



Republic of Iraq
Ministry of Higher Education and Scientific
Research
University of Diyala
College of Science
Computer Science Department



"NTRU Modification Against LLL Attack"

*A Thesis Submitted to
the University of Diyala / College of Science / Department of Computer Science, In
partial Fulfillment of the Requirements for the Degree of Master in Computer Science*

By
Omar Sapti Guma'a

Supervised By

Prof. Dr.
Ziyad Tariq Mustafa
Al-Ta'i

Prof. Dr.
Qasim Mohammed Hussein
Al-Shamry

December / 2020 A.D.

Diyala

Rabi Al-Akhar / 1441 A.H

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

"وَقُلْ رَبِّ أَدْخِلْنِي مُدْخَلَ صِدْقٍ وَأَخْرِجْنِي مُخْرَجَ
صِدْقٍ وَاجْعَلْ لِي مِنْ لَدُنْكَ سُلْطَانًا نَصِيرًا"

صَدَقَ اللَّهُ الْعَظِيمُ

DEDICATED TO

To the martyrs of Iraq ...

To my father ...

To my mother ...

To my wife ...

To my brothers ...

To my sisters ...

Omar

ACKNOWLEDGMENTS

*Above all else, I want to express my great thanks to my God (Allah) for his uncountable gifts and for helping me to present this work. Furthermore, I would like to express my deepest gratitude to my supervisors (**Prof. Qasim Mohammed Hussein**) and (**Prof. Ziyad Tariq Mustafa Al-Ta'i**) for their powerful guidance, motivation, valuable suggestions, support and attention throughout this research.*

Special thanks to the Faculty of Science at the University of Diyala / Postgraduate to give me the opportunity to join the Master's in Computer Science.

I would like to say "thank you" to my faithful friends for supporting and giving me advice

Finally, I am most grateful to my parents who provided me with endless love and support morally and materistically during all my years of studying. As I would like to mention that without my wife insistence, I might not be pursuing my study. Words are simply not enough to express my gratitude to My father who always believed in me, and recovered me when I find myself in the gloom. Many thanks to him.

Thanks for All

Omar

ABSTRACT

The existence of quantum computers makes the execution of algorithms that requiring high computational very possible, which makes many current cryptosystems broken with the execute of these algorithms. Peter Shor developed a quantum algorithm called Shor's algorithm in 1994, and considered this an important discovery in this area. Therefore, breaking many existing cryptosystems such as RSA and ECC is possible with a quantum computer.

Consequently, there is needs to new cryptographic schemes that be resistant to quantum. These new cryptographic constructions should be efficient and secure enough to be used in practice and standardized for a large number of years, replacing the current ones if needed.

Lattice-based public-key cryptosystem (LB-PKC) consider as one presented solutions to overcome this challenge. NTRU is one of a LB-PKCs that based on truncated polynomial ring $\mathbb{Z}[x]/(x^N - 1)$. But, NTRU system has been attacked by using the LLL algorithm under certain conditions, which can find the secret key when the length of the public key is less than 127.

The aim of the thesis is to make modifications on the original NTRU cryptosystem to ensure the failing the LLL attack on it. This thesis can be divided in two parts: first, presents a method to add new parameter to the parameters of the original NTRU, the values of this parameter depends on the linear feedback shift registers (LFSR). Secondly, presents a modification on the original NTRU by doing swapping operation between the values of public key. in addition to generate a long keys sequences dynamically to use in encryption. Experimental results on the proposed methods have demonstrated the ability of these methods to resist against the LLL algorithm attacking, even if the public key length does not exceed 11, also these proposals could generate approximately $(Np-1)$ versions of key sequence with length N .

Table of Contents

<u><i>Title</i></u>	<u><i>Page No.</i></u>
Acknowledgement.....	i
Abstract.....	ii
Table of Contents	iii
List of Figures.....	vi
List of Tables	vi
List of Algorithms	vii
List of Abbreviations	viii

Chapter One: Introduction

1.1 Overview.....	1
1.2 Related Work	2
1.3 Problem Statement	4
1.4 Aim of Thesis.....	5
1.5 Thesis Outline	5

Chapter Two: Theoretical Background

2.1 Introduction	7
2.2 Lattice-based Cryptography (LBC)	7
2.3 Mathematical and Computational background	8
2.3.1 Linear Feedback Shift Register (LFSR).....	8
2.3.2 Truncated polynomial ring.....	10
2.3.3 Finding the greatest common divisor	13
2.3.4 Extended Euclidean algorithm.....	13

2.3.5 Finding the polynomial multiplicative inverse.....	15
2.3.6 Convolution multiplication of two polynomials.....	16
2.3.7 Inverse of truncated polynomial rings.....	18
2.3.8 Lattices.....	21
2.4 NTRU Public Key Cryptosystem.....	20
2.4.1 NTRU public parameters.....	22
2.4.2 Key generation.....	23
2.4.3 Encryption Algorithm.....	24
2.4.4 Decryption Algorithm	25
2.5 NTRU Parameter Selection.....	26
2.5.1 Small polynomial.....	27
2.5.2 Choice NTRU parameter.....	27
2.6 Low hamming weight polynomial.....	28
2.7 The advantages of NTRU.....	30
2.8 Encryption and Decryption Speeds.....	31
2.9 Attack on NTRU cryptosystem.....	32
2.10 Lattice-based attack.....	32
2.10.1 The LLL algorithm.....	32
2.10.2 Lattice-based attack on the public key.....	34
2.10.3 Lattice-based attack on the cipher text.....	35

Chapter Three: Design of The Proposed System

3.1 Introduction.....	37
3.2 Objectives.....	38
3.3 The Proposal One.....	39
3.3.1 Key Generation.....	39
3.3.2 Encryption.....	41
3.4 The Proposal Two.....	42
3.4.1 Key Generation.....	44

3.4.2 Encryption.....	45
-----------------------	----

Chapter Four: Results and Evaluation

4.1 Introduction.....	47
4.2 Initialization.....	47
4.3 LLL Algorithm Attack.....	48
4.3.1 LLL Algorithm Attack on the public Key.....	48
4.3.1.1 The attack on the original NTRU.....	49
4.3.1.2 The attack on the proposal one.....	52
4.3.1.2 The attack on the proposal two.....	55
4.3.2 LLL Algorithm Attack on the Ciphertext.....	57
4.3.2.1 The attack on the original NTRU.....	58
4.3.2.2 The attack on the proposal one.....	60
4.3.2.3 The attack on the proposal two.....	62
4.4 Dynamically Generation of Keys Sequences.....	63
4.5 Performance Analysis.....	65
4.5.1 Encryption Time and Decryption Time Analysis.....	66

Chapter Five: Conclusions and Suggestions for Future Works

5.1 Introduction.....	68
5.2 Conclusions	68
5.2 Suggestions for Future Works	69
<i>Publications</i>	70
<i>References</i>	70

List of Figures

<u><i>Figure</i></u>	<u><i>Page No.</i></u>
2.1 Block diagram of LFSR	8
2.2 Block diagram of LFSR example with 5-stage	9
2.3 Encryption and Decryption Speeds of NTRU	31
3.1 Flowchart of dynamic key generation	43
4.1 Performance timings of NTRU and the proposed algorithms	67

List of Tables

<u><i>Table</i></u>	<u><i>Page No.</i></u>
1.1 A review of some proposed modifications for NTRU	2
2.1 The contents of LFSR after shifting process	20
2.2 NTRU Parameters and Keys.....	22
2.3 The parameters sets	28
2.4 Encryption and Decryption of NTRU	31
4.1 The lattice basis L of h (NTRU cryptosystem).....	49
4.2 The LLL algorithm results (NTRU cryptosystem)	50
4.3 The lattice basis L for h (proposal one).....	53
4.4 The LLL algorithm results (proposal one).....	54
4.5 The lattice basis of L for h (proposal two).....	55
4.6 The LLL algorithm results (proposal two).....	56
4.7 The lattice basis of L for the original NTRU	58
4.8 The LLL algorithm outputs for the original NTRU	59
4.9 The lattice basis of L (proposal one).....	60
4.10 The LLL algorithm outputs	61

4.11 The lattice basis of L (proposal two).....	62
4.12 The LLL algorithm outputs.....	63
4.13 Performance analysis for basic NTRU and the proposals.....	65
4.14 Encryption Time and Decryption Time Analysis for original NTRU and the proposed algorithms per second	66

List of Algorithms

<u>Table</u>	<u>Page No.</u>
2.1 GCG Algorithm.....	13
2.2 The multiplicative inverse of a mod b (a^{-1}_b)	14
2.3 The extended Euclidean algorithm to find polynomial inverse.....	16
2.4 Convolution multiplication of two polynomials with degree(N-1).....	17
2.5 Compute the inverse of polynomial in $(\mathbb{Z}/2\mathbb{Z})(X)/(X^N-1)$	18
2.6 Compute the inverse of polynomial in $(\mathbb{Z}/3\mathbb{Z})(X)/(X^N-1)$	19
2.7 Compute the inverse of polynomial in $(\mathbb{Z}/p\mathbb{Z})(X)/(X^N-1)$	20
2.8 Gram-Schmidt process	34
2.9 LLL Algorithm.....	34
3.1 Public key Generation (proposal one).....	40
3.2 Encryption process (proposal one).....	41
3.3 Public key Generation (proposal two).....	44
3.4 Encryption process (proposal two).....	45

List of Abbreviations

<u><i>Abbreviation</i></u>	<u><i>Meaning</i></u>
AES	Advanced Encryption Standard
CP	Cipher text Policy
CVP	Closest Vector Problem
DES	Data Encryption Standard
ECC	Elliptic Curve Cryptosystem
FFT	Fast Fourier Transform
gcd	Greater Common Divisor
KP-ABE	Key Policy - Attribute-Based Encryption
LB-PKC	Lattice-Based Public Key Cryptography
LFSR	Linear Feedback Shift Register
LLL	Lenstra - Lenstra –Lovasz
LWE	Learning with errors
NTRU	Nth Degree Truncated Polynomial Ring Unit
RFID	Radio-frequency identification
RSA	Rivest, Shamir, and Adelman
SVP	Shortest Vector Problem

Chapter One

Introduction

Chapter One

Introduction

1.1 Overview

The great development in the field of the Internet and technology, in the modern world today and the increase in the number of devices that send and receive data, in addition to the fact that there are many of these devices that perform several functions without human intervention, it led to a large data transfer process. Certainly, these data contains sensitive data and personal information, so there is an urgent need to protect this data from attacks. There are many ways to protect information. This can be overcome by exploiting public-key cryptography (PKC) which has features such as confidentiality, data integrity, authentication, and non-repudiation. With these features, PKC can provide security for communication networks, especially for ensuring the privacy and confidentiality of important information. There are many PKC used at the present time. RSA and elliptic curve cryptosystem (ECC) are two currently popular public key systems in modern cryptography. These cryptosystems are based on mathematical problems named hard problems. These cryptosystems are considered to be somewhat safe nowadays due to the lack of effective algorithms able to resolve this type of cryptosystems. This situation does not last long with the progress in the development that we are witnessing and the emergence of quantum computing technology, because the computers that depend on this concept differ greatly from computers that depend on electronics. Quantum computers are greatly able to solve many hard mathematical problems[1][2]. Hence, more effective methods are needed to meet this

challenge. Lattice-based constructions are currently important candidates for post-quantum cryptography. Lattice-based public-key cryptosystem (LB-PKC) has many advantages compared with ‘traditional’ cryptography. NTRU is one of a LB-PKC that based on truncated polynomial ring $\mathbb{Z}[x]/(x^N - 1)$, it has good features, which it makes to be an effective alternative to cryptosystems such as RSA and ECC [3].

This system has weak points, including the ability to attack it under certain condition using Lenstra–Lenstra–Lovász lattice basis reduction algorithm [4] discover the private keys or other private key called an alternative key that can be used to decrypt the cipher text, as well as it can be used to discover the plaintext by cipher text and public parameters[5]. To avoid the LLL algorithm attack, this thesis presents two proposals that suggest modifications on NTRU cryptosystem.

1.2 Related Work

There are many proposed cryptosystems that suggest modifications on NTRU to improve it. Table (1.2) illustrates that:

Table (1.1): A review of some proposed modifications for NTRU

Publishing, Year	Description
[6,2010]	<ul style="list-style-type: none"> • Presents cryptosystem that depends on NTRU structure called OTRU. • OTRU has been designed based on the NTRU core and exhibits high levels of parallelism with full operand length.

	<ul style="list-style-type: none"> • OTRU is the first step in the design of the public key cryptosystems with a non-associative algebra.
[7,2013]	<ul style="list-style-type: none"> • Proposed a new cryptosystem that based on NTRU structure called ETRU, and the ring of this cryptosystem is $Z[w]$ and the coefficients are integer numbers. • The LLL algorithm can be used to find short vector in this work because there is a relatively short vector.
[8,2015]	<ul style="list-style-type: none"> • It is based on ideal lattice which is special structured lattices. This scheme was motivated by ETRU[7], which is used to make this scheme provably secure.
[9,2015]	<ul style="list-style-type: none"> • Introduced CQTRU cryptosystem based on commutative quaternions algebra. • The resistance of CQTRU to lattice attack is at least four times better than that of NTRU at the same dimension and also in CQTRU even with the use of LLL reduce algorithm an attacker cannot always guarantee to find the private key.
[10,2016]	<ul style="list-style-type: none"> • The role played by Z in NTRU replaced by the ring $Q[\alpha]$ of polynomial in one variable α over the Rational Field. The same value of N, CTRU is faster than NTRU, but not always. • This proposed scheme may be secure against the LLL algorithm attack but not proved.
[11,2018]	<ul style="list-style-type: none"> • A general framework for NTRU is considered, and a new PKC called D-NTRU is proposed. • It is shown that the D-NTRU cryptosystem reduces the

	<p>ciphertext expansion of the NTRU algorithm, and the encryption and decryption algorithms of D-NTRU perform even asymptotically faster than the NTRU algorithm only at the cost of slightly enlarged secret and public keys.</p> <ul style="list-style-type: none"> • The IND-CPA security of D-NTRU was proven under the NTRU one-wayness hardness assumption.
[12,2019]	<ul style="list-style-type: none"> • Proposed GTRU cryptosystem that based on group based of NTRU algorithm. • The parameters (f and g) must be choosing as $n \times n$ matrices instead of small polynomials. • NTRU key size is much smaller than GTRU key size that constructed by G_n. • This work is slower than NTRU, but it is more secure against lattice-based attacks.
[13,2020]	<ul style="list-style-type: none"> • In this work, RCPKC is proposed, a secure and effective congruential, modulo q, public-key cryptosystem using big numbers. • It uses the same encryption/decryption mechanism as NTRU does but works with numbers.

1.3 Problem Statement

Many public key cryptosystems, such as RSA, ECC and Diffie-Hellman protocol, are based on hardness problem (factorization or the discrete logarithm problem). But, these cryptosystems can breaking by use efficient algorithm for quantum computers [14][15]. Therefore, we

need other ways to overcome this challenge. Lattice-based cryptography is NP-hard problem, post-quantum and it is considered one of the solutions presented because there is still no algorithm that can break these problems. NTRU is a lattice-based public key cryptosystem, but this system can be attacked by LLL algorithm under certain conditions. Therefore, the basic research problem is to find suitable solutions to prevent the LLL algorithm from breaking this system.

1.4 Aim of Thesis

The aim of the thesis is to make modifications on encryption algorithm of the NTRU public key cryptosystem, to ensure that the LLL algorithm cannot able to break this system.

Another aim is to find a method to generate long sequences of keys dynamically, without need addition private keys to ensure that the key is not repeated when encrypted data size is large, also and in order to increase encryption security strength.

