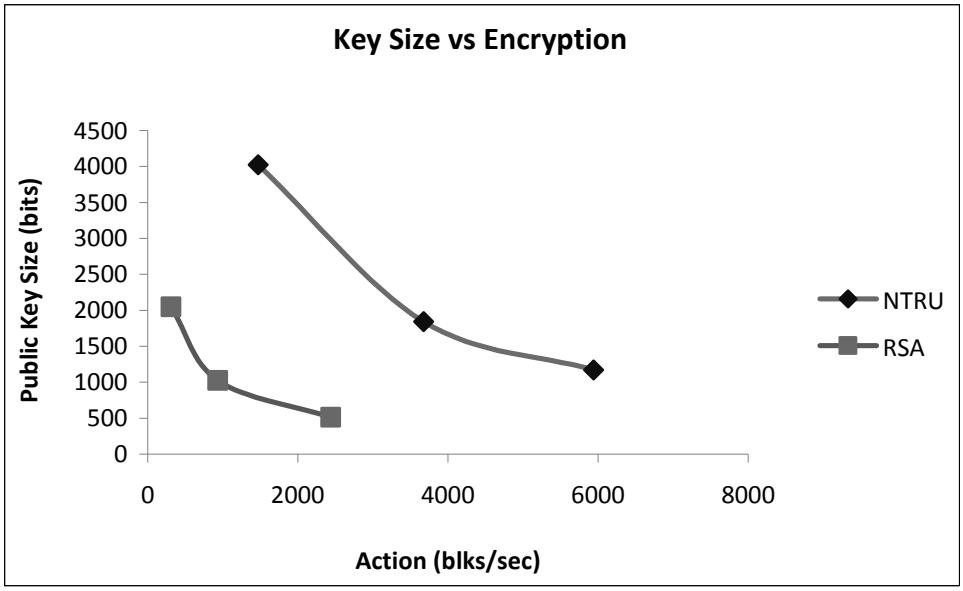

Figure 6.6. Key Size vs Encryption Performance Graph (NTRU and RSA).

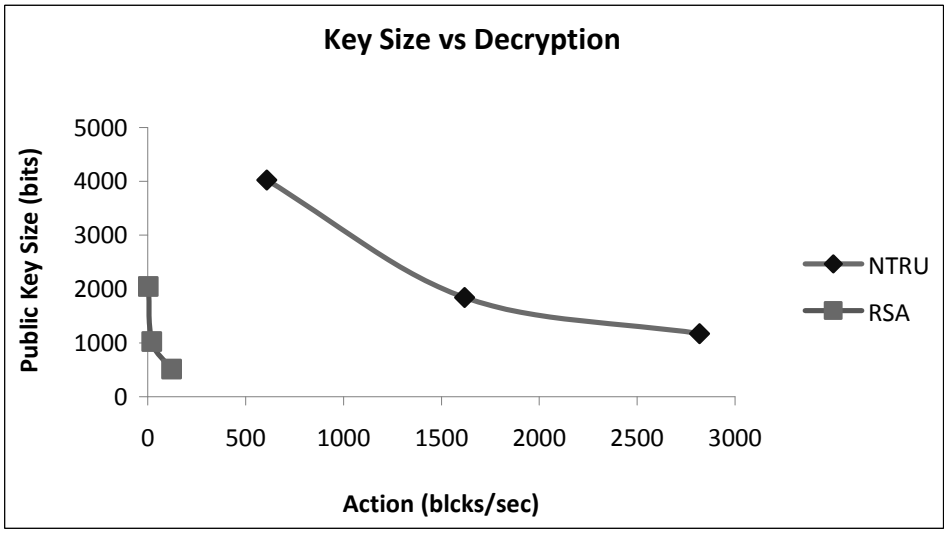

Figure 6.7. Key Size vs Decryption Performance Graph (NTRU and RSA).

From Table 6.3 one can drive that NTRU has better performance values than RSA. From Figure 6.2, Figure 6.3 and Figure 6.4 it is easy for us to conclude that NTRU has better performance. Figure 6.5., Figure 6.6 and Figure 6.7. show that for same key sizes NTRU can encrypt / decrypt more message in a second than RSA, also key generation for same key sizes is faster in NTRU.

We make use of C code implemented in GMP[15], which is the outcome of (Atay 2006), while measuring key generation, encryption and decryption times of ECC. The curves used are NIST and/or SEC - Certicom recommended elliptic curves over prime fields (WEB_8 2006, WEB_10 2006), and encryption and decryption measurements are taken in several coordinate systems such as, affine, projective, Jacobian, Chudnovsky, and modified Jacobian.

Table 6.4. gives timing measurements for NTRU and ECC cryptosystems where the code is compiled (with no optimizations) and run on a personal computer with Windows XP Professional OS, P4 2.80 GHz CPU and 1 GB RAM. Figure 6.5., Figure 6.6. and Figure 6.7. respectively give the graphical representation of key generation, decryption and encryption performance comparisons of NTRU and the *best* timing results of ECC among five coordinate systems while Figure 6.8., Figure 6.9. and Figure 6.10. give the graphical representation of key generation, decryption and encryption performance comparisons of NTRU and the *worst* timing results of ECC among five coordinate systems.

Table 6.4. Key Generation, Encryption and Decryption Times.