1.5 Thesis Outline

This thesis is structured around Five chapters, including chapter one, it contains the following chapters:

- **Chapter Two: (Theoretical Background)**

This chapter explains in detail background of IoT, NTRU public key cryptosystem, as well as LLL algorithm.

- **Chapter Three :(Design of The Proposed System)**

In this chapter, the proposed algorithm design and the implementation steps are given.

- **Chapter Four: (Results and Evaluation)**

This chapter is dedicated to showing the outputs and tests of the proposed system.

- **Chapter Five: (Conclusions and Recommendations for Future Works)** some concluding remarks which are derived from the outputs of the conducted tests are given in this chapter; also some suggestions for future work are presented.

Chapter Two

Theoretical Background

Chapter Two

Theoretical Background

2.1 Introduction

This chapter provides a brief description NTRU public key cryptosystem with its mathematical background and algorithms. In addition to an explanation of Linear Feedback Shift Registers (LFSR) and how it works with simple example that it used in the proposed method. Finally, an explanation of the method of attack NTRU by using the LLL algorithm on the public key and the cipher text.

2.2 Lattice-Based Cryptography (LBC)

Miklós Ajtai, in 1996 [16] presented cryptographic based on lattice and it considers the first lattice-based cryptographic, the security in this method depends on lattice problems classified as hardness problems.

Lattice-based cryptography (LBC) is a common technique that is more secure against several attacks. It cannot be deciphered in polynomial time even by using a computer with high capabilities and has been proven resilient against various known traditional cyber-attacks [17]. Therefore, many attacks can handle effectively by using LBC.

LBC provides strong security and high efficiency make it highly suitable for IoT applications data and suitable can accommodate further advances of smart IoT services, in addition LBC obtains the most intensive attention among all subfields of cryptography [3].

NTRU is a public key cryptosystem and classifies as the fastest known cryptosystems and consider a lattice-based cryptography (LBC) as well as it considered as efficient alternative to ECC and RSA [18]. The performance of NTRU is efficient and suitable for the most constraints devices (as a smartcards and RFID (Radio-Frequency Identification)) as well as, NTRU needs small average power [19].

2.3 Mathematical and Computational Background

This section shows an overview of some algebraic background that is useful for thesis work.

2.3.1 Linear Feedback Shift Register (LFSR)

A linear feedback shift register (LFSR) has L stages register and it has a linear combined feedback function to produce a sequence of bits. XOR operations are common used to tap some stages from the register. The sequence must have maximum period, high linear complexity and good random statistical properties. The block diagram of LFSR is shown in figure 2.2.

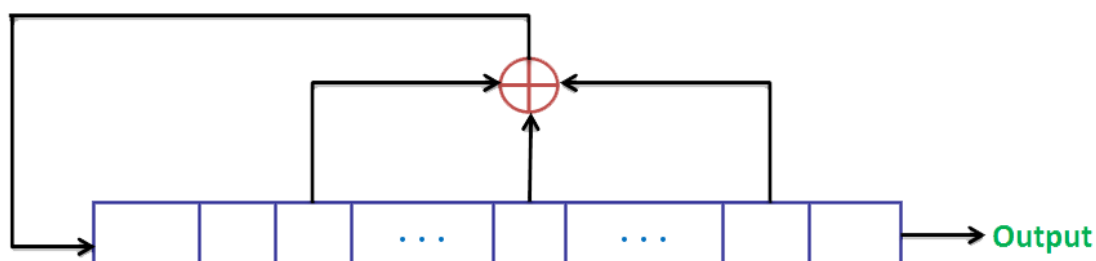


Figure (2.1): Block diagram of LFSR

The stream of values produced by the register is completely determined by its current (or previous) state. Initially, the first value of the LFSR is called the seed and because the operation of the register is deterministic. The

complexity max period is $(2^L - 1)$ where L is the length of shift register[20][21].

For example if the 5-stage LFSR with taps at positions 5 and 3[31], and it contains the sequence (1 0 0 1 0):

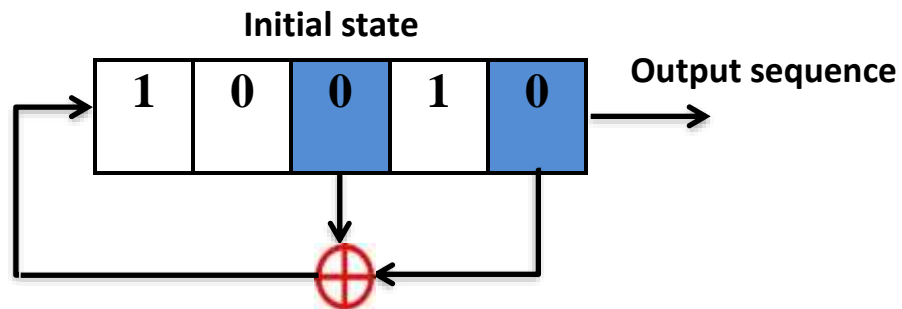


Figure (2.2): Block diagram of LFSR example with 5-stage

If shifting it with ($N=5$) the content becomes as follow:

Table (2.1): The contents of LFSR after shifting process

clock	0	1	2	3	4	
--	1	0	0	1	0	← initial values
1	0	1	0	0	1	
2	1	0	1	0	0	
3	1	1	0	1	0	
4	0	1	1	0	1	
5	0	0	1	1	0	

2.3.2 Truncated Polynomial Ring

A polynomial ring is a ring formed by the set of polynomials with coefficients in a ring. In this section we describe polynomials and polynomial rings in order to introduce the truncated polynomial rings.

A. Polynomials

A polynomial in X with coefficients in a field K is an expression of the form:

$$F(X) = a_0 + a_1X + \cdots + a_{m-1}X^{m-1} + a_mX^m \quad \dots (2.1)$$

Where a_0, \dots, a_m , the coefficients of $F(X)$, are elements of K and X, X^2, \dots, X^m are formal symbols (“the powers of X ”). Such expressions can be added and multiplied, and then brought into the same form using the ordinary rules for manipulating algebraic expressions, such as associativity, commutativity, distributivity, or take common factors. Any term a_kX^k with zero coefficient, $a_k = 0$, may be omitted. Using the summation symbol the same polynomial can be expressed more compactly as follows:

$$F(X) = \sum_{k=0}^m a_kX^k \quad \dots (2.2)$$

It is understood that the number of terms is finite, i.e. a_k is zero for all enough large values of k , in our case, for $k > m$. The degree of a polynomial is the largest k such that the coefficient a_k is not zero.

B. Polynomial Rings

Polynomials rings are essential in the NTRU public-key algorithm in order to generate random polynomials. A polynomial ring is defined by a ring which contains the values the coefficients can obtain and a delimiter or maximum degree when polynomials over one variable are represented. A polynomial ring $R[X]$ over the ring R in one variable X is formed by the set of all polynomials with coefficients in R . The elements of $R[X]$ are the polynomials with the form:

$$F(X) = \sum_{i=0}^n a_i X^i = a_0 + a_1 X + a_2 X^2 + \dots + a_n X^n, \quad \dots (2.3)$$

where $a_i \in R$ and $0 \leq i \leq n$. The symbol X is commonly called the variable, and the ring $R[X]$ is also called the ring of polynomials in one variable over R , to distinguish it from more general rings of polynomials in several variables. In general, X and its powers X_i are treated as formal symbols, not as elements of the field R . In order for $R[X]$ to form a ring, all powers of X have to be included, and this leads to the definition of polynomials as linear combinations of the powers of X , with coefficients in R for the ring $R[X]$ [22].

NTRU cryptosystem works in the ring $R = Z[x]/(X^N - 1)$. The degree of these polynomials is $N-1$ and its coefficients are integer numbers:

$$a = a_0 + a_1 X + a_2 X^2 + a_{N-1} X^{N-1} \quad \dots (2.4)$$

The coefficients of this may be 0.

Many operations can apply with the polynomials in R such as adding and multiplying operations, the adding operation can be represented such as the following:

$$a + b = (a_0 + b_0) + (a_1 + b_1)X + \dots + (a_{N-1} + b_{N-1})X^{N-1} \quad \dots (2.5)$$

and the multiplying:

$$a * b = c_0 + c_1X + \dots + c_{N-1}X^{N-1} \quad \dots (2.6)$$

after performing the multiplication operation, the X that have N degree must be substituted by 1 degree, and the X^{N+1} must be substituted by X , and so on [23] [24].

$$R = Z[X]/X^{N-1} \quad \dots (2.7)$$

In that equation R is the name of the ring, $Z[x]$ indicates that the coefficients of X are integer numbers and X^{N-1} indicates the greatest possible degree of the polynomial.

The value of N (the maximal degree of the polynomials in the ring $+1$) is one of the three components that determine the particular NTRU cryptosystem (in contrast with the other NTRU cryptosystems in the family). Some particular rings (and thus N -s) are better for the usage of NTRU than others, but generally the rule is the bigger N , the bigger the set of polynomials, and thus the more secure the system (and also a bit slower).

The other two components determining the NTRU cryptosystem are two co-prime (having no common divisor different from 1) numbers, usually named p and q . Usually p is a very small integer number, almost always 3 is used, while q is usually some power of the number 2 (for example 128 or 256). Since 3 and any number that is a power of 2 have no common divisor, they are co-prime and we can use them for our NTRU encryption [22].

2.3.3 Finding the Greatest Common Divisor(GCD)

The Euclidean algorithm works with the equation $\gcd(a, b) = \gcd(b, a \bmod b)$ (where a and b are positive integers), assume $a > b > 0$. This algorithm is shown in algorithm (2.1) below [23]:

Algorithm (2.1): GCD Algorithm

Input : a (Nonnegative integer), and b (positive integer)
Output : The greatest common divisor, A
Step 1: $A = a; B = b$ Step 2: if $B = 0$ return $A = \gcd(a, b)$ Step 3: $R = A \bmod B$ Step 4: $A = B$ Step 5: $B = R$ Step 6: goto Step 2

To compute greatest common divisor of two polynomials must extend the Euclidean algorithm for the coefficients of a field. The algorithm (2.1) above also can be used to find the (GCD) of two polynomials (where a and b are polynomials, the degree of polynomial b lesser than a degree).

2.3.4 Extended Euclidean algorithm

A modular multiplicative inverse of a modulo m can be found by using the extended Euclidean algorithm. The Euclidean algorithm determines the greatest common divisor (gcd) of two integers, say a and m . If a has a multiplicative inverse modulo m , this gcd must be 1. The last of several equations produced by the algorithm may be solved

for this gcd. Then, using a method called "back substitution", an expression connecting the original parameters and this gcd can be obtained. In other words, integers x and y can be found to satisfy

$$ax + by = \gcd(a, b) = 1,$$

Rewritten, this is

$$ax - 1 = (-y)b,$$

that is,

$$ax \equiv 1 \pmod{b},$$

So, a modular multiplicative inverse of a has been calculated. A more efficient version of the algorithm is the extended Euclidean algorithm, which, by using auxiliary equations, reduces two passes through the algorithm (back substitution can be thought of as passing through the algorithm in reverse) to just one [25].

Algorithm (2.2) The multiplicative inverse of a mod b (a^{-1}_b)

Input : Two non-negative integers a and b with $a \geq b$
Output : The multiplicative inverse of a mod b
Step 1: $x=0$; Step 2: do Step 3: $x = x + 1$ Step 4: $y = a * x$ Step 5: $y = y \bmod b$ Step 6: if $y = 1$ then Step 7: return x Step 8: while $x < b$

Example 2.1:

Assume we have $p=3$ and $q=32$, to compute a multiplicative inverse of 3 (mod 32) follow the following steps:

$$3x = 1 - 32y$$

$$ax \equiv 1 \pmod{b}, \quad 3x \equiv 1 \pmod{32}$$

By using extended Euclidean algorithm:

$$32 = 10 * 3 + 2$$

$$3 = 1 * 2 + 1$$

Backwards substitution for two above steps:

$$2 = 32 - 10 * 3$$

$$1 = 3 - 1 * 2$$

$$= 3 - 1 * (32 - 10 * 3)$$

$$= 3 - 32 + 10 * 3$$

$$= 11 * 3 - 32$$

$$11 * 3 = 1 + 32 \equiv 1 \pmod{32}$$

Satisfying to $3x \equiv 1 \pmod{32}$:

$$x = 11 \text{ is represents } p^{-1}_q.$$

2.3.5 Finding the Polynomial Multiplicative Inverse

By using the extended Euclidean algorithm can compute the multiplicative inverse of the polynomial in \mathbb{Z}_p . The degree of $m(x)$ is larger

than the degree of $b(x)$ where $\gcd[m(x), b(x)] = 1$ then the multiplicative inverse of $b(x)$ modulo $m(x)$ computed by using the extended Euclidean algorithm. Therefore, when $\gcd[m(x), b(x)] = 1$, the polynomial multiplicative inverse algorithm is clarified in algorithm(2.3)[26].

Algorithm (2.3) The extended Euclidean algorithm to find polynomial inverse

Input : The polynomials $m(x)$ and $b(x)$
Output : $B2(x)$ (The multiplicative inverse)
<p>Step 1: $[A1(x), A2(x), A3(x)] \leftarrow [1, 0, m(x)];$ $[B1(x), B2(x), B3(x)] \leftarrow [0, 1, b(x)]$</p> <p>Step 2: if $B3(x) = 0$ return $A3(x) = \gcd[m(x), b(x)];$ no inverse</p> <p>Step 3: if $B3(x) = 1$ return $B3(x) = \gcd[m(x), b(x)];$ $B2(x) = b(x) \bmod m(x)$</p> <p>Step 4: $Q(x) = \text{quotient of } A3(x)/B3(x)$</p> <p>Step 5: $[T1(x), T2(x), T3(x)] \leftarrow [A1(x) - Q(x)B1(x), A2(x) - Q(x)B2(x),$ $A3(x) - Q(x)B3(x)]$</p> <p>Step 6: $[A1(x), A2(x), A3(x)] \leftarrow [B1(x), B2(x), B3(x)]$</p> <p>Step 7: $[B1(x), B2(x), B3(x)] \leftarrow [T1(x), T2(x), T3(x)]$</p> <p>Step 8: go to 2</p>

2.3.6 Convolution multiplication of two polynomials

with degree (N-1)

The algorithm (2.4) below represents as a function to compute the convolution multiplication of two polynomials with degree (N-1). The

variable M (to perform the modular process) is either p or q in NTRU cryptosystem depending upon which one is passed into the function[27].

Algorithm (2.4) Convolution multiplication of two polynomials

with degree $(N-1)$

Input: The parameter N , the modulus M , the two polynomials to be multiplied, a and b .
Output: The product polynomial, c .
Step 1: for $k = N - 1$ downto 0 do Step 2: $c[k] = 0$ Step 3: $j = k + 1$ Step 4: for $i = N - 1$ downto 0 do Step 5: if $j = N$ then Step 6: $j = 0$ Step 7: end if Step 8: if $a[i] \neq 0$ and $b[j] \neq 0$ then Step 9: $c[k] = c[k] + (a[i] \cdot b[j]) \bmod M$ Step 10: end if Step 11: $j = j + 1$ Step 12: end for Step 13: end for

2.3.7 Inverse of Truncated Polynomial Rings

The property $a * A = 1 \pmod{q}$, the polynomial A is the inverse polynomial modulo q . But, an inverse modulo q cannot apply for every polynomial. Therefore, if the inverse of polynomial modulo q exists then it is easy to compute it.

The truncated polynomials ring modulo q in the ring $R_q = (\mathbb{Z}/\mathbb{Z}q)[x]/(x^N - 1)$ where the polynomial that a random chosen $a(x)$ is invertible for the ring R_q ($a(1)=1$) [28].

A. Algorithm to Find Inversion in $(\mathbb{Z}/2\mathbb{Z})/(X^N-1)$

The algorithm below to compute the inverse of the polynomial in the ring $(\mathbb{Z}/2\mathbb{Z})(x)/(m(x))$, it is providing $\gcd(a(x), m(x)) = 1$ and $m(0)=1$ [28].

Algorithm (2.5): Compute the inverse of polynomial in

$$(\mathbb{Z}/2\mathbb{Z})(X)/(X^N - 1)$$

Input : $a(x)$
Output : $b(x) = a(x)^{-1}$ in $(\mathbb{Z}/2\mathbb{Z})(X)/(X^N - 1)$
Step 1: Initially: $k = 0, b(x) = 1, c(x) = 0, f(x) = a(x), g(x) = x^N - 1$ Step 2: while $f_0 = 0$ do Step 3: $f(x) = f(x)/x, c(x) = c(x) * x, k = k + 1$ Step 4: if $f(x) = 1$ then return $x^{N-k}b(x) \pmod{x^N - 1}$ Step 5: if $\deg(f) < \deg(g)$ then Step 6: exchange b and c and exchange f and g Step 7: $f(x) = f(x) + g(x) \pmod{2}$ Step 8: $b(x) = f(x) + c(x) \pmod{2}$ Step 9: go to loop.

B. Algorithm to Find Inversion in $(\mathbb{Z}/3\mathbb{Z})/(X^N-1)$

To generate The NTRU keys (public and private), it needs to compute the inverse of the polynomial (modulo p). For the prime $p=3$, an algorithm below illustrates the steps for finding the inverse of a polynomial modulo p .

Algorithm (2.6): Compute the inverse of polynomial in

$$(\mathbb{Z}/3\mathbb{Z}) (x) / (x^N - 1)$$

Input : $a(x)$
Output : $b(x) = a(x)^{-1}$ in $(\mathbb{Z}/3\mathbb{Z}) (x) / (x^N - 1)$
Step 1: Initially: $k = 0, b(x) = 1, c(x) = 0, f(x) = a(x), g(x) = x^N - 1$. Step 2: loop Step 3: while $f_0 = 0$ do Step 4: $f(x) = f(x)/x, c(x) = c(x) * x, k = k + 1$ Step 5: if $f(x) = \pm 1$ then return $\pm x^{N-k} b(x) \pmod{x^N - 1}$ Step 6: if $\deg(f) < \deg(g)$ then Step 7: exchange b and c and exchange f and g Step 8: if $f_0 = g_0$ then Step 9: $f(x) = f(x) - g(x) \pmod{3}$ Step 10: $b(x) = b(x) - c(x) \pmod{3}$ Step 11: else Step 12: $f(x) = f(x) + g(x) \pmod{3}$ Step 13: $b(x) = b(x) + c(x) \pmod{3}$ Step 14: goto Step 2

Generally, every computation in the above algorithm is done modulo 3, this means the coefficients will be in the set $\{-1,0,1\}$ only.

C. Algorithm to Inverse in $(\mathbb{Z}/p\mathbb{Z})/(X^N-1)$

Also to generate the NTRU keys pairs (public/private), it needs to compute the inverse of the polynomial modulo for any integer value, but it must be a prime power (a power of 2 particularly).

So, there are a simple method to determine an inverse modulo a prime p . this method based on Newton iteration to increase the computations speed for the inverse modulo p^r . Therefore, this algorithm will converged quadratically, with $\log_2(r)$ [34].

Algorithm (2.7): Compute the inverse of polynomial in $(\mathbb{Z}/p\mathbb{Z}) (X) / (X^N - 1)$

Input: $a(x)$, p (a prime), r
Output: $B(x)=a(x)^{-1}$ in $(\mathbb{Z}/p\mathbb{Z})(x) / (x^N-1)$
Step 1: Initialize $k = 0$, $b(x) = 1$, $c(x) = 0$, $f(x) = a(x)$, $g(x) = x^N - 1$ Step 2: loop Step 3: while $f_0 = 0$ do Step 4: $f(x) = f(x)/x$, $c(x) = c(x) * x$, $k = k + 1$ Step 5: if $\deg(f) = 0$ then Step 6: $b(x) := f_0^{-1}b(x) \pmod{p}$ Step 7: return $x^{N-K}b(x) \pmod{X^N - 1}$ Step 8: if $\deg(f) < \deg(g)$ then Step 9: exchange b and c and exchange f and g Step 10: $u = f_0g_0^{-1} \pmod{p}$ Step 11: $f(x) = f(x) - u * g(x) \pmod{p}$ Step 12: $b(x) = b(x) - u * c(x) \pmod{p}$ Step 13: goto Step 2

2.3.8 Lattices

Gauss, Lagrange, Korkine-Zolotareff, Hermite and others and to Minkowski's geometry of numbers was develops the theory of quadratic forms and that considers as a history of lattice reduction [29].

The concept of lattice is of great importance in terms of cryptography. In one hand, it can be used to break a cryptosystem and on the other hand, it is considered as a source of hard problems to present a highly secure cryptosystem [30].

In 1980 by an algorithm number theory, the two famous problems have occurred, they are the closest vector problem (CVP) and the shortest vector problem (SVP). Since that time, the lattice problem became important in the cryptography as a source of hard problems in order to design cryptosystem as well as using lattice to break cryptosystem [31] [32] [33].

A. Hard Problem in Lattice

Closest Vector Problem (CVP) and Short Vector Problem (SVP) are two important problems with a lattice, L . The (CVP) is the problem that used to compute the lattice point x^T the closest in norm to vector v (minimizing $||Ax - v||$) and the (SVP) is a problem that used to find the shortest non-zero vector in lattice with minimum norm [34] [35].

There is no polynomial time algorithm able to solve these problems although it has been good studied in both theoretically and experimentally[31].

2.4 NTRU Public Key Cryptosystem

At the Crypto '96 conference J. Hoffstein, J. Pipher and J. Silverman were presents a public key cryptosystem called NTRU. This cryptosystem based on in both polynomials multiplications and additions in truncated polynomials ring $Z[x]/(x^N - 1)$ and uses two integer parameters p and q for modulo reduction [31].

The NTRU cryptosystem computations are made in the ring $R = Z[x]/(x^N - 1)$. Where the degree of polynomials is $(N-1)$ and the coefficients of these polynomials are integer form:

$$a = \sum_{i=0}^{N-1} a_i x^i = [a_0, a_1, \dots, a_{N-1}] \quad \dots (2.8)$$

Those polynomials are reduced modulo $x^N - 1$, and then reducing the coefficients of each polynomial modulo p or modulo q , respectively.

2.4.1 NTRU Public Parameters

The parameters and keys of NTRU PKCS listed in table 2.2 [32] [36]

Table (2.2): NTRU Parameters and Keys

NTRU parameters	Explanation
N	The degree $N - 1$ for each polynomial in the truncated polynomial ring. (Non-secret)
q	The coefficients of the polynomials reduces modulo q . (Large modulo)
p	The coefficients of the polynomials in message reduces modulo p . (small modulo)

f	The private key (Secret polynomial)
g	The private key (Secret polynomial used to generate the public key h)
h	The public key (polynomial)
r	Random polynomial (Secret used to blind the message and discarded later)
d_f	The f coefficients that are equal to 1 is d_f , equal to -1 is (d_f-1) and the rest coefficients are 0's.
d_g	The g coefficients that are equal to 1 is d_g , equal to -1 is (d_g-1) and the rest coefficients are 0's.
d_r	The r coefficients that are equal to 1 is d_r , equal to -1 is (d_r-1) and the rest coefficients are 0's.

2.4.2 Key Generation

The key generation steps as the following:

1. The first step is selecting the polynomials (f and g) in the truncated polynomials ring R . These polynomials represent the private keys and f must have inverse modulo p and q .
2. Compute f_q and f_p (f_q, f_p : the inverse of polynomial $f \bmod q$ and p , respectively) and must be:

$$f * f_q = 1 \pmod{q} \text{ and } f * f_p = 1 \pmod{p}$$

Where $*$ denotes to convolution multiplication in the ring R .

3. Then compute:

$$h = p \cdot f_q * g \pmod{q} \quad \dots (2.9)$$

$$\text{where } h_k = \sum_{i=0}^k f_i g_{k-i} + \sum_{i=k+1}^{N-1} f_i g_{N+k-i}$$

and the polynomial, h , is the public key.

Example 2.2

To generate a key using the NTRU cryptosystem, the parameters (N, p, q)

Let $N = 11, p = 3$ and $q = 32$

The receiver must choose two polynomials f and g as private keys. Let the polynomial f has $d_f = 4$ and the polynomial g has $d_g = 3$.

$$f = -1 + X + X^2 - X^4 + X^6 + X^9 - X^{10}$$

$$g = -1 + X^2 + X^3 + X^5 - X^8 - X^{10}$$

Then, find f_p and f_q (according to algorithm (2.6) because $p=3$, So the calculate of the polynomial inverse of f (modulo p) and the polynomial inverse of f (modulo q) respectively in $(\mathbb{Z}/3\mathbb{Z})[x]/(x^N - 1)$):

$$f_p = 1 + 2X + 2X^3 + 2X^4 + X^5 + 2X^7 + X^8 + 2X^9$$

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

Finally, find the public key, h (equation 2.9):

$$h = 8 + 25X + 22X^2 + 20X^3 + 12X^4 + 24X^5 + 15X^6 + 19X^7 + \\ 12X^8 + 19X^9 + 16X^{10} \quad (\text{modulo } 32).$$

The pair f , and g represents the private keys

2.4.3 Encryption Algorithm

To encrypt the message, compute:

The message, m , must be converted into a polynomial, the coefficients should be reduced by modulo p .

1. Select r (small random polynomial) $r \in R$, for scrambling the message.
2. Compute the cipher text , e :

$$e = p.r * h + m(\text{modulo } q) \quad \dots (2.10)$$

The coefficients of r and m are reduced modulo p .

Example 2.3

This example shows NTRU encryption process. The message of the sender must be converting to the form of m :

$$m = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}$$

Suppose that the sender chooses randomly a blinding value r :

$$r = -1 + X^2 + X^3 + X^4 - X^5 - X^7$$

Then, compute the encrypted message (equation 2.10), as:

$$e = (14 + 11X + 26X^2 + 24X^3 + 14X^4 + 16X^5 + 30X^6 + 7X^7 + 25X^8 + 6X^9 + 19X^{10}), (\text{modulo } 32).$$

Where e is the cipher text.

2.4.4 Decryption Algorithm

The sender uses the following steps to decryption process:

1. Using the polynomial f , (private key) to compute

$$a = f * e (\text{modulo } q) \quad \dots (2.11)$$

The coefficients of a must be in $[-\frac{q}{2}, \frac{q}{2}]$ by shifting it.

2. Computing the polynomial $b = a (\text{modulo } p)$ \dots (2.12)
3. Finally using f_p (the private key) to find the plain text, m :

$$m = f_p * b \text{ (modulo } p) \quad \dots (2.13)$$

Example 2.4

The cipher text, e , which generated in encryption process:

$$e = 14 + 11X + 26X^2 + 24X^3 + 14X^4 + 16X^5 + 30X^6 + 7X^7 + 25X^8 + 6X^9 + 19X^{10}$$

The receiver uses the polynomial f and chooses values between $[-15, 16]$ to compute (equation 2.11)

$$a = 3 - 7X - 10X^2 - 11X^3 + 10X^4 + 7X^5 + 6X^6 + 7X^7 + 5X^8 - 3X^9 - 7X^{10}$$

And then, the receiver reduces the coefficients of a modulo 3 (equation 2.12) to compute b :

$$(b = -X - X^2 + X^3 + X^4 + X^5 + X^7 - X^8 - X^{10}), \text{ (modulo 3).}$$

The final step is computing the polynomial, m , by using f_p (equation 2.13):

$$m = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}$$

2.5 NTRU Parameters Selection

There are several considerations that need to be taken like eliminating the possibility of brute force searches, minimizing decryption failure, requiring an impossibly large estimated time for the LLL algorithm to return a short vector, maintaining efficiency and speed, the parameters of the NTRU cryptosystem must be selected [31][5]. The message and keys spaces should be chosen to get an adequate level of combinatorial security.

Therefore, for keeping the cryptosystem as efficient as possible they (the key and the message space) should be chosen to maximize the security.

2.5.1 Small Polynomial

The coefficients of private polynomials (f and g) must be small, where $(0, 1)$ used for binary representation or $(-1, 0, 1)$ for ternary.

NTRU PKCS used these polynomials to compute the convolutions multiplication in three stages:

- Key generation: $h = p \cdot f \cdot q \cdot g$.
- Encryption process: $e = r \cdot h + m \pmod{q}$.
- Decryption process: $a = f \cdot e \pmod{q}$.

In other word, to increase the speed of the convolution multiplication must use a few coefficients which they are nonzero polynomial. Then, some of the NTRU cryptosystem polynomials are small (not too small). To avoid the short vector attacks, the N must equal to 251 for the commercial applications and the short vector should have 72 non-zero entries[5].

2.5.2 Choice Considerations of NTRU Parameters

The values of the parameters should be carefully selected in order to maximize efficiency. To prevent attack due to Gentry [37], the value of N must be prime and to prevent lattice-based attack, also the parameter N must be large enough. When N value is a power of 2 the FFT (Fast Fourier Transformation) can be used to find the convolution multiplication.

The parameter p is much smaller than q and both should be selected prime relatively. The greater common divisor $\gcd(p, q) = 1$ to safeguard the security of NTRU cryptosystem.

The polynomials f , g , r , and m are drawn from a small space relatively, they may be vulnerable to an exhaustive search. So, the number of 1's and -1's for the keys (polynomials f and g) that are used to generate the public key need to be selected according to certain values. The parameters should be taken carefully in order to success the decrypt.

So, there are considerations to set the parameters in order to maximize efficiency and security for NTRU, table (2.3) illustrates it [32][38].

Table (2.3): The parameters sets [39]

Security level	N	q	p	f		g	r
				d_f	$d_f - 1$		
Moderate	107	64	3	15	14	12	5
Standard	167	128	3	61	60	20	18
High	263	128	3	50	49	24	16
Highest	503	256	3	216	215	72	55

2.6 Low Hamming Weight Polynomial

Low hamming weight polynomial used to speed up the convolution multiplication when the coefficients of a polynomial are 1 and 0 only. The require time for multiplying small polynomial by another polynomial is much lower than the ordinary convolution multiplication process. By applying this way, the convolution does not need any multiplications and every coefficient is just the sum of two values [40]. For more details, the following example (2.5) shows the convolution multiplication for a polynomial when $N=3$.

Example 2.5

$$[4 + 2X + 2X^2] * [2 + 5X + 3X^2]$$

$$[4, 2, 1] * [2, 5, 3] = 4 \cdot [2, 5, 3]$$

$$+ 2 \cdot [3, 2, 5]$$

$$+ 1 \cdot [5, 3, 2]$$

$$= [8, 20, 12]$$

$$+ [6, 4, 10]$$

$$+ [5, 3, 2] = [8+6+5, 20+4+3, 12+10+2] = [19, 27, 24]$$

The operations that needed for this example are nine multiplies and six additions, but when the coefficients for one of the polynomials are 1's and 0's only as the following:

$$[1, 1, 0, 0, 1] * [2, 4, 3, 9, 5] = 1 \cdot [2, 4, 3, 9, 5]$$

$$+ 1 \cdot [5, 2, 4, 3, 9]$$

$$+ 0 \cdot [9, 5, 2, 4, 3]$$

$$+ 0 \cdot [3, 9, 5, 2, 4]$$

$$+ 1 \cdot [4, 3, 9, 5, 2]$$

$$= 1 \cdot [2, 4, 3, 9, 5]$$

$$+ 1 \cdot [5, 2, 4, 3, 9]$$

$$+ 1 \cdot [4, 3, 9, 5, 2]$$

$$= [2+5+4, 4+2+3, 3+4+9, 9+3+5, 5+9+2]$$

$$= [11, 9, 16, 17, 16]$$

As clear above, each coefficient is just the sum of two values, and no need to multiplications at all.