Cryptosystem	Security Level (bits)	Key Generation* (msec)	Encryption* (msec)	Decryption* (msec)
NTRU-251	80	75.65	1.68	8.22
ECC-192	between 80 - 112	57.87 – 152.73	37.81 – 116.39	19.15 –57.68
NTRU-347	112	144.16	3.11	15.70
ECC-224	112	234.11 – 367.98	52.52 – 164.50	26.35 – 81.52
NTRU-397	128	188.92	3.97	20.26
ECC-256	128	478.22 – 656.63	68.72 – 223.29	35.00 – 111.16
NTRU-491	160	288.31	5.97	30.96
NTRU-587	192	412.10	8.42	44.42
ECC-384	192	947.43 – 1429.11	182.35 – 586.20	90.61 – 290.94
NTRU-787	256	738.75	14.49	79.48
ECC-521	256	2055.04 – 3175.87	423.25 – 1257.56	211.35 – 626.33

[15] http://www.swox.com/gmp.

* ECC timings are given as minimum - maximum of the values observed over all coordinate systems. In key generation minumum values are in Affine coordinate system while maximum values are in Madified Jacobian coordinate system. In both encryption and decryption, minimum values are in Chudnosky coordinate system and maximum values are in Affine coordinate system.

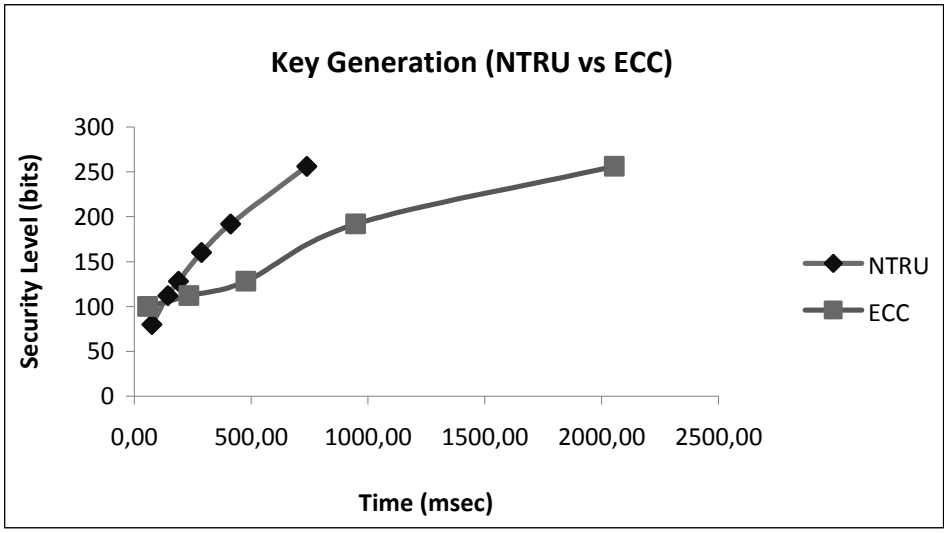

Figure 6.8. Key Generation Performance Graph (NTRU ECC-b[16]).

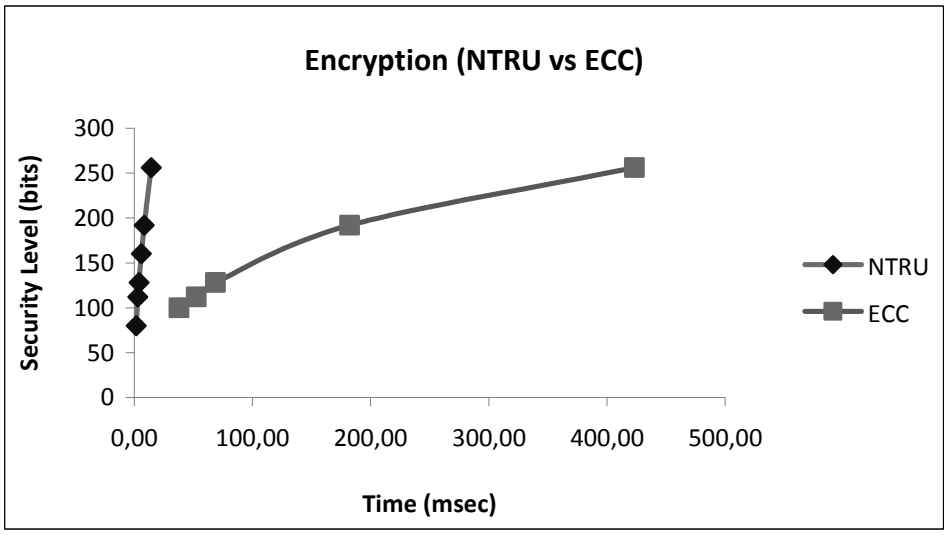

Figure 6.9. Encryption Performance Graph (NTRU and ECC-b).

[16] ECC-b: minimum values of ECC timings observed.