Generally, if a is a small polynomial have d_n coefficients, and b is other polynomial therefore every coefficient in the result $(a * b)$ by sum operation for the coefficients of b only.

So, each small polynomial must have about 72 non – zero entries with $N = 251$ in order to ensure the security considerations of using low hamming weight polynomials and must certain that the search of brute force is as yet impracticable. Therefore, when $N = 251$ and f_i have eight 1's then $(251 \text{ choose } 8)^3$ is the number of possible f_s [41].

2.7 The Advantages of NTRU

NTRU PKC has many advantages as the following [42] [43]:

- The encryption and decryption process are more efficient for software and hardware.
- Key generation process is fast enough, this feature useful for disposable keys because generation process is computational cheap.
- NTRU does not need large memory, this is useful for devices that have low memory.
- NTRU has been observed to be multiple times, which is faster compared to cryptosystems such as ECC and RSA.
- It needs least resources such as CPU and battery life.
- For large-scale deployments, it decreases server resource utilization.
- When integrated NTRU with SSL, the data throughput (over RSA) is improved.

- When code size be a major limitation, NTRU is considered ideal in environments that hard to access or low power.
- Resistant to Quantum computing attacks.

2.8 Encryption and Decryption Speeds

The encryption speed is very important for IoT environment. NTRU PKCS shows perfect speed (both encryption and decryption) which make it suitable for IoT devices, therefore this thesis selected NTRU cryptosystem as well as, other features existed with NTRU. The encryption and decryption time for NTRU illustrated in table (2.4) below:

Table (2.4): Encryption and Decryption of NTRU [44]

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

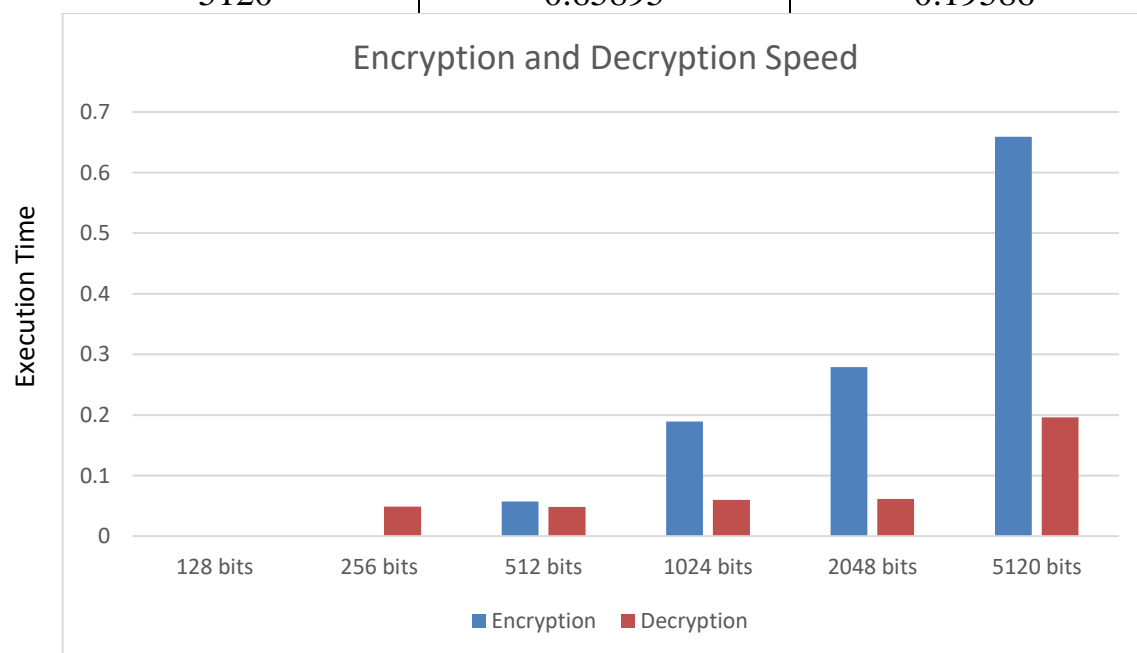


Figure (2.3): Encryption and Decryption Speeds of NTRU

The table (2.4) illustrates the fact that the decryption time is a lot faster than encryption time.

2.9 Attacks on NTRU Cryptosystem

Attack defined as a technique or method that used to circumvent the security of a cryptographic through finding a weakness point in it. The attack on cryptography may only apply to a certain set of cryptosystem parameters, which does not mean its complete breakdown. An attack may enable the attacker to discover a little information about a certain message or restore it completely, or discover the encryption keys[45].

2.10 Lattice-based Attack

This section presents two methods used to attack NTRU cryptosystem that based on lattices, lattice-based attack on the public key and lattice-based attack on the cipher text. Primarily, these two methods are performed by using the LLL algorithm [46].

2.11 The LLL Algorithm

Arjen Lenstra, Hendrik Lenstra and L'aszl'o Lov'asz in 1980 presents reduction algorithm based on lattice, namely LLL algorithm (2.9) [4]. This algorithm is the most famous algorithm for reducing a lattice basis and was a real breakthrough. It provides a reduced basis of a lattice and also runs in polynomial time. It returns an approximation of the short vector. The exponential approximation factor for this algorithm is $2^{(n-1)/2}$. It has good advantages in practice than in theory, especially for low dimensions and for many lattice problems, an approximation of the shortest vector or a

reduced basis suffice. The complexity of this reduction algorithm to calculate the LLL-reduced basis is $O(n^2 \log^3 B)$ [47].

The LLL algorithm depending on the Gram-Schmidt orthogonalization. The Gram-Schmidt operation is an iterative method to orthonormalize the basis of a vector space [48].

Theorem 1 Suppose the vector space V , with n dimension and a basis of $V: (b_1, \dots, b_n)$

.

$$b_1^* = b_1, \quad b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*,$$

For $j < i$

$$\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$$

where (b_1^*, \dots, b_n^*) is an orthogonal basis of V .

The bases (b_1^*, \dots, b_n^*) and (b_1, \dots, b_n) satisfy

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ \mu_{2,1} & 1 & 0 & 0 & \dots & 0 \\ \mu_{3,1} & \mu_{3,2} & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mu_{n-1,1} & \mu_{n-1,2} & \mu_{n-1,3} & \dots & q & 0 \\ \mu_{n,1} & \mu_{n,2} & \mu_{n,3} & \dots & \mu_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} b_1^* \\ b_2^* \\ b_3^* \\ \vdots \\ b_{n-1}^* \\ b_n^* \end{bmatrix} \quad \dots (2.14)$$

The basis (b_1^*, \dots, b_n^*) is an orthogonal basis, generally, if (b_1, \dots, b_n) is the basis of the lattice, L (b_1^*, \dots, b_n^*) is not a basis for L .

Algorithm (2.8): Gram-Schmidt process [49]

Input: A basis (b_1, \dots, b_n) of a space vector $V \in R^n$
Output: An orthogonal basis (b_1^*, \dots, b_n^*)
Step 1: Set $b_1^* = b_1$
Step 2: for $i = 1, 2, \dots, n$ do
Step 3: for $j = 1, 2, \dots, i - 1$ do
Step 4: Compute $\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$
Step 5: end for
Step 6: Compute $b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$
Step 7: end for

Algorithm (2.9): LLL Algorithm [49]

Input: $B = (b_1, \dots, b_n) \in \mathbb{Z}_n$, $\delta \in (\frac{1}{4}, 1)$ (B is a basis)
Output: δ : LLL reduced basis of $\Lambda(B)$
Step 1: procedure LLL(B, δ)
Step 2: for $i = 1$ to n do
Step 3: for $j = i-1$ to 1 do
Step 4: $c_{ij} = \left\lfloor \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right\rfloor$
Step 5: $b_i = b_i - c_{ij} b_j$
Step 6: end for
Step 7: end for
Step 8: if $\delta \ \pi_i(b_i)\ ^2 > \ \pi_i(b_{i+1})\ ^2$ for some i then
Step 9: swap b_i and b_{i+1} , and go to step 1.
Step 10: end if
Step 11: return B
Step 12: end procedure

2.11.1 Lattice-based Attack on the Public Key

By using the LLL algorithm to cryptanalyze NTRU PKC, to find both the original private key or other key (called an alternative key can be

used in decryption process and able to decrypt the cipher texts). This method was first presented by Don Coppersmith and Adi Shamir [48] in 1997. The goal is to compute the private key $(f; g)$ just by depending on the public parameters (public key). The attack is based on the following $(2N \times 2N)$ -lattice determined by h and p , which are both public:

The NTRU-Lattice is the lattice spanned by the basis

$$L = \begin{bmatrix} I_N & 0 \\ H & q \cdot I_N \end{bmatrix} \quad \dots (2.15)$$

The dimension $2N$, this means that the shortest vector length will be $\det(L)^{\frac{1}{2N}} = \sqrt{q}$ approximately as described in [56]. The vector $(f; g)$ and circular shifts are also in the NTRU that generated by equation (2.15) above. Further, it is very likely that this vector is a shortest vector, since its norm is $\sqrt{2 \cdot df - 1 + 2 \cdot dg} < \sqrt{q}$ in the cases recommended by the NTRUEncrypt. So, by reducing $\Lambda(L)$ with the LLL algorithm, the probability is high to find (f, g) or circular shifts of it, which can also be used to decrypt the message. Two conditions must be available in the row that have a good chance to represent the secret key (f, g) or an alternate key to decrypt the secret message.

2.11.2 Lattice-based Attack on the Cipher Text

This method is the same as to the attack on the private key with a little of additions, by using the LLL algorithm to cryptanalyze the scheme, to discover the plaintext depending on the public key and the encrypted

message for the original NTRU cryptosystem. This method was first presented by Don Coppersmith and Adi Shamir [50] in 1997.

The NTRU-Lattice is the lattice spanned by the basis

$$L = \begin{bmatrix} I_N & 0 & 0 \\ h & q \cdot I_N & c \end{bmatrix} \quad \dots (2.16)$$

Where the public key represented by h and c is the cipher text. The shortest vector length is $\det(L)^{\frac{1}{2N}} = \sqrt{q}$ approximately as described in [51]. The vector $(f; g)$ and circular shifts are also in the NTRU that generated by equation (2.16) above. Further, it is very likely that this vector is a shortest vectors, since its norm is $\sqrt{2 \cdot df - 1 + 2 \cdot dg} < \sqrt{q}$ in the cases recommended by the NTRUEncrypt. So, by reducing $\Lambda(L)$ with the LLL algorithm, the probability is high to find the message(plaintext).

Chapter Three

Design of The Proposed System

Chapter Three

Design of The proposed system

3.1 Introduction

NTRU cryptosystem suffers from lattice-based attack by the LLL algorithm. This chapter presents the work of two proposals that introduces modification on NTRU cryptosystem in order to be secure against the LLL algorithm attack. The first proposal suggests adding a new parameter to the parameters of the original NTRU, the value of this parameter depends on content on LFSR in certain stage .

In other hand, in recently years, there is a large data transfer over network and for example the key length of NTRU cryptosystem is 256 or 512 integer in each generation. So, to encrypt all data, you need to use many generation keys. To generate these keys, there are two ways:

- Firstly, repeating the same private keys many times to generate the public key.
- Secondly, using different private keys to generate the public keys for the next time that need keys.

The first way yields a weak encryption process result, and gives a gift for crypto-analyzer to recover the secret data. The second way needs to use many different secret private keys, f & g . The second part of proposal two depends on the context of the second way by exploiting features of convolution multiplication in the truncated polynomial in the ring $\mathbb{Z}[x]/(x^N - 1)$. A particular feature in convolution multiplication between any two polynomial coefficients leads to a significant change in

multiplication results. This feature exploited to generate different dynamic public keys sequences without need to use new private keys, new polynomials f & g . This modification does not require additional private keys or memory and without affecting on the security strength, this can done through the use of the previous results.

Therefore, two parts for the proposal two: one proposes the use of swapping operations for the public key values to prevent the LLL algorithm attack, while second part of this proposal generates a long sequence of keys dynamically.

3.2 Objectives

The objectives that prompted to propose modifications on NTRU are:

1. To avoid lattice-based attack by using LLL algorithm that attacks the original NTRU cryptosystem depending on knowing the public key of the system to discover either the private key, or an alternative private key which is useful to decrypt the cipher texts. Also, to avoid the same attack on the ciphertext to discover the original message without needs to private keys.
2. To give strength to the encryption process by long generating sequence of keys dynamically which based on the public key values, this means each block of plaintext will encrypted by its own keys (independent key) that differ from the others.

3.3 The Proposal One

In the proposal one, a modification on NTRU public key cryptosystem is introduced. The modification depends on using four parameters (N, p, X, q) instead of using three parameters, N, p , and q in the original NTRU, the parameter X is a prime number whose value changes dynamically in each generating key process. The values of X depends on the contents of LFSR so that if the value of $SR[i] = 0$, then we multiply by the first value of X , but if the value is 1, we multiply by the second value of X ($SR[i]$ represents the value of element in LFSR at the location i). In advance, the two parties must determine the values of parameter X and the content of the LFSR. The aim of this modification is to thwart the ability of the LLL algorithm in attacking the NTRU algorithm under any conditions.

3.3.1 Key Generation

The key generation steps of this proposal as the following:

1. Generating the public key, h , by the original NTRU.
2. Firstly, fill and shifting the content of LFSR with N times, to get a random sequence of 0's and 1's that can be used for a several purposes, this steps use to change the value of the parameter X to second value by depending on the agreement between two parties (sender and receiver).
3. Finally, multiply the public key, h , by the appended parameter values (X) such as:

$$newh = h * X \pmod{q} \quad (3.1)$$

The algorithm (3.1) illustrates the method of public key generation in the proposed method.

Algorithm (3.1): Public key Generation

Input: N, p, q, x_1, x_2 , two polynomials (f and g), seeds of SR (LFSR)
Output: new public key ($newh$)
Step 1: Generating the public key, h //by the original NTRU. Step 2: Fill SR and shifting it N times Step 3: for $i = 0$ to $N - 1$ do Step 4: If $SR[i] = 0$ then Step 5: $X = x_1$ // x_1 agreed between sender and receiver. Step 6: $newh[i] = h[i] \cdot X \pmod{q}$ Step 7: else Step 8: $X = x_2$ // x_2 agreed between sender and receiver. Step 9: $newh[i] = h[i] \cdot X \pmod{q}$ Step 10: end if Step 11: end for

Where SR is a LFSR and $SR[i]$ is value of element in LFSR at the location (i) (that may be 0 or 1), and the values of polynomial, h , is the public key that consider as goal to the attack.

3.3.2 Encryption

The encryption process steps are as the following:

1. Convert the message, m , into the polynomial form whose coefficients are chosen modulo p .
2. Randomly, the user must choose r (small polynomial where $r \in R$) to scramble the message.
3. Compute the cipher text , e , is as a polynomial:

$$e = (newh.X_q^{-1}) * r + m \pmod{q} \quad \dots (3.2)$$

Where X_q^{-1} is a modular multiplicative inverse of X modulo q (Algorithm 2.2).

The algorithm (3.2) illustrates the encryption process.

Algorithm (3.2): Encryption process

Input : N, q , Public Key h , message m , random polynomial r , SR, x_1, x_2
Output : The encrypted message, e .
Step 1: Fill SR and shifting it N times Step 2: for $i = 0$ to $N - 1$ do Step 3: if $SR[i] = 0$ then Step 4: $X = x_1$ // x_1 agreed between sender and receiver. Step 5: $h[i] = (h[i] * X_q^{-1}) \pmod{q}$ Step 6: else Step 7: $X = x_2$ // x_2 agreed between sender and receiver. Step 8: $h[i] = (h[i] * X_q^{-1}) \pmod{q}$ Step 9: end if Step 10: end for Step 11: MultiplayPoly (r, h, e, N, q) Step 12: for $i = 0$ to $N - 1$ do

Step 13: $e[i] = e[i] + m[i] \bmod q$ Step 14: end for

Where X_q^{-1} is the inverse of X modulo q , depends on the same certain stages in the LFSR in the key generation step and MultiplayPoly is a function to compute convolution multiplication of two polynomials with degree $(N-1)$ (algorithm (2.4)).

Finally, the decryption process is the same to the original NTRU.

3.4 The Proposal Two

As we mentioned earlier, this method will use a different public key with each block of plaintext with length N , because repeating the same public key that will make the encryption process weak, which is considered as a gift for the crypto analyzer, therefore the public key must be changed dynamically.

Also to avoid weakness points of NTRU cryptosystem in order to prevent attack it by LLL algorithm. The generation process of a new public key $(h_1, h_2, \dots, \text{etc.})$ need choosing a new private key. this proposal introduce a modification on the original NTRU cryptosystem, this modification is doing by using swapping operations for certain values of the public key (in advance the sender and receiver should agree to select two locations (modulo N) in order to swap their values).

Therefore, this proposed modification will generate a dynamic new private key (polynomial g_i) from previous public key (h_{i-1}) by applying the equation:

$$g_{i+1} = (h_i \bmod p) - 1 \quad \dots (3.3)$$

The value of p is either 2 or 3, where $p = 2$ is used for binary representation, and $p = 3$ is used for ternary $\{-1, 0, 1\}$ according to work of NTRU cryptosystem. Figure (3.1) illustrates the flowchart of the proposed algorithm.

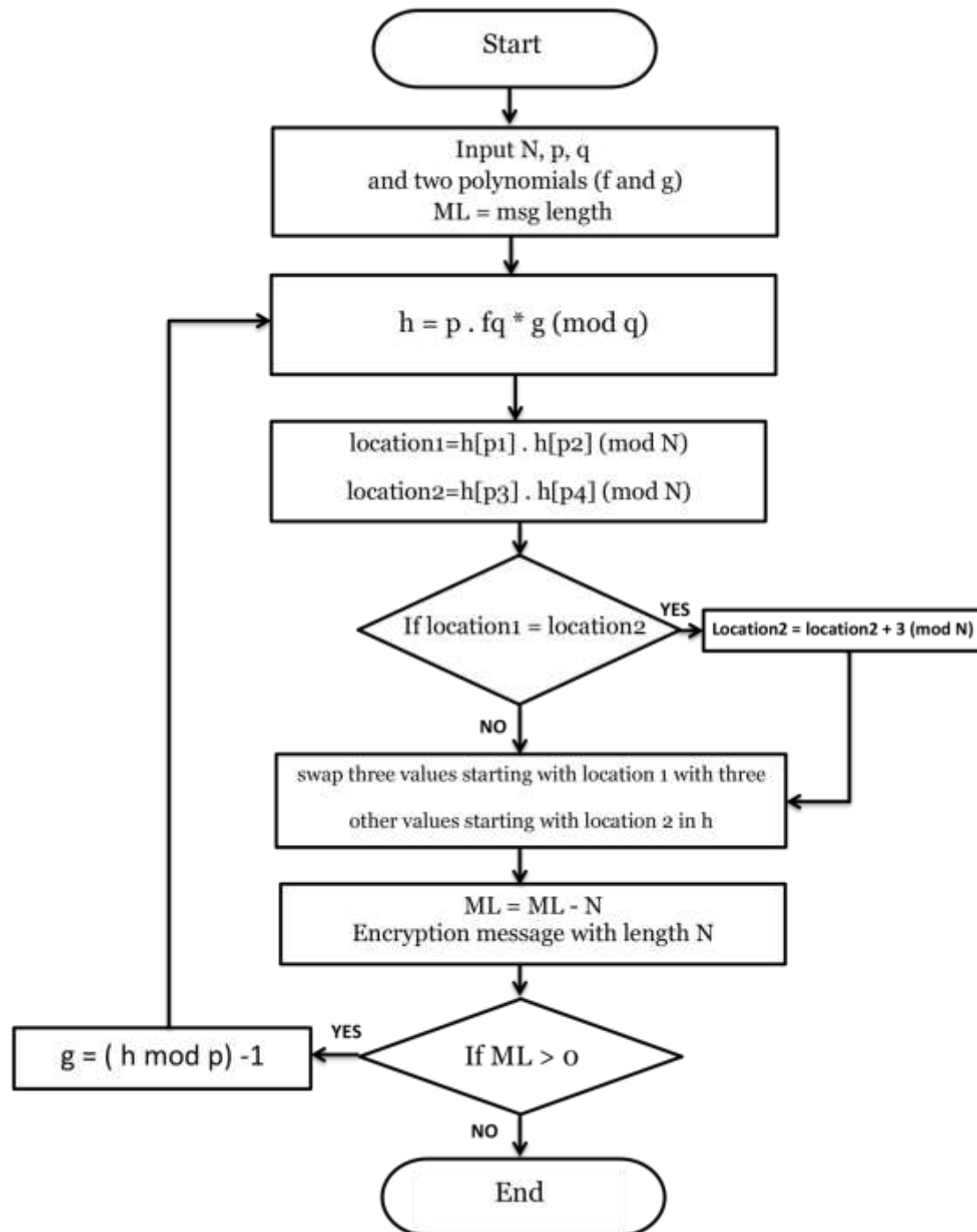


Figure (3.1): Flowchart of dynamic key generation

Where p_1, p_2, p_3 and p_4 are a positions in the public key, h , (must be agreed previously between sender and receiver, and the value of these positions must be mod N), where $h[p_1]$ is the value in the public key, h according to the value of position p_1 .

3.4.1 Key Generation

The algorithm (3.3) illustrates the public key generation of proposed modification.

Algorithm (3.3): Public key Generation

Input: N, p, q , two polynomials (f and g), p_1, p_2, p_3 and p_4
Output: new public key ($newh$)
Step 1: Generating the public key, h //by the original NTRU. Step 2: $Location_1 = (h[p_1] \cdot h[p_2]) \bmod N$ Step 3: $Location_2 = (h[p_3] \cdot h[p_5]) \bmod N$ Step 4: If $Location_1 = Location_2$ then Step 5: $Location_2 = Location_2 + 3$ Step 6: for $i = 0$ to 2 do // swapping three values. Step 7: $Location_1 = (Location_1 + i) \bmod N$ Step 8: $Location_2 = (Location_2 + i) \bmod N$ Step 9: $temp = h[Location_1]$ Step 10: $h[Location_1] = h[Location_2]$ Step 11: $h[Location_2] = temp$ Step 12: end for

Where $Location_1$ and $location_2$ are parameters represents the positions in the public key, polynomial, h , which will be used to swap the values of it. The loop that starting from step 5 to step 11 are used to swap three values of public key with other three values of the same public key, the first three values starting with $location_1$ and the second starting with $location_2$.

3.4.2 Encryption

The algorithm (3.4) illustrates encryption process for the proposed algorithm:

Algorithm (3.4): Encryption process

Input : Public key h , N , q , random polynomial r , message m , length of message (ML).
Output : The encrypted block of message, e with length N .
Step 1: do // for each block with length N . Step 2: for $i = 0$ to 2 do Step 3: $temp = h[Location_1]$ Step 4: $h[Location_1] = h[Location_2]$ Step 5: $h[Location_2] = temp$ Step 6: $Location_1 = (Location_1 - 1) \bmod N$ Step 7: $Location_2 = (Location_2 - 1) \bmod N$ Step 8: if $Location_1 < 0$ then Step 9: $Location_1 = Location_1 + N$ Step 10: if $Location_2 < 0$ then Step 11: $Location_2 = Location_2 + N$ Step 12: end for Step 13: MultiplayPoly (r, h, e, N, q) // algorithm (2.4) Step 14: for $i = 0$ to $N - 1$ do Step 15: $e[i] = e[i] + m[i] \bmod q$ Step 16: for $i = 0$ to $N - 1$ do Step 17: $g_{i+1} = (h_i \bmod p) - 1$ Step 18: $ML = ML - N$ Step 19: while ($L > 0$) // end of do

Firstly, the values of public key, h , that have been swapped in the key generation step (algorithm 3.3) should be returned to their original locations in the public key, h , to get the original public key in order to be used in encryption process.

The values of $location_1$ and $location_2$ are the last values reached through the previous steps (section 3.4.1) in the process of calculating the public key. Finally, the decryption process is the same to the original NTRU.

Chapter Four

Results and Evaluation

Chapter Four

Results and Evaluation

4.1 Introduction

This chapter is dedicated to present the results and tests performance evaluation of the proposed cryptosystem. Firstly, generating the public keys and then attacked it by the LLL algorithm for the original NTRU and the proposals, as well as the cipher text also attacked by the LLL algorithm. In order to be more accurate explanation, this chapter presents the same input parameters for the original NTRU and the proposed algorithms to show the improvements of the proposed algorithms upon the original NTRU.

Section 4.3.1 presents LLL algorithm attack on the public key by the example 4.1 for the original NTRU (section 4.3.1.1) and the proposal one & two (section 4.3.1.2 and section 4.3.1.1) respectively. Section (4.3.2.1) presents LLL algorithm attack on the cipher text. Finally, shows performance analysis in section (4.5).

4.2 Initialization

Computer programs in C++ language (for the original NTRU, the proposals, and LLL algorithm) were implemented on the laptop. The specifications of this laptop are: Operating system Windows 7 Ultimate 64-bit, processor Intel(R) Core i5 2.67GHz, and installed memory (RAM) with 4.00GB. Different examples for NTRU and all proposed cryptosystems with different values of their parameters (the public key, h , and the cipher text, e) of these examples attacked by the LLL.

4.3 LLL Algorithm Attack

This section will test the LLL algorithm attack on both the private keys and the cipher text for the original NTRU and the proposed algorithms. Section (4.3.1) shows the attack results on the public key and section (4.3.2) shows the attack results on the cipher text.

4.3.1 LLL Algorithm attack on the Public Key

This section shows the attack on the public key (the polynomial h) to discover the private keys (the polynomials f, g) or alternative private keys that can be used to decrypt the encrypted message. In this thesis, the same parameters for the original NTRU and the proposed algorithms are used to show the performance differences between each of them, as we mentioned earlier.

Example 4.1

Let the parameters are $(N = 11, p = 3, q = 32)$ and the two polynomials (f, g)

$$f = X + X^4 - X^5 - X^6 + X^7 + X^8 - X^{10},$$

$$g = 1 - X^3 - X^5 - X^6 + X^8 + X^9.$$

For this example, the section (4.3.1.1) illustrates the LLL algorithm attack on public key of the original NTRU cryptosystem; sections (4.3.1.2) and (4.3.1.3) illustrates the same attack on the proposal one and the proposal two, respectively.

4.3.1.1 The Attack on the Original NTRU

The generating public key, h , for the previous parameters is (section 2.4.2):

$$\begin{aligned} h &= p \cdot g * f_q(\text{mod } q) \\ &= 8 + 25X + 22X^2 + 20X^3 + 12X^4 + 24X^5 + 15X^6 \\ &\quad + 19X^7 + 12X^8 + 19X^9 + 16X^{10}. \end{aligned}$$

To attack this generated public key, one constructs the lattice basis, L (table 4.1) according to equation (2.15), for the above public key, so $h = p_q^{-1} \cdot h$ (where p_q^{-1} is the multiplicative inverse of p in Z_q , algorithm (2.2)).

Table (4.1): the lattice basis L of h

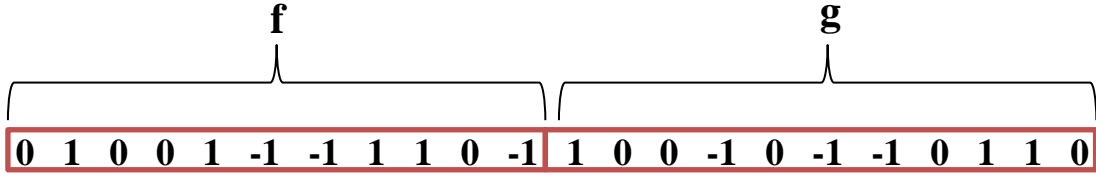
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
88	176	209	132	209	165	264	132	220	242	275	32	0	0	0	0	0	0	0	0
275	88	176	209	132	209	165	264	132	220	242	0	32	0	0	0	0	0	0	0
242	275	88	176	209	132	209	165	264	132	220	0	0	32	0	0	0	0	0	0
220	242	275	88	176	209	132	209	165	264	132	0	0	0	32	0	0	0	0	0
132	220	242	275	88	176	209	132	209	165	264	0	0	0	0	32	0	0	0	0
264	132	220	242	275	88	176	209	132	209	165	0	0	0	0	0	32	0	0	0
165	264	132	220	242	275	88	176	209	132	209	0	0	0	0	0	0	32	0	0
209	165	264	132	220	242	275	88	176	209	132	0	0	0	0	0	0	0	32	0
132	209	165	264	132	220	242	275	88	176	209	0	0	0	0	0	0	0	0	32
209	132	209	165	264	132	220	242	275	88	176	0	0	0	0	0	0	0	0	0
176	209	132	209	165	264	132	220	242	275	88	0	0	0	0	0	0	0	0	0

Secondly, applying LLL algorithm (algorithm (2.8)) to the lattice basis, L , leads to the basis like the following table 4.2:

Table (4.2): The LLL algorithm results

no	The results																					no. 1	no. -1	difference	
1	0	0	-1	1	1	-1	-1	0	1	0	-1	0	1	0	1	1	0	-1	-1	0	-1	0	7	6	1
2	1	0	-1	0	1	0	0	1	-1	-1	1	1	1	0	1	0	0	-1	0	-1	-1	0	7	6	1
3	1	1	-1	-1	0	1	0	-1	0	0	-1	1	1	0	-1	-1	0	-1	0	0	1	0	6	7	1
4	-1	1	1	-1	-1	0	1	0	-1	0	0	0	1	1	0	-1	-1	0	-1	0	0	1	6	7	1
5	1	1	0	-1	0	1	0	0	1	-1	-1	0	1	1	0	1	0	0	-1	0	-1	-1	7	6	1
6	0	-1	1	1	-1	-1	0	1	0	-1	0	1	0	1	1	0	-1	-1	0	-1	0	0	7	6	1
7	0	1	0	0	1	-1	-1	1	1	0	-1	1	0	0	-1	0	-1	-1	0	1	1	0	7	6	1
8	-1	0	0	-1	1	1	-1	-1	0	1	0	0	0	1	0	1	1	0	-1	-1	0	-1	6	7	1
9	0	1	0	-1	0	0	-1	1	1	-1	-1	-1	0	-1	0	0	1	0	1	1	0	-1	6	7	1
10	1	0	-1	0	0	-1	1	1	-1	-1	0	0	-1	0	0	1	0	1	1	0	-1	-1	6	7	1
11	1	-1	-1	0	1	0	-1	0	0	-1	1	1	0	-1	-1	0	-1	0	0	1	0	1	6	7	1
12	-2	3	0	0	2	2	1	1	2	-3	-1	7	-5	-3	3	-4	3	5	-3	-1	-2	0	-	-	-
13	1	0	1	0	1	-2	-3	-4	3	0	0	-4	2	4	-4	-2	-1	0	8	-5	-1	3	-	-	-
14	1	-3	1	2	3	-2	0	-2	-2	-1	-1	5	1	2	-1	-7	5	2	-3	3	-2	-5	-	-	-
15	-1	1	-3	1	2	3	-2	0	-2	-2	-1	-5	5	1	2	-1	-7	5	2	-3	3	-2	-	-	-
16	-2	3	-2	1	-3	1	-2	-1	-2	1	4	6	2	3	6	0	1	5	2	6	1	0	-	-	-
17	-3	-3	3	0	1	2	1	0	0	4	-2	0	6	-5	-3	3	-3	3	5	-4	0	-2	-	-	-
18	2	1	-1	3	-3	-3	-2	2	0	1	0	3	4	-4	-2	-2	0	7	-4	-2	4	-4	-	-	-
19	1	5	-2	0	-7	2	-5	0	-2	4	0	1	0	-3	4	1	-5	3	-1	1	1	-2	-	-	-
20	0	-1	0	-2	-1	1	-3	3	3	2	-2	2	-4	4	-3	-4	4	2	2	0	-7	4	-	-	-
21	2	0	1	0	2	1	-1	3	-3	-3	-2	-4	-2	4	-4	3	4	-4	-2	-2	0	7	-	-	-
22	2	-3	2	-2	1	-3	-1	-1	1	4	-1	1	2	6	1	2	5	3	6	1	-1	6	-	-	-