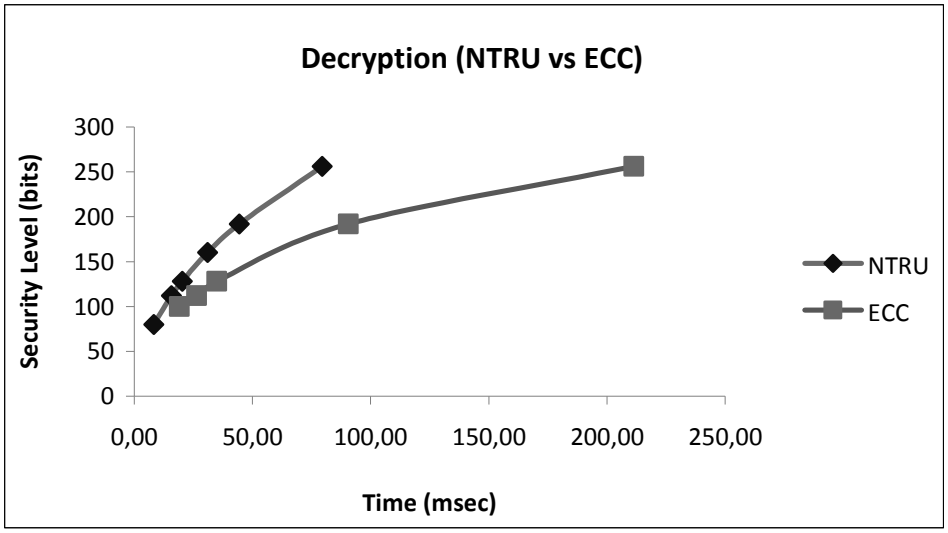

Figure 6.10. Decryption Performance Graph (NTRU and ECC-b).

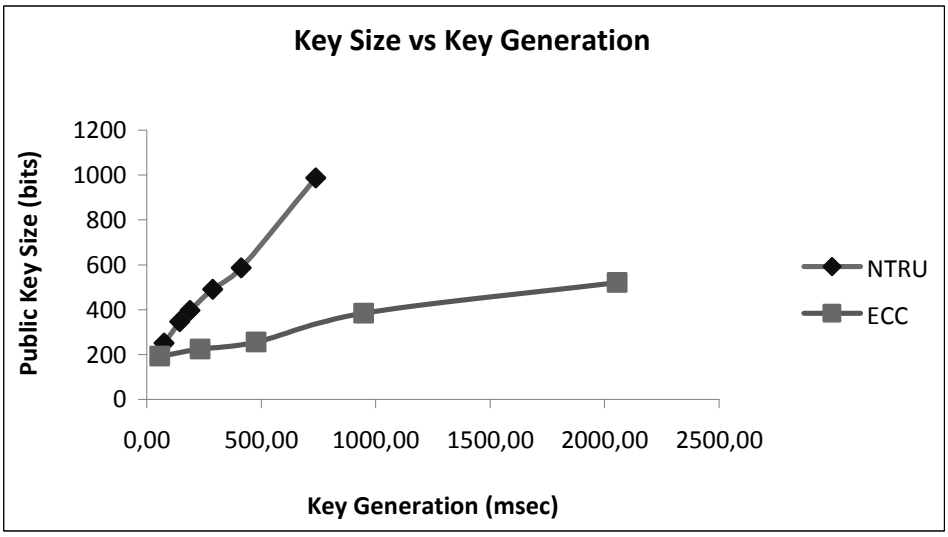

Figure 6.11. Key Size vs Key Generation Performance Graph (NTRU vs ECC-b).

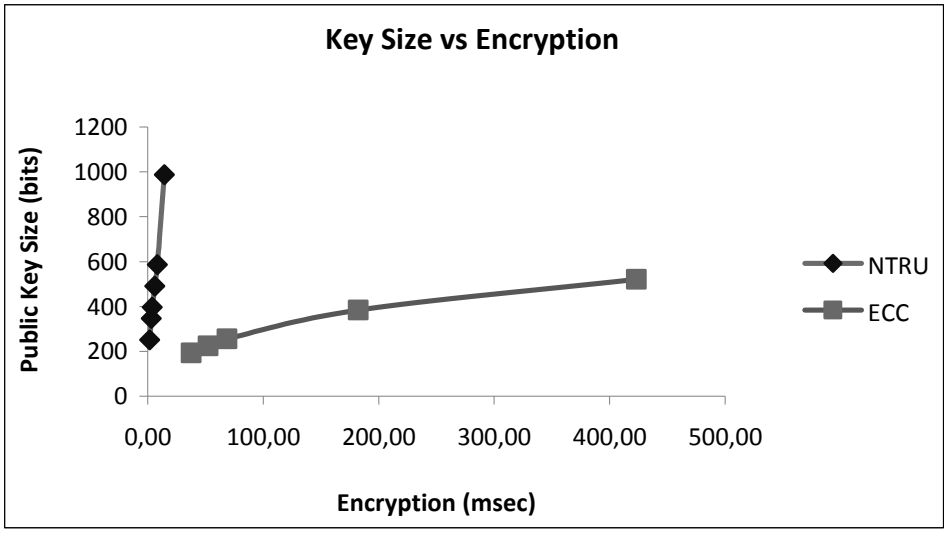

Figure 6.12. Key Size vs Encryption Performance Graph (NTRU and ECC-b).