From table (4.2), since the coefficient values of the private polynomials f and g are in $[-1, 0, 1]$, so one can eliminate the rows that include values out of $[-1, 0, 1]$. The first seven rows vectors contain exactly as many 1's and -1's, so one can expect for f and g and coincide exactly with



corresponding to the polynomials:

$$f = X + X^4 - X^5 - X^6 + X^7 + X^8 - X^{10},$$

$$g = 1 - X^3 - X^5 - X^6 + X^8 + X^9.$$

The difference between the number of 1's and number of 0's (the coefficients of the polynomials f and g) is 1; this is one of the conditions of NTRU cryptosystem. This applies for the rows vectors (1-6 and 8-11) that consider as alternative secret keys that give the same public key and can be used to decrypt the cipher text. To prove that, when applying NTRU key generation steps on the first row in table (4.2) that is (f & g respectively):

(0 0 -1 1 1 -1 -1 0 1 0 -1 0 1 0 1 1 0 -1 -1 0 -1 0),

where the polynomial, f , represented in first N entries for the above sequence: $f = 0 0 -1 1 1 -1 -1 0 1 0 -1$,

while the polynomial, g , is the second N entries:

$$g = 0 1 0 1 1 0 -1 -1 0 -1 0$$

Then compute the public key, h , to get:

$$\begin{aligned} h &= p \cdot g * f_q(\text{mod } q) \\ &= 8 + 25X + 22X^2 + 20X^3 + 12X^4 + 24X^5 + 15X^6 \\ &\quad + 19X^7 + 12X^8 + 19X^9 + 16X^{10}. \end{aligned}$$

As it is clear, this is the same public key of the original NTRU for the example 4.1. Therefore, more than one private key (the polynomials f , g) gives the same public key, h .

Therefore, when performing LLL algorithm on lattice basis, L , to get a basis, all values in the row of LLL algorithm output are -1, 0, or 1 only $row[k]$ in $\{-1,0,1\}$, and the difference between the number of "1" and the number of "-1" equal 1.

$$\sum_{k=1}^N row[k] = 1 \quad (4.1)$$

Applying the equation (4.1) for the rows of table (4.2), one can show that only the contents of the sequence (-1, 0, 1), this is useful for knowing the secret keys form. In other word, the rows (1-11) fulfill the equation (4.1). Therefore, the LLL algorithm discovers the original private keys (f and g) as well as discovers alternative keys.

4.3.1.2 The Attack on the Proposal One

To compare with original NTRU and our proposal, the same parameters that are used in the example (4.1) will be used with this modification to show the lattice-based attack cannot discover the private keys of this proposal while it discovered the private keys of the original NTRU

(section 4.3.1.1).

The steps of the proposal one: The first step fill SR with its seeds LFSR, for example when the LFSR contents this sequence (1 1 1 0 0 1 0 1 1 1 1) and then shifting it with length N (here $N=11$) to get (0 0 1 0 1 1 1 0 0 1 0), next applying the other steps of the proposed algorithm

(suppose the sender and receiver selected the values of X as X=3 if SR[i] = 0 else X=5 for this example) to get on the public key(according to the algorithm (3.1)):

$$h = 24 + 11X + 14X^2 + 28X^3 + 28X^4 + 24X^5 + 11X^6 + 25X^7 + 4X^8 + 31X^9 + 16X^{10}.$$

Now, test the lattice-based attack by using LLL algorithm on this public key that is constructing the lattice basis like table (4.3):

Table (4.3): The lattice basis L for h (proposal one)

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
264	176	341	44	275	121	264	308	308	154	121	32	0	0	0	0	0	0	0	0	0
121	264	176	341	44	275	121	264	308	308	154	0	32	0	0	0	0	0	0	0	0
154	121	264	176	341	44	275	121	264	308	308	0	0	32	0	0	0	0	0	0	0
308	154	121	264	176	341	44	275	121	264	308	0	0	0	32	0	0	0	0	0	0
308	308	154	121	264	176	341	44	275	121	264	0	0	0	0	32	0	0	0	0	0
264	308	308	154	121	264	176	341	44	275	121	0	0	0	0	0	32	0	0	0	0
121	264	308	308	154	121	264	176	341	44	275	0	0	0	0	0	0	32	0	0	0
275	121	264	308	308	154	121	264	176	341	44	0	0	0	0	0	0	0	32	0	0
44	275	121	264	308	308	154	121	264	176	341	0	0	0	0	0	0	0	0	32	0
341	44	275	121	264	308	308	154	121	264	176	0	0	0	0	0	0	0	0	0	32
176	341	44	275	121	264	308	308	154	121	264	0	0	0	0	0	0	0	0	0	32

The next step is applying the lattice basis reduction algorithm (LLL) to above public key, h , leads to the basis like the following:

Table (4.4): The LLL algorithm results (proposal one)

-4	-1	1	0	2	-2	0	3	-2	-1	3	-2	0	-4	0	3	1	-3	-2	0	-1	0
1	1	3	-2	-1	2	-2	1	-1	4	-2	4	-1	1	-3	-1	-2	2	-2	1	1	0
4	-1	0	2	0	0	0	0	0	-1	-3	3	4	-1	2	-3	2	-1	2	-2	2	0
1	1	3	2	0	-2	0	0	-3	-1	1	0	-1	-5	0	-2	-2	-5	-1	-1	1	0
3	-1	-1	1	-3	0	0	0	-1	1	1	5	-2	-2	4	-3	-1	-1	1	1	0	-2
0	1	3	0	1	-1	0	0	2	1	-3	0	-1	2	-2	1	1	2	-4	-1	3	-1
-3	-1	3	-2	1	-2	2	1	1	-3	2	-4	-1	1	-1	2	-1	0	1	-1	-1	-3
-5	-1	2	-3	-1	1	-2	1	1	0	2	-2	1	-1	-3	-2	0	1	0	2	-3	-1
-2	0	-1	-1	-3	0	0	-1	0	5	0	4	1	0	-4	-2	1	4	3	2	0	-1
2	1	1	-4	-1	1	-2	3	0	1	-2	-1	3	-3	1	-3	-1	3	0	1	-1	1
1	1	2	-2	1	1	-1	2	-1	-1	1	1	-1	-5	2	1	5	-2	0	-2	0	1
-4	-1	2	-4	-2	-3	1	1	3	4	-1	0	2	1	-1	2	-2	1	-2	1	-1	-1
0	-1	0	1	4	3	2	3	0	0	0	-1	0	0	-2	3	1	-1	3	0	-1	-2
-4	-3	-2	-3	0	0	0	0	1	0	-1	-3	-1	1	-3	0	1	2	1	0	0	2
0	-3	1	-1	0	-1	-2	0	0	-3	0	2	2	-1	-2	0	-3	-5	-2	0	2	-1
0	0	-3	0	0	-1	1	-3	-3	2	2	-2	-2	-4	0	-4	1	2	1	2	-2	0
5	4	0	2	1	-1	0	-3	-1	1	-4	0	0	0	-3	0	1	1	2	1	0	-2
1	-2	0	0	-1	-3	2	-2	3	-1	3	-1	-4	3	2	-2	1	0	0	-1	0	2
-1	-3	-4	-2	-3	0	1	-1	3	-1	-1	0	0	6	2	-1	0	1	-2	-1	-2	-3
3	-1	-3	2	-1	2	-2	0	2	-1	-1	5	-1	2	-1	-1	-1	2	-3	1	-2	-1
1	3	0	0	0	-2	1	-2	1	-3	-3	1	-2	-2	3	-3	2	-1	0	-2	2	2
3	0	0	0	1	3	1	0	-2	-3	1	1	-1	2	1	0	1	-3	1	3	-3	-2

From the table (4.4) results don't found any row vector in form of sequences $(-1, 0, 1)$ only or there is no row fulfills the equation (4.1) in order to say this vector is a candidate to be private keys (the polynomials f and g).

4.3.1.2 The Attack on the Proposal Two

Also the same parameters for the example (4.1) will be used for this proposal to compare the performance between the original NTRU cryptosystem and this proposed cryptosystem. In this example, we generate new private key, g_2 , from previous h_1 to generate new public key, h_2 , by using equation (3.3) as follows:

$$g_2 = (h_1 \bmod p) - 1 = 1 \ 0 \ 0 \ 1 \ -1 \ -1 \ -1 \ 0 \ -1 \ 0 \ 0.$$

Now, generating h_2 by equation (2.5) as follows:

$$h_2 = p \cdot fq * g_2 \pmod{q} = 14 \ 23 \ 12 \ 2 \ 1 \ 28 \ 23 \ 11 \ 3 \ 6 \ 31 ,$$

$$location_1 = h_0 \cdot h_4 \pmod{11} = 14 * 1 \pmod{11} = 3 ,$$

$$location_2 = h_1 \cdot h_5 \pmod{11} = 23 * 28 \pmod{11} = 6 .$$

Finally, swapping operation between three values of (h) starting in location (3) with three values starting in location (6) (algorithm 3.4 steps (13-19)):

$$h_2 = 14 \ 23 \ 12 \ 23 \ 11 \ 3 \ 2 \ 1 \ 28 \ 6 \ 31,$$

and then construct the lattice basis, L , as array in (2.6) to applying LLL algorithm such as the following:

Table (4.5): The lattice basis of L for h (proposal two)

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
154	341	66	208	11	22	33	121	253	132	253	32	0	0	0	0	0	0	0	0	0
253	154	341	66	208	11	22	33	121	253	132	0	32	0	0	0	0	0	0	0	0
132	253	154	341	66	208	11	22	33	121	253	0	0	32	0	0	0	0	0	0	0
253	132	253	154	341	66	208	11	22	33	121	0	0	0	32	0	0	0	0	0	0
121	253	132	253	154	341	66	208	11	22	33	0	0	0	0	32	0	0	0	0	0
33	121	253	132	253	154	341	66	208	11	22	0	0	0	0	0	32	0	0	0	0
22	33	121	253	132	253	154	341	66	208	11	0	0	0	0	0	0	32	0	0	0
11	22	33	121	253	132	253	154	341	66	208	0	0	0	0	0	0	0	32	0	0
208	11	22	33	121	253	132	253	154	341	66	0	0	0	0	0	0	0	0	32	0
66	208	11	22	33	121	253	132	253	154	341	0	0	0	0	0	0	0	0	0	32
341	66	208	11	22	33	121	253	132	253	154	0	0	0	0	0	0	0	0	0	32

Finally, performing the LLL algorithm to the above lattice results the basis like the following:

Table (4.6): The LLL algorithm results (proposal two)

0	-3	0	3	-3	3	-3	3	0	-3	3	5	-1	-1	2	-4	0	1	-1	0	0	-1
-1	-1	2	2	-1	2	-1	2	-1	-1	2	0	-5	-4	-3	-5	-1	-1	-2	-1	-1	-1
1	-2	-4	3	-1	-2	1	1	-2	0	1	-2	-2	-5	-2	-1	3	-1	-2	1	1	2
1	1	-4	-1	-2	-2	-1	-1	-2	0	0	1	2	-3	-2	3	-3	1	1	1	1	0
2	2	-2	6	3	-1	1	-2	-4	0	0	0	5	-3	3	1	0	1	-2	0	-3	0
-3	-1	3	-2	0	1	-4	2	1	1	3	-1	1	-4	2	-5	-6	0	5	1	2	-1
-3	-2	4	-1	-3	1	-1	-1	-2	2	3	0	-1	-3	4	-4	-1	2	-2	-2	-2	-5
-1	-1	-2	2	0	-3	-2	1	-1	0	1	2	2	1	1	-1	4	4	-2	1	-5	-3
1	-2	-2	4	-2	1	1	1	-2	-2	1	1	1	0	1	-3	-4	1	4	2	3	0
0	2	2	-1	0	2	-2	-2	-1	3	0	5	2	1	3	2	-1	1	0	-2	1	2
-3	-2	1	-2	-2	-2	-1	2	0	3	2	-1	3	0	-1	-3	-2	3	1	-1	0	1
1	3	3	0	2	0	-1	-1	2	2	0	-3	-1	0	-3	0	-4	1	1	2	2	3
-4	-3	-1	0	-1	1	0	0	1	1	1	0	-2	4	0	-2	2	2	-1	-2	1	-4
3	-2	-3	3	0	-1	0	2	1	-1	2	-3	5	1	1	-2	1	1	0	1	3	0
0	2	-3	-1	0	-4	1	0	0	0	0	-3	-3	2	-1	-1	4	0	-3	3	2	-2

4	0	-2	-1	2	-3	-1	-1	-3	3	1	0	0	-1	3	4	2	0	-1	0	-2	1
1	1	-1	0	2	3	-1	-3	0	2	0	1	4	0	5	-1	0	4	-1	-3	-1	0
2	0	-4	1	1	-3	-4	-1	0	0	1	5	2	2	-3	3	1	0	-2	0	1	1
1	-1	1	-2	1	2	-4	3	4	-3	3	4	-2	-1	-4	-1	1	2	1	2	1	-1
1	0	0	4	2	3	1	-2	0	1	2	-1	-1	-2	2	-6	1	-1	0	0	0	0
4	1	-2	3	3	5	2	-1	-1	-2	-2	2	1	1	0	2	-2	2	-3	1	-1	1
0	-2	1	2	0	2	2	0	1	0	0	3	0	1	3	-3	1	1	-2	-2	0	-6

Also, in the table (4.6) don't found any row vector in form of sequences $(-1, 0, 1)$ only in order to say this row is a candidate to be private keys (the polynomials f and g). In other words, there is no row vector fulfills the equation (4.1). Therefore, this proposal succeeds to bypass the lattice-based attack.

4.3.2 LLL Algorithm Attack on the Cipher text

This section shows the lattice-based attack on the cipher text, e , and the public key, h , to discover the plaintext by using the same example 4.2. for the original NTRU and the proposed algorithms. This done by using LLL algorithm to show the performance differences between each of them.

Example 4.2

Let we have the input parameters such as the following:

$$N=7, \quad p=3, \quad q=32, \quad f = -1 + X + X^4 + X^5 - X^6,$$

$$\text{and } g = X - X^2 - X^4 + X^6.$$

$$\text{The message } m = -1 + X - X^2 + X^3 - X^5 + X^6,$$

$$\text{The polynomial } r = 1 + X - X^2 - X^4.$$

The follows sections illustrate the lattice-based attack by using LLL algorithm on the cipher text.