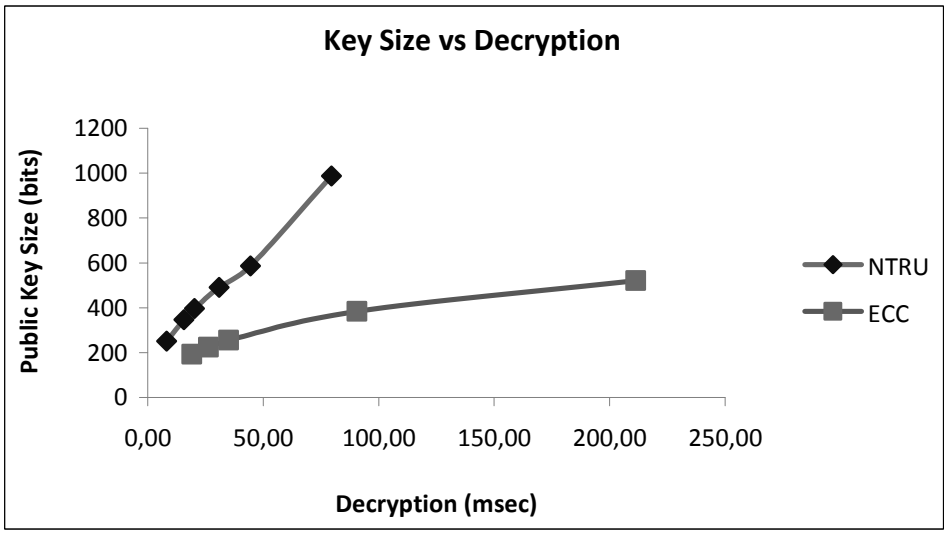

Figure 6.13. Key Size vs Decryption Performance Graph (NTRU vs ECC-b).

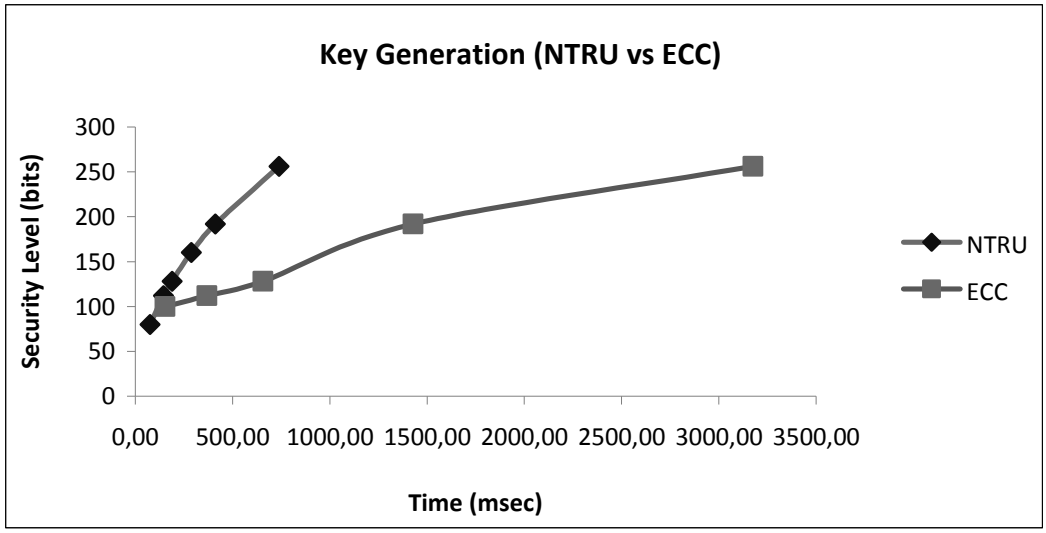

Figure 6.14. Key Generation Performance Graph (NTRU ECC-w[17]).

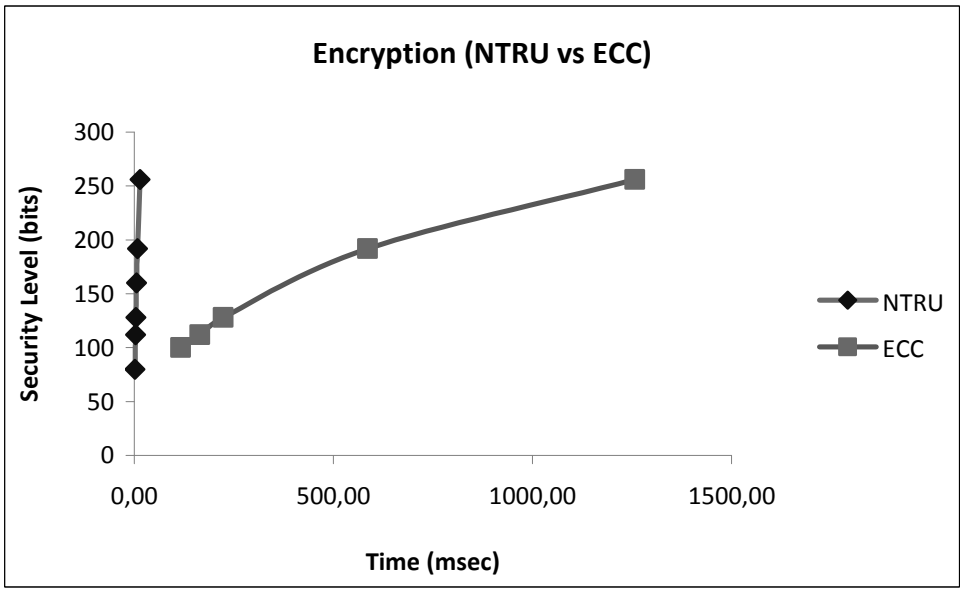

Figure 6.15. Encryption Performance Graph (NTRU and ECC-w).

[17] ECC-w: maximum values of ECC timings observed.

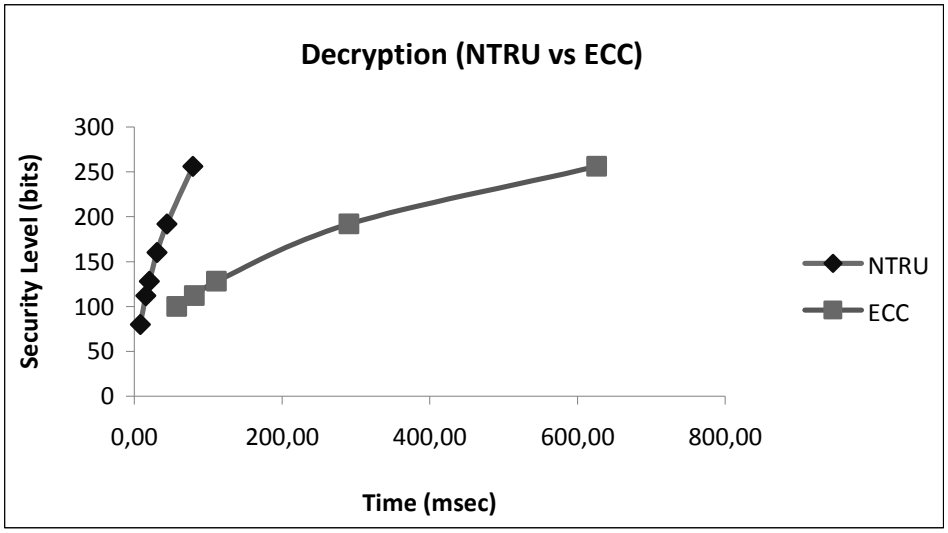

Figure 6.16. Decryption Performance Graph (NTRU and ECC-w).

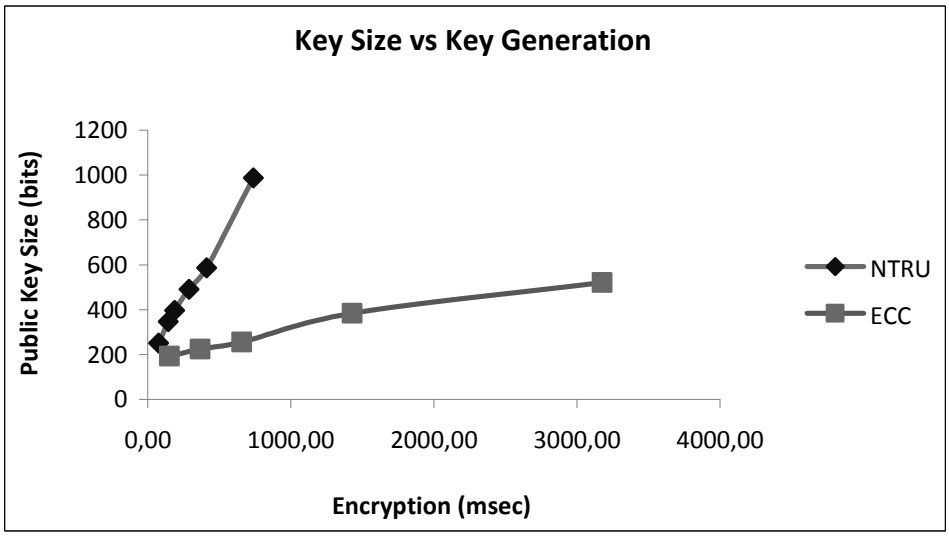

Figure 6.17. Key Size vs Key Generation Performance Graph (NTRU vs ECC-w).

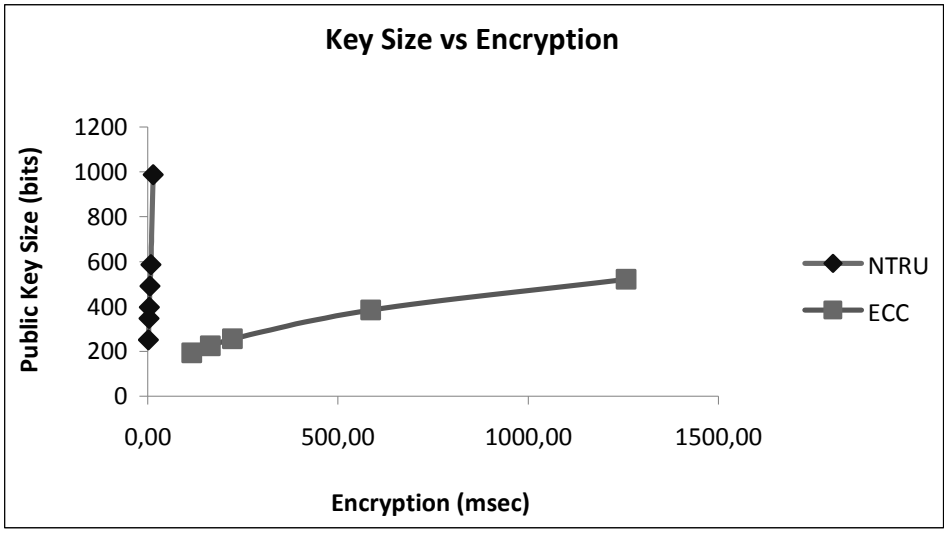

Figure 6.18. Key Size vs Encryption Performance Graph (NTRU and ECC-w).