4.3.2.1 The Attack on the Original NTRU

The public key that will resulted for the original NTRU cryptosystem for the above example (4.2) is:

$$h = 15 + 19x^1 + 24x^2 + 26x^3 + 5x^4 + 23x^5 + 16x^6,$$

and the ciphertext c:

$$c = 13 + 14x + 4x^2 + 16x^3 + 24x^4 + 14x^5 + 11x^6.$$

Firstly, construct the lattice basis, L, as array form according equation (2.16) for the above public key and the cipher text as the following:

Table (4.7): The lattice basis of L for the original NTRU

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
15	16	23	5	26	24	19	32	0	0	0	0	0	0	13
19	15	16	23	5	26	24	0	32	0	0	0	0	0	14
24	19	15	16	23	5	26	0	0	32	0	0	0	0	4
26	24	19	15	16	23	5	0	0	0	32	0	0	0	16
5	26	24	19	15	16	23	0	0	0	0	32	0	0	24

23	5	26	24	19	15	16	0	0	0	0	0	32	0	14
16	23	5	26	24	19	15	0	0	0	0	0	0	32	11
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Applying LLL algorithm to this public key, h , and the encrypted message, c , leads to the basis like the following:

Table (4.8): The LLL algorithm outputs

-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0
-1	-1	1	0	1	0	0	-1	1	-1	1	0	-1	1	1
1	0	-2	1	0	-1	0	1	2	1	-4	0	-2	2	-1
2	1	-3	1	0	-2	-1	-2	2	-2	-1	0	1	2	-1
-1	0	3	0	-2	1	2	0	-3	0	0	3	0	0	0
0	2	-1	0	0	1	-1	1	2	-2	-1	-3	1	2	-1
0	-3	0	2	-1	-2	1	3	0	0	-3	0	0	0	0
1	-1	0	2	-2	0	3	1	2	1	-1	0	-2	-1	-1
-1	1	2	1	1	0	-1	1	4	-2	-2	0	1	-2	-2
-2	-1	-1	-2	3	-1	1	0	-2	0	1	2	1	-2	-2
0	-1	3	-2	-1	-2	2	0	0	1	-1	-1	-2	3	-3
3	2	0	-3	1	-3	1	3	1	0	1	0	-3	-2	0
-3	1	3	2	0	-3	1	-3	-2	3	1	0	1	0	0
-2	1	1	-2	1	-1	2	1	2	0	-4	-1	1	1	2
1	0	-1	2	-2	0	0	8	3	3	5	5	3	5	2

Then, we search for the row vector in the above array whose 7 first rows that must contain the sequence $\{-1, 0, 1\}$ only. This holds for the first and second rows. By trying the next 7 entries, the second row vector contains the plaintext $(-1 \ 1 \ -1 \ 1 \ 0 \ -1 \ 1)$. This means the lattice-based attack succeed to discover the plaintext of the original NTRU.

4.3.2.2 The Attack on the Proposal One

In this section, we apply the LLL algorithm to the same parameters of example (4.2) for the proposed algorithm. If the LFSR contains the sequence (1 1 0 0 1 0 1) and after shifting it for N times will result the sequence (0 0 1 0 1 1 0), and then the public key will be as follows (algorithm 3.1):

$$h = 13 + 25x + 24x^2 + 14x^3 + 25x^4 + 19x^5 + 16x^6,$$

and when the polynomial $r = 1 + X - X^2 - X^4$, the message $m = -1 + X - X^2 + X^3 - X^5 + X^6$, will be encrypted (algorithm 3.2) as:

$$c = 7 + 10X + 20X^2 + 16X^3 + 24X^4 + 10X^5 + X^6.$$

Firstly, construct the lattice basis, L, as array form according to equation (2.16) for the above public key and the cipher text as the following:

Table (4.9): The lattice basis of L (proposal one)

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
13	16	19	25	14	24	25	32	0	0	0	0	0	0	7
25	13	16	19	25	14	24	0	32	0	0	0	0	0	10
24	25	13	16	19	25	14	0	0	32	0	0	0	0	20
14	24	25	13	16	19	25	0	0	0	32	0	0	0	16

25	14	24	25	13	16	19	0	0	0	0	32	0	0	24
19	25	14	24	25	13	16	0	0	0	0	0	32	0	10
16	19	25	14	24	25	13	0	0	0	0	0	0	32	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Applying LLL algorithm to the lattice basis, L , in table (4.10) leads to the basis like the following:

Table (4.10): The LLL algorithm outputs

-1	-1	0	0	3	-1	-1	-4	-1	1	-2	-3	2	-1	0
-1	-1	-4	1	1	4	0	-2	-2	2	-1	-1	1	3	0
0	0	3	-1	-1	-1	-1	1	-2	-3	2	-1	-4	-1	0
2	-2	-2	2	-3	3	-3	0	-1	0	-2	0	-2	1	1
2	-3	-2	-2	1	0	1	0	0	0	1	-2	2	-1	1
-1	-1	-1	-1	0	0	3	2	-1	-4	-1	1	-2	-3	0
-3	1	1	1	1	0	0	3	-2	1	4	1	-1	2	0
-1	1	0	0	0	-1	-1	0	2	2	-2	2	-3	-1	2
0	0	1	-1	-4	0	-1	-2	-1	-1	3	0	0	1	3
0	0	2	0	-2	0	0	3	4	0	2	-2	0	1	-1
0	-3	0	0	-1	0	-1	2	-2	0	-1	-2	2	1	-1
-2	-1	3	0	2	0	-1	0	-1	-2	-2	-1	1	-3	2
-1	3	-1	0	-1	1	1	1	-1	2	-3	-1	2	0	2
0	-2	-2	2	2	-1	3	2	2	-1	0	1	1	3	1
-1	1	-2	2	-1	0	-4	-3	1	-4	-2	-2	1	1	0

In the table 4.11 don't find any row vector as the form of the sequences $\{-1, 0, 1\}$ only, this means the proposed algorithm bypassing the lattice-based attack on the ciphertext by using the LLL algorithm.

4.3.2.3 The Attack on the Proposal Two

Also this section applying LLL algorithm to discover the plaintext for the same parameters of example (4.2), the public key for this proposal (algorithm 3.3) is:

$$h = 26 + 5X + 23X^2 + 15X^3 + 19X^4 + 24X^5 + 16X^6,$$

and the cipher text (algorithm 3.4), c is:

$$c = 13 + 14X + 4X^2 + 16X^3 + 24X^4 + 14X^5 + 11X^6$$

Firstly, construct the lattice basis, L, as array form according equation (2.16) with the above public key and the cipher text as the following:

Table (4.11): The lattice basis of L (proposal two)

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
26	16	24	19	15	23	5	32	0	0	0	0	0	0	13
5	26	16	24	19	15	23	0	32	0	0	0	0	0	14
23	5	26	16	24	19	15	0	0	32	0	0	0	0	4
15	23	5	26	16	24	19	0	0	0	32	0	0	0	16
19	15	23	5	26	16	24	0	0	0	0	32	0	0	24
24	19	15	23	5	26	16	0	0	0	0	0	32	0	14
16	24	19	15	23	5	26	0	0	0	0	0	0	32	11
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Applying a LLL algorithm to the lattice basis, L , table (4.12) leads to the basis like the following:

Table (4.12): The LLL algorithm outputs

1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
-1	-1	-1	-1	0	3	-1	-2	1	0	0	2	-1	0	-1
2	-1	0	-1	-1	-2	1	-1	2	2	0	0	1	-4	-2
1	-1	1	-2	0	2	2	0	-1	0	-1	1	2	-1	-4
0	2	-1	0	1	-2	-3	1	-2	1	0	1	2	-3	3
-1	0	-2	-2	-2	3	2	1	0	-4	1	1	0	1	0
3	-2	1	1	1	-3	-2	-1	-1	-2	0	1	3	0	-2
-1	2	-2	3	-1	-2	-1	0	1	-2	0	2	-2	1	-1
-2	0	3	-1	-2	1	0	1	1	-1	-1	-2	0	2	3
-3	0	1	-2	1	-1	2	0	-1	0	2	-2	0	1	2
0	3	-1	-1	-1	1	1	-2	1	-1	1	-1	0	2	-4
-1	0	-1	0	1	0	0	3	2	-1	-4	0	2	-2	-2
2	-1	0	-2	-2	-2	3	1	1	0	-4	1	1	0	0
2	-3	-2	1	0	2	2	-1	0	-1	-1	0	4	-1	0
-2	1	0	-1	1	0	1	3	6	6	2	6	5	4	-2

In the table (4.11) don't find any row vector as the form of the sequences $\{-1,0,1\}$ only, this means the proposed algorithm bypassing the lattice-based attack on the ciphertext by using the LLL algorithm.

4.4 Dynamically Generation of Keys Sequences

This section dedicated to show a method for generating long keys sequence dynamically (the polynomial h_i) that based on the previous public key, h_{i-1} . Without repeating any key block with this method then each block of plaintext will be encrypted by the specific public key that different from the other block.

Example 4.3

When the parameters are $(N, p, q) = (16, 3, 64)$ and

$$f = -1 + X - X^2 + X^3 - X^5 + X^6 + X^8 - X^9 + X^{10} + X^{13} - X^{15},$$

$$g = -X + X^7 - X^8 + X^9 + X^{13} - X^{15}.$$

The public key that generated by the original NTRU as follows:

$$h_0 = 42 + 17X + 40X^2 + 34X^3 + 26X^4 + 30X^5 + 46X^6 + 29X^7 + 15X^8 + 33X^9 + 57X^{10} + 35X^{11} + 45X^{12} + 48X^{13} + 46X^{14} + 33X^{15}.$$

To generate a sequences of keys dynamically, the first step is applying the proposed method (equation (3.3) $g = (h \bmod p) - 1$) on h_0 in order to generate the new polynomial g_1 (here $g_1 = -1 \ 1 \ 0 \ 0 \ 1 \ -1 \ 0 \ 1 \ -1 \ -1 \ -1 \ 1 \ -1 \ -1 \ 0 \ -1$) and then computing the new public key, h_1 , to use with the second block of data ($h = p \cdot f_q * g \pmod{q}$) as the following:

$$h_1 = 55 \ 59 \ 23 \ 47 \ 47 \ 43 \ 56 \ 27 \ 35 \ 60 \ 14 \ 33 \ 52 \ 33 \ 29 \ 1$$

and then doing the same steps to generate a long dynamic sequences of keys to cover all blocks (with length N) of the plaintext. The public keys below for first ten blocks of data:

$$h_2 = 11 \ 15 \ 2 \ 42 \ 39 \ 25 \ 11 \ 38 \ 0 \ 53 \ 15 \ 50 \ 26 \ 29 \ 42 \ 59,$$

$$h_3 = 7 \ 47 \ 33 \ 22 \ 51 \ 7 \ 39 \ 27 \ 52 \ 13 \ 17 \ 14 \ 18 \ 3 \ 39 \ 4,$$

$$h_4 = 48 \ 40 \ 49 \ 8 \ 45 \ 6 \ 41 \ 41 \ 44 \ 19 \ 5 \ 39 \ 49 \ 63 \ 33 \ 40,$$

$$h_5 = 5 \ 23 \ 13 \ 55 \ 35 \ 45 \ 49 \ 16 \ 40 \ 3 \ 36 \ 49 \ 51 \ 60 \ 27 \ 5,$$

$$h_6 = 37 \ 48 \ 27 \ 55 \ 56 \ 0 \ 17 \ 63 \ 0 \ 35 \ 11 \ 18 \ 29 \ 48 \ 12 \ 50,$$

$$h_7 = 23 \ 36 \ 53 \ 0 \ 61 \ 21 \ 49 \ 32 \ 41 \ 28 \ 11 \ 61 \ 3 \ 40 \ 18 \ 29,$$

$h_8 = 40 \ 32 \ 41 \ 1 \ 59 \ 46 \ 53 \ 54 \ 7 \ 1 \ 28 \ 3 \ 25 \ 52 \ 9 \ 0,$

$h_9 = 2 \ 1 \ 59 \ 56 \ 1 \ 25 \ 29 \ 58 \ 53 \ 36 \ 24 \ 46 \ 11 \ 57 \ 19 \ 35,$

$h_{10} = 61 \ 8 \ 51 \ 28 \ 45 \ 30 \ 13 \ 15 \ 8 \ 2 \ 33 \ 0 \ 17 \ 30 \ 40 \ 15,$

...

... etc. and so on, until encrypt all blocks of data.

The new keys, h_i , has no relationship between its values (previous and subsequent public keys) that may be exploited to attack.

4.5 Performance Analysis

The performance analysis of these proposals are presented in this section and compare the complexity of different computation with the original NTRU cryptosystem. The first proposal based on four parameters (N, X, p, q) where p and q are a prime relatively. The specifications of the original NTRU are defined with three parameters (N, p, q) . The plaintext blocks size is the same for the original NTRU and the proposed algorithms. The complexity of the proposals gives the same to the original NTRU:

Table (4.13): Performance analysis for basic NTRU and the proposals

Characteristics	The complexity [19]
Plaintext Block	$N \log_2 p$ bits
Encrypted Text Block	$N \log_2 q$ bits
Speed of Encryption	$O(N^2)$ operations
Speed of Decryption	$O(N^2)$ operations
Expansion of Message	$\log_p q$ to 1

Length of Private Key	$2N \log_2 p$ bits
Length of Public Key	$N \log_2 q$ bits
LLL algorithm Security	$2 \left(\frac{\pi^2 a e^2}{3\pi q^2!} \right)^{1/4}$ for NTRU, but the proposals are secure.

So, the modification for the proposal two can generate approximately (N^p-1) versions of key sequence with length N from the original public key, this means that we do not repeat the public key with each block, but each block is encrypted with a key that is completely different from the other keys that precedes it and there is no relationship between them that may be exploited by the attacker to break these keys.

4.5.1 Encryption Time and Decryption Time Analysis

The same parameters ($N=32$, $p=3$, $q=64$) for the original NTRU cryptosystem and the proposals are executing on the same laptop gives the encryption time and decryption time as the following:

Table (4.14): Encryption Time and Decryption Time Analysis for original NTRU and the proposed algorithms per second

Algorithm	Encryption Time	Decryption Time
NTRU	0.041	0.025
Proposal One	0.059	0.034
Proposal Two	0.051	0.029

So, the difference between them is limited.