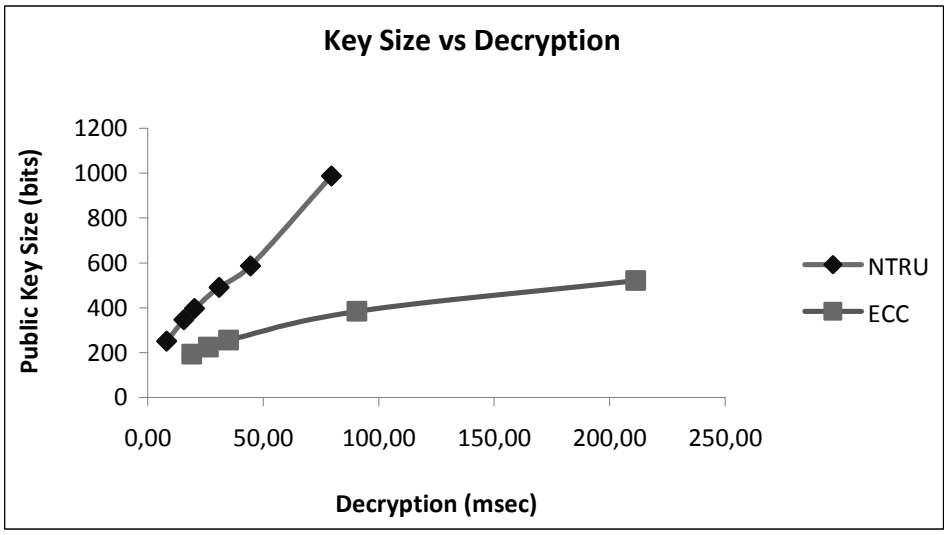

Figure 6.19. Key Size vs Decryption Performance Graph (NTUR and ECC-w).

As one can derive from Table 6.4, NTRU seems faster than ECC with respect to all the security levels defined above. This mainly stems from the fact that NTRU operations are relatively simple and not as demanding as ECC's. For instance, they do not even require the use of multiprecision arithmetic in the sense ECC operations do. In addition, though the timing measurements can be affected by many things such as runtime environment, compiler options, code optimizations and, in case of ECC, the

pros and cons of using a general purpose multiprecision arithmetic library etc., it is highly unlikely that ECC has better runtime performance.

Figure 6.8., Figure 6.9. and Figure 6.10. show that NTRU has better performance values compared with ECC best performance values. Also Figure 6.11., Figure 6.12. and Figure 6.13. indicate that for the same key sizes NTRU has better performance. Figure 6.14., Figure 6.15., Figure 6.16., Figure 6.17., Figure 6.18. and Figure 6.19. show that NTRU has better, naturally, performance values than ECC's worst performance values.

CHAPTER 7

CONCLUSION

In this study, we aimed to gain the knowledge of lattice based cryptography and investigate the performance of lattice based cryptography. As the study continued, we noticed that NTRU is the only, for now, candidate to be used as a secure and affordable public key cryptosystem among all lattice based cryptosystems. So we concentrate on the performance comparisons of NTRU and other popular public key cryptosystems, namely RSA and ECC. In order to achieve these goals; we coded NTRU cryptosystem compatible with "IEEE 1363.1 Draft Standard for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices" standard. We used ANSI C as our programming language.

From Chapter 6 one can derive that among all three ECC has the best key size. At lower security levels RSA has smaller key sizes than NTRU's. But as the security level increases; the key size of RSA increases more than NTRU's. From key size point of view ECC has the best results where for lower security levels RSA is better than NTRU.

When the time has come to the actual performance analysis, we obtained that for the same security levels NTRU has better performance than ECC's and RSA's. Though the timing measurements can be affected by many things such as runtime environment, compiler options, code optimizations and, in case of ECC and RSA, the pros and cons of using a general purpose multiprecision arithmetic library etc., it is highly unlikely that ECC and RSA have better runtime performance.

Throughout the study we outlined NTRU and demonstrated, in basic terms, how it performs against RSA and ECC.

Many researchers have been scrutinizing NTRU since the time it proposed for the first time. There has been serious amount of analysis on the security and performance issues, and NTRU seems to be quite a decent public key cryptosystem which will be useful in the field of security for many years to come.

At last NTRU is a promising alternative for the future of public key cryptography. The nature of the problem which NTRU is based on (Geometrical

problems) is completely different than the current popular public key cryptosystems' (Number theoretic problems). Because of this reason even though current public key cryptosystems are broken, NTRU will be the reserve for the future use. NTRU is a relatively new cryptosystem and it seems that we can not replace ECC or RSA with NTRU so easily. There is more way to go for this purpose.

REFERENCES

Adleman, L. M., 1983. "On Breaking Generalized Knapsack Public Key Cryptosystems", *Proceedings of the 15th ACM Symposium on Theory of Computing*, pp. 402-412, ACM Press.

Ajtai, M., 1996. "Generating Hard Instances of Lattice Problems", *Proceedings of the 28th ACM Symposium on Theory of Computing, Philadelphia, USA*, pp.99-108, ACM Press.

Ajtai, M., 1998. "The Shortest Vector Problem in L^2 is NP-hard for Randomized Reductions", *Proceedings of the 30th Annual ACM Symposium on Theory of Computing, New York, USA*, pp. 10-19, ACM Press.

Ajtai, M. and Dwork, C., 1997. "A Public Key Cryptosystem with Worst-Case / Average-Case Equivalence", *Proceedings of the 29th Annual ACM Symposium on Theory of Computing, El Paso, Texas, United States*, pp. 284-293, ACM Press.

Atay, S., 2006. "Performance Issues of Elliptic Curve Cryptographic Implementations", Unpublished Ph.D. Dissertation Thesis, Ege University, Graduate School of Natural and Applied Sciences.

Babai, L., 1986. "On Lovasz Lattice Reduction and the Nearest Lattice Pint Problem", *volume 9 of Combinatorica*, pp. 1-13.

Cai, J. Y., 2000. "The Complexity of Some Lattice Problems", *Proceedings of the 4th International Symposium on Algorithmic Number Theory, volume 1838 of Lecture Note in Computer Science*, pp. 1-32, Springer-Verlag.

Chee, Y. M., Joux, A. and Stern, J., 1991. "The Cryptanalysis of a New Public Key Cryptosystem Based on Modular Knapsacks", *Proceedings of Crypto'91, volume 576 of LNCS*, pp. 204-212, Springer-Verlag.

Coppersmith, D., 1996. “Finding a Small Root of a Univariate Modular Equation”, *Proceedings of EUROCRYPT '96 volume 1070 of Lecture Note in Computer Science*, pp. 155-165, IACR, Springer.

Frieze, A. M., Hastad, J., Kannan, R., Lagarias, J. C. and Shamir, A., 1988. “Reconstructing Truncated Integer Variables Satisfying Linear Congruences”, *SIAM Journal on Computing, volume 17, no. 2*, pp. 262-280.

Gauss, C. F., 1801. “Disquisitiones arithmeticae”, (Leipzig)

Gentry, C., 2001. “Key Recovery and Message Attacks on NTRU-Composite”, *Proceedings of EUROCRYPT '01: the International Conference on the Theory and Application of Cryptographic Techniques, London, UK*, pp. 182-194, Springer-Verlag.

Goldreich, O., Goldwasser, S. and Halevi, S., 1997. “Public Key Cryptosystems from Lattice Reduction Problems”, *Proceedings of CRYPTO '97: the 17th Annual International Cryptology Conference on Advances in Cryptology, London, UK*, pp. 112-131, Springer-Verlag.

Hankerson, D., Menezes, A. and Vanstone, S. 2004. “Guide to Elliptic Curve Cryptography” (Springer-Verlag, New York).

Hermite, C., 1850. “Extraits de Lettres de M. Hermite à M. Jacobi sur Différents Objets de la Théorie des Nombres, Deuxième Lettre”, (J. Reine Angew) pp. 279-290.

Hışıl, H., 2005. “A Distributed Multiprecision Cryptographic Library Design”, Unpublished MS. Thesis, İzmir Institute of Technology, The Graduate School of Engineering and Science.

Hoffstein, J., Pipher, J. and Silverman, J.H., 1998. “NTRU: A Ring Based Public Key Cryptosystem”, *Proceedings of ANTS III, Portland, Oregon, USA, volume 1423 of Lecture Note in Computer Science*, pp. 267-288, Springer-Verlag.

Howgrave-Graham, N., Nguyen, P.Q., Pointcheval, D., Proos, J., Silverman, J.H., Singer, A. and Whyte, W., (2003). "The Impact of Decryption Failures on the Security of NTRU Encryption", *volume 2729 of Lecture Note in Computer Science*, pp. 226-246, Springer-Verlag.

Joux, A. and Stern, J., 1991. "Cryptanalysis of Another Knapsack Cryptosystem", *Proceedings AsiaCrypt'91, volume 739 of Lecture Note in Computer Science*, pp. 470-476, Springer-Verlag.

Korkine, A. and Zolotarev, G., 1873. "Sur les Formes Quadratiques" *volume 261 of Mathematische Ann.* pp. 336-389

Langrange, L., 1773. "Recherches d'Arithmétique", (Nouv. Mém. Acad., Berlin).

Lenstra, A.K, Lenstra, Jr. H.W. and Lovász, L., 1982. "Factoring Polynomials with Rational Coefficients" *volume 261 of Mathematische Ann.*, pp. 513-534.

Lenstra, Jr. H. W., 1983. "Integer Programming with a Fixed Number of Variables", *volume 8 of Mathematics of Operations Research*, pp. 538-548.

Merkle, R. C. and Hellman, M. E., 1978. "Hiding Information and Signatures in Trapdoor Knapsacks", *volume 24 no 5 of IEEE Transactions on Information Theory*, pp. 525-530, IEEE.

Mersin, A. and Beyazıt, M., 2007. "A Memory Management Model for Cryptographic Software Libraries", *In Proceedings of International Conference on Security of Information and Networks, Gazimagusa, TRNC*, to be published.

Micciancio, D., 1998. "The Shortest Vector in a Lattice is Hard to Approximate to Within Some Constant", *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pp. 92-98, IEEE.

Micciancio, D. and Goldwasser S. 2002. "Complexity of Lattice Problems a Cryptographic Perspective", (Kluwer Academic Publishers, USA).

Minkowski, H., 1910. “Geometrie der Zahlen”, (Teubner, Leipzig).