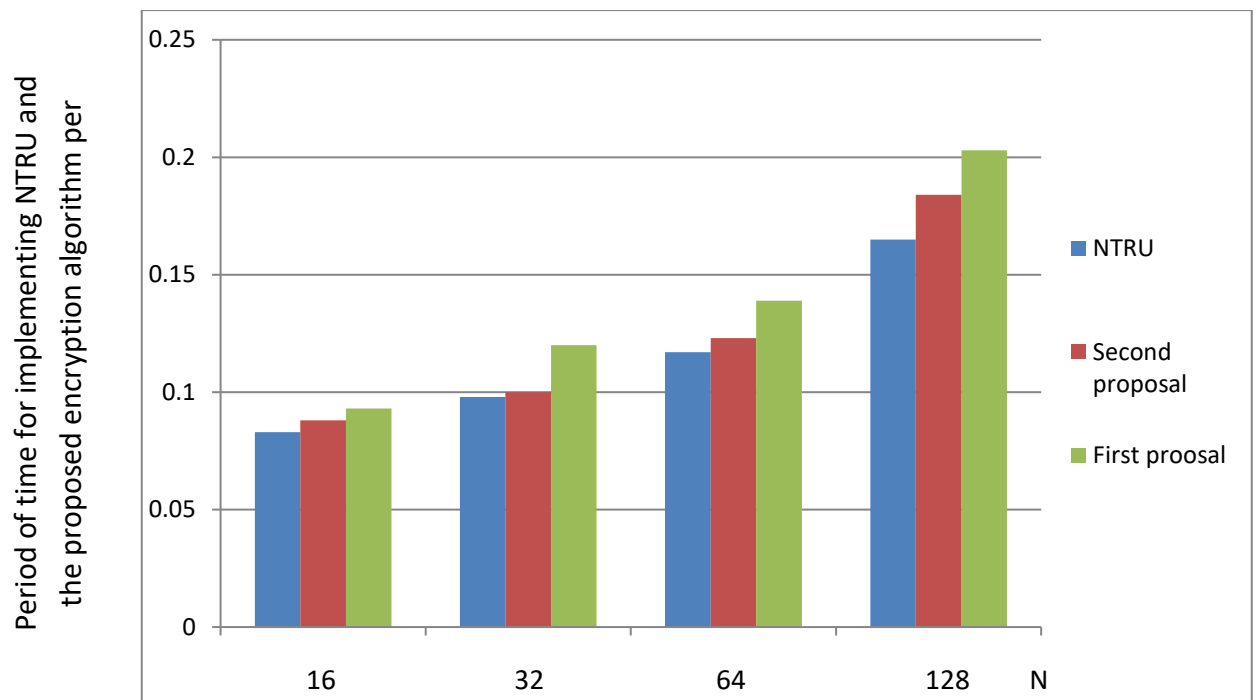


Figure (4.1): Performance timings of NTRU and the proposed algorithms

Figure 1. shows that there is no significant difference in time between the implementation of the original NTRU and the proposed algorithms.

Chapter Five

*Conclusions and Suggestions
for Future Works*

Chapter Five

Conclusions and Suggestions for Future Works

5.1 Introduction

This chapter presents some of conclusions and recommendations for future works. The conclusions from this work are presented in the section (5.2). While, suggestions for the future work are clarified in section (5.3).

5.2 Conclusions

Some conclusions can be inferred from the results and tests of these proposals as follows:

- 1- The original NTRU cryptosystem has weak points, including the ability to attack it by using LLL algorithm to discover either the original secret key, or an alternative secret key which is useful to decrypt the cipher texts under certain lengths.
- 2- Increasing length N leads to increase processing time and security level.
- 3- The proposed methods succeeded to prevent an attack it by LLL algorithm.
- 4- Long keys can be generated without duplication the same keys for all each block of data without the need for more private keys.
- 5- The row containing the private key or an alternate key can be found according to equation (4.1).

5.3 Suggestions for Future Works

Through this work and search, in order to find solutions for weakness in NTRU cryptosystem has been reached several points that could be of great value in the future, such as the following:

1. Finding method to use variable length of parameter N dynamically for each block in order to make security of NTRU is high.
2. The need for faster methods to find the inverse of truncated polynomial ring or computing the multiplication between polynomials in order to increase the speed of the NTRU algorithm.
3. Using independent public keys for each block of data for parallel processing (multiprocessor or Field-Programmable Gate Arrays (FPGA)).
4. Using NTRU modification to secure IoT applications data, because it has many features that make it suitable for IoT devices (constrained devices).

Publications

- [1] Omar Sapti Guma'a, Prof. Qasim Mohammed Hussein and Prof. Ziyad Tariq Mustafa, "Q-NTRU Cryptosystem for IoT Applications", Journal of Southwest Jiaotong University, 54 (4), pp. 1-12, 2019.
- [2] Omar Sapti Guma'a, Prof. Qasim Mohammed Hussein and Prof. Ziyad Tariq Mustafa, "Dynamic Keys Generation for Internet of Things", Indonesian Journal of Electrical Engineering and Computer Science, Vol 18, No 2: May 2020.

References

- [1] Bernstein, Daniel J. "Introduction to post-quantum cryptography." Post-quantum cryptography. Springer, Berlin, Heidelberg, 2009. 1-14.
- [2] Huang, Wei-Lun, Jiun-Peng Chen, and Bo-Yin Yang. "Correlation Power Analysis on NTRU Prime and Related Countermeasures." IACR Cryptol. ePrint Arch. 2019 (2019): 100.
- [3] Gaithuru, Juliet N., and Majid Bakhtiari. "Insight into the operation of NTRU and a comparative study of NTRU, RSA and ECC public key cryptosystems." 2014 8th. Malaysian Software Engineering Conference (MySEC). IEEE, 2014.
- [4] Lenstra, Arjen K., Hendrik Willem Lenstra, and László Lovász. "Factoring polynomials with rational coefficients." Mathematische annalen 261.ARTICLE (1982): 515-534.
- [5] Qasim Mohammed Hussein. "Recover the NTRU Private Keys from Known Public Information and Public Key." PhD thesis, University of Technology, Computer Science Department, 2009.

- [6] Malekian, Ehsan, and Ali Zakerolhosseini. "OTRU: A non-associative and high speed public key cryptosystem." 2010 15th CSI International Symposium on Computer Architecture and Digital Systems. IEEE, 2010.
- [7] Jarvis, Katherine, and Monica Nevins. "ETRU: NTRU over the Eisenstein Integers." *Designs, Codes and Cryptography* 74.1 (2015): 219-242.
- [8] Karbasi, Amir Hassani, and Reza Ebrahimi Atani. "ILTRU: An NTRU-Like Public Key Cryptosystem Over Ideal Lattices." *IACR Cryptol. ePrint Arch.* 2015 (2015): 549.
- [9] Majeed, A. A. "CQTRU Cryptosystem Based on Commutative Rings of Quaternion." Diss. Master Thesis, University of Technology, Baghdad, 2015.
- [10] Thakur, Khushboo, and B. P. Tripathi. "BTRU, A Rational Polynomial Analogue of NTRU Cryptosystem." *International Journal of Computer Applications, Foundation of Computer Science (FCS)*, NY, USA 145.12 (2016).
- [11] Wang, Baocang, Hao Lei, and Yupu Hu. "D-NTRU: More efficient and average-case IND-CPA secure NTRU variant." *Information Sciences* 438 (2018): 15-31.
- [12] Shuai, Li, et al. "A Group-Based NTRU-Like Public-Key Cryptosystem for IoT." *IEEE Access* 7 (2019): 75732-75740.
- [13] Ibrahim, Anas, et al. "NTRU-Like Random Congruential Public-Key Cryptosystem for Wireless Sensor Networks." (2020).

- [14] Shor, Peter W. "Algorithms for quantum computation: discrete logarithms and factoring." Proceedings 35th annual symposium on foundations of computer science. IEEE, 1994.
- [15] Shor, Peter W. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer." SIAM review 41.2 (1999): 303-332.
- [16] Ajtai, Miklós. "Generating hard instances of lattice problems." Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. 1996.
- [17] Regev, Oded. "On lattices, learning with errors, random linear codes, and cryptography." Journal of the ACM (JACM) 56.6 (2009): 1-40.
- [18] Ali Hussein Raheem. "An Integrated Security Protocol Communication Scheme for Internet of Things using the Locator/ID Separation Protocol Network." PhD thesis, Faculty of Science and Technology, Middlesex University, United Kingdom, P.P.34-36, 2017.
- [19] Wang, Chen, and Huaixi Wang. "A new ring signature scheme from NTRU lattice." 2012 Fourth International Conference on Computational and Information Sciences. IEEE, 2012.
- [20] Ward, Roy William, and Timothy Christopher Anthony Molteno. "Table of linear feedback shift registers." Electronics Group, University of Otago, 2012.
- [21] Mishra, Shivshankar, Ram Racksha Tripathi, and Devendra Kr Tripathi. "Implementation of configurable linear feedback shift register in VHDL." 2016 International Conference on Emerging

Trends in Electrical Electronics & Sustainable Energy Systems (ICETEESES). IEEE, 2016.

- [22] Monteverde, Mariano. "NTRU software implementation for constrained devices." Master's thesis, Katholieke Universiteit Leuven (2008).
- [23] Baldoni, M. Welleda, Ciro Ciliberto, and Giulia Maria Piacentini Cattaneo. "Elementary number theory." cryptography and codes. Berlin: Springer, 2009.
- [24] NTRU Cryptosystem Inc., "The NTRU Encryption Public Key (Basic Tutorial)", 2005. Available from <http://www.ntru.com/cryptolab>
- [25] Menezes, Alfred J., Paul C. Van Oorschot, and Scott A. Vanstone. "Handbook of Applied Cryptography CRC Press." Boca Raton (1997).
- [26] Kyurkchiev, Pavel, and Viktor Matanski. "The faster Euclidean algorithm for computing polynomial multiplicative inverse." Collection of scientific works from conference, Pamporovo, Bulgaria. 2018.
- [27] O'Rourke Colleen Marie. "Efficient NTRU Implementations". Master thesis, Worcester Polytechnic Institute, 2002.
- [28] Silverman, Joseph H. "Almost inverses and fast NTRU key creation." NTRU Cryptosystems, (Technical Note# 014) (1999).
- [29] Mersin, Ali. "The comparative performance analysis of lattice based NTRU cryptosystem with other asymmetrical cryptosystems." MS thesis. İzmir Institute of Technology, 2007.

- [30] Nguyen, Phong Q., and Jacques Stern. "Lattice reduction in cryptology: an update." *Algorithmic Number Theory: 4th International Symposium, ANTS-IV Leiden, The Netherlands, July 2-7, 2000 Proceedings*. Springer, 2006.
- [31] Pipher, Jill. "Lectures on the NTRU encryption algorithm and digital signature scheme: Grenoble, June 2002." Brown University, Providence RI 02912, Report (2002).
- [32] Hoffstein, Jeffrey, Jill Pipher, and Joseph H. Silverman. "NTRU: A ring-based public key cryptosystem." *International Algorithmic Number Theory Symposium*. Springer, Berlin, Heidelberg, 1998.
- [33] Malekian, Ehsan, Ali Zakerolhosseini, and Atefeh Mashatan. "QTRU: quaternionic version of the NTRU public-key cryptosystems." *ISecure-The ISC International Journal of Information Security* 3.1 (2011): 29-42.
- [34] Silverman, Joseph H. "An introduction to the theory of lattices and applications to cryptography." *Computational Number Theory and Applications to Cryptography*, University of Wyoming (2006): 1-212.
- [35] Linder Richard. "Current Attack On NTRU." master thesis, university of Darmstadt, 2006.
- [36] Hoffstein J., Silverman J., "Optimizations for NTRU." *proceeding of Public-Key Cryptography and Computational Number Theory*, Warsaw, September, 2000.
- [37] Gentry, Craig. "Key recovery and message attacks on NTRU-composite." *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, 2001.

- [38] Howgrave-Graham, Nick, Joseph H. Silverman, and William Whyte. A Meet-in-the-Middle Attack on an NTRU Private key. Vol. 4. Technical report, NTRU Cryptosystems, June 2003. Report, 2003.
- [39] Silverman J. "Invertibility in Truncated Polynomial Rings." NTRU Technical Report #009. 1998.
- [40] Hoffstein, Jeffrey, and Joseph H. Silverman. "Random small Hamming weight products with applications to cryptography." Discrete Applied Mathematics 130.1 (2003): 37-49.
- [41] Hoffstein, J., J. Pipher, and J. H. Silverman. "An Introduction to Mathematical Cryptography, UTM." (2008).
- [42] Tata, Praveen Gauravaram, Harika Narumanchi, and Nitesh Emmadi. "Analytical study of implementation issues of NTRU." 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, 2014.
- [43] Nguyen, Hien Ba. "An overview of the NTRU cryptographic system." Diss. San Diego State University, 2014.
- [44] Rahma, Abdul Monem S., and Qasim Mohammed Hussein. "A New Attack on NTRU Public Key Cryptosystem Depend on Using Public Key and Public Information." Engineering and Technology Journal 28.6 (2010): 1061-1072.
- [45] Howgrave-Graham, Nick. "A hybrid lattice-reduction and meet-in-the-middle attack against NTRU." Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 2007.

- [46] Coppersmith, Don, and Adi Shamir. "Lattice attacks on NTRU." International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1997.
- [47] Nguên, Phong Q., and Damien Stehlé. "Floating-point LLL revisited." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2005.
- [48] Nguyen, Phong Q., and Brigitte Vallée. "The LLL algorithm." Springer Berlin Heidelberg, 2010.
- [49] Thiemann, René, and Akihisa Yamada. "Formalizing Jordan normal forms in Isabelle/HOL." Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs. 2016.
- [50] Rosenberg, Daniel. "NTRUencrypt and lattice attacks." Diss. PhD thesis, Royal Institute of Technology, 2010.
- [51] Phong Q. Nguyen and Brigitte Valle. "The LLL Algorithm: Survey and Applications." Springer Publishing Company, Incorporated, 1st edition, 2009.

المخلص

إن وجود أجهزة الكمبيوتر الكمومية يجعل تنفيذ الخوارزميات التي تتطلب عمليات حسابية عالية أمراً ممكناً للغاية ، مما يجعل العديد من أنظمة التشفير المستخدمة حالياً قابلة للكسر بتنفيذ هذه الخوارزميات. طور بيتر شور (Peter Shor) خوارزمية كمومية تسمى خوارزمية شور في عام 1994 ، واعتبرت اكتشافاً مهماً في هذا المجال. لذلك. يمكن كسر العديد من أنظمة التشفير الحالية مثل RSA و ECC باستخدام كمبيوتر كمي.

وبالتالي ، هناك حاجة إلى مخططات تشفير جديدة تقاوم الكم. يجب أن تكون هذه الإنشاءات المشفرة الجديدة فعالة وآمنة بدرجة كافية لاستخدامها في الممارسة العملية وقياسية لعدد كبير من السنوات ، لتحل محل الهياكل الحالية إذا لزم الأمر.

يعتبر نظام تشفير المفتاح العام القائم على الشبكة (LB-PKC) أحد الحلول المقدمة للتغلب على هذا التحدي. NTRU هي واحدة من LB-PKCs التي تعتمد على الحلقة متعددة الحدود المقطوعة $(x^N - 1) / Z[x]$ ، ولكن ، تم مهاجمة نظام NTRU باستخدام خوارزمية LLL في ظل ظروف معينة ، والتي يمكن أن تجد المفتاح السري عندما يكون طول المفتاح العام أقل من 127.

الهدف من الأطروحة هو إجراء تعديلات على نظام التشفير الأصلي NTRU لضمان فشل هجوم LLL عليه. يمكن تقسيم هذه الأطروحة إلى جزأين: أولاً ، تقدم طريقة لإضافة معلمة جديدة إلى معلمات NTRU الأصلية ، وتعتمد قيم هذه المعلمة على سجلات تحول التغذية المرتدة الخطية (LFSR). ثانياً ، يقدم تعديلاً على NTRU الأصلي عن طريق إجراء عملية مبادلة بين قيم المفتاح العام. بالإضافة إلى توليد سلاسل مفاتيح طويلة ديناميكياً لاستخدامها في التشفير.

أظهرت النتائج التجريبية على الطرق المقترحة قدرة هذه الطرق على المقاومة ضد هجوم خوارزمية LLL ، حتى إذا كان طول المفتاح العام لا يتجاوز 11 ، كما يمكن لهذه المقترحات أن تولد تقريباً (N^{p-1}) من تسلسل المفاتيح بطول N .



جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة ديالى / كلية العلوم / قسم علوم الحاسبات



تعديل $NTRU$ ضد هجوم LLL

رسالة مقدمة

الى/ جامعة ديالى /كلية العلوم / قسم علوم الحاسبات كجزء من متطلبات نيل شهادة
الماجستير في علوم الحاسبات.

من قبل

عمر سبتي جمعة

بإشراف

أ.د. قاسم محمد حسين الشمري

أ.د. زياد طارق مصطفى الطائي