Nguyen, P., 1999. “Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem”, *Proceedings of Crypto ’97, volume 1666 of Lecture Note in Computer Science*, pp. 288-304, Springer-Verlag.

Nguyen, P. and Stern, P., 1998. “A Converse to the Ajtai-Dwork Security Result and Its Cryptographic Implications”, submitted to Symposium on Theory of Computing 98.

Nguyen, P. and Stern, J., 2001. “The Two Faces of Lattices in Cryptology”, *volume 2146 of Lecture Note in Computer Science*, pp. 146-180, Springer-Verlag.

Rivest, R., Shamir. A. and Adleman, L., 1978. “A Method for Obtaining Digital Signatures and Public Key Cryptosystems”, *volume 21 Issue 8 of Communications of the ACM*, pp. 120-126, ACM Press.

Schnorr, C.P., 1987. “A Hierarchy of Polynomial Lattice Basis Reduction Algorithms”, *volume 53 of Theoretical Computer Science*, pp. 201-224.

Schnorr C.P., 1988. “A More Efficient Algorithm For a Lattice Basis Reduction”, *volume 9 of Journal of Algorithms*, pp. 47-62.

Shamir, A., 1982. “A Polynomial Time Algorithm For Breaking the Basic Merkle-Hellman Cryptosystem”, *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Scienc*e, pp. 145-152, IEEE.

Stern, J., 1987. “Secret Linear Congruential Generators are not Cryptographically Secure”, *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pp. 421-426, IEEE.

Stern, J. and Toffin, P., 1990. “Cryptanalysis of a public-key cryptosystem based on approximations by rational numbers”, *Proceedings of Eurocrypt'90, volume 473 of LNCS*, pp. 313-317, Springer-Verlag.

WEB_1, 2006. Hoffstein, J., Lieman, D., Pipher, J. and Silverman, J.H., 1999. "NTRU: A Public Key Cryptosystem", 15/04/2006. http://grouper.ieee.org/groups/1363/lattPK/submissions.html#NTRU1

WEB_2, 2006. Hoffstein, J. and Silverman, J.H., 2000. "Protecting NTRU Against Chosen Ciphertext and Reaction Attacks", NTRU Cryptosystems Technical Report 016, Version 1, 15/04/2006. http://www.ntru.com/cryptolab/tech_notes.htm#016

WEB_3, 2006. Hoffstein, J. and Silverman, J., 2000. "Optimizations for NTRU", NTRU CryptoLab Articles, 15/04/2006. http://www.ntru.com/cryptolab/articles.htm

WEB_4, 2006. Hoffstein, J., Silverman, J., H. and Whyte, W., 2003. "Estimated Breaking Times for NTRU Lattices", NTRU Cryptosystems Technical Report, No. 12, Version 2, 15/04/2006. http://www.ntru.com/cryptolab/tech_notes.htm#012

WEB_5, 2006. Howgrave-Graham, N., Silverman, J.H., Singer, A., Whyte and W., 2003. "Provable Security in the Presence of Decryption Failures", 15/04/2006. http://www.ntru.com/cryptolab/articles.htm#2003_3

WEB_6, 2006. Howgrave-Graham, N., Silverman, J.H. and Whyte, W., 2005. "Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3", Cryptology ePrint Archive, Report 2005/045, 15/04/2006. http://eprint.iacr.org

WEB_7, 2007. IEEE P1363.1 Draft Download "Draft Standard for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices", 10/01/2007. http://grouper.ieee.org/groups/1363/lattPK/draft.html

WEB_8, 2006. "Fips 186-2 Digital Signature Standard" 15/05/2006. http://www.itl.nist.gov/fipspubs/fip186.htm

WEB_9, 2007. "The Case For Elliptic Curve Cryptography", 15/01/2007. http://www.nsa.gov/ia/industry/crypto_elliptic_curve.cfm.

WEB_10, 2006: “Recommended Elliptic Curves Domain Parameters”, 30/08/2006. http://www.secg.org/index.php?action=secg,docs_secg

WEB_11, 2006. Silverman, J.H., 2000. “Plaintext Awareness and the NTRU PKCS”, NTRU Cryptosystems Technical Report 007, Version 2, 15/04/2006. http://www.ntru.com/cryptolab/tech_notes.htm#007

WEB_12, 2006. Silverman, J.H., 2001. “Wraps, Gaps and Lattice Constants”, NTRU Cryptosystems Technical Report 011, Version 2, 15/04/2006. http://www.ntru.com/cryptolab/tech_notes.htm#011

WEB_13, 2006. Silverman, J. and Whyte, W., 2003. “Estimating Decryption Failure Probabilities for NTRUEncrypt”, NTRU Cryptosystems Technical Report 018, Version 1, 15/04/2006. http://www.ntru.com/cryptolab/tech_notes.htm#018

WEB_14 2007. Silverman, J.H. and Whyte, W., 2006. “Timing Attacks on NTRUEncrypt via Variation in the Number of Hash Calls”, NTRU Cryptosystems Technical Report 021, 15/01/2007. http://grouper.ieee.org/groups/1363/lattPK/submissions.html#2006-06

Yu, W. and He, D., 2005. “Study on NTRU Decryption Failures”, *Proceedings of ICITA '05: the Third International Conference on Information Technology and Applications Volume 2,Washington, DC, USA*, pp. 454-459, IEEE Computer Society.