Abstract ---

The NTRU [Number theory research unit] cryptosystem is a relatively new public key cryptographic algorithm that was first introduced in 1998, and that key runs are much faster than conventional public key algorithms such as RSA, ECC. The main advantage of this cryptosystem is its high speed generation keys, which is often the most important part of public key cryptography.

The security of NTRU cryptosystem comes from the interaction of the polynomial mixing system with the independence of reduction modulo two relatively prime integers' p and q.

This thesis introduces the concepts behind NTRU as a new public key cryptosystem. NTRU features are reasonably short, easily created keys, high speed, and low memory requirement. These features make it favourable in mobile communication systems, broadcast and satellite channels for its low cost hardware needs.

NTRU encryption and decryption use a mixing system suggested by polynomial algebra combined with a clustering principle based on elementary probability theory.

Also an approach variant of the NTRU public key cryptosystem called Matrix NTRU cryptosystem is proposed and has been shown to be much faster and have higher efficiency than the classical NTRU cryptosystem.

The thesis describes the NTRU Signature Scheme with enhanced document encoding, signature, verification, with provision of documented algorithms and examples.

The test and performance analysis performed using a PC with the following specification (processor 1.7 dual cores,memory 512 MB with windows XP-SP2 operating system), and all programs are developed in Visual Basic. .

Acknowledgements ----------------------------

Thanks God for everything

I would like to express my deepest gratitude to my supervisors Dr.Abdul munem S.Rama and Dr.Akram M.Al-Abood for their support, encouragement and patience throughout this thesis.

I am gratefully to the head of department and the staff members of the department of Mathematical & Computer Applications, whom gave me all help facilities during my work.

Special thanks to my family members for their love and their support in every stage of my life.

I would like to thank all friends and especially to my friend in Al-Nahrain university and the university of technology whom have supported me in any way during the creation period of this thesis.

Finally, Special thanks to Mr. Kasim AlShmmary for his faithful help.

Ayad Hazim Ibrahim 2008

Supervisor certification ---------------------

We certify that this thesis was prepared under our supervision at the Department of Mathematics and Computer Applications, College of Science, Al-Nahrain University as a partial fulfillment of the requirements for the degree of Master of Science in mathematics.

Signature:	**Signature:**
Name: Dr. Abdul Munem S. Rahma	**Name: Dr Akram M. Al-Abood**
Date: / / 2008	**Date: / / 2008**

In view of the available recommendations, I forward this thesis for debate by the examining committee.

Signature:

Name: Assist. Prof. Dr Akram M. Al-Abood

Head of the Department

Date: / / 2008

We certify that we have read this thesis entitled **“Cryptosystem approach using modified NTRU”** and as examining committee examined the student (Ayad Hazim Ibrahim) in its contents and in what it connected with, and that is in our opinion it meets the standards of a thesis for the degree of Master of Science in Mathematics.

(Chairman)	***(Member)***
Signature:	Signature:
Name: Dr. Amir S. Al-Malah	Name: Dr. Dr. Radhi Ali Zboon
Date: / / 2008	Date: / / 2008
(Member)	***(Member and Supervisor)***
Signature:	Signature:
Name: Dr. Hala B. Abdul Wahab	Name: Dr.Abdul Munem S. Rahma
Date: / / 2008	Date: / / 2008
(Member and Supervisor)	
Signature:	
Name: Dr. Akram M. Al-Abood	
Date: / / 2008	

Approved by the Collage of Science

Signature

Name: Dr. LAITH ABDUL AZIZ AL-ANI

Dean of the Collage of Science

Date: / / 2008

1.1Introduction:

In this chapter we shall describe the public key cryptography, mathematical basic material and lattice that use for NTRU cryptosystem.

1.2 Public-Key Cryptography:

The idea of symmetric or private key, cryptosystems is that the communicating parties agree on common keys which they use to encrypt their messages to each other. The same key is used to decrypt, i.e. to unscramble the encrypted message. Eavesdroppers, who can capture the exchanged encrypted messages, are unable to understand the messages as long as the key used remains secret.

One of the weak points of these private key systems is key agreement. Clearly, before the parties are able to communicate securely, they must have a way to agree on their key. But how this is possible if all exchanged messages can be eavesdropped by adversaries. The development of complexity theory since the 50’s has made it possible to solve the problem in a revolutionary way. The main idea was to split the key, one public key for encryption and one private key for decryption. For instance Party *A* could then encrypt his/her messages to party *B* by using *B*’s public encryption key. As *B*’s decryption key is private, only *B* can decrypt the messages. In theory, the knowledge of the encryption key is sufficient to determine the decryption key. However, the system can be constructed in such a way, that the amount of work required for cryptanalysis is beyond the scope of any realistic adversary.

For a long time, the speed of the best symmetric cryptosystems was superior to all suggested public key cryptosystems. Hence it was not sensible to encrypt large amounts of data with public key systems. Instead, one used public key systems to exchange a key for a fast symmetric system. With the development of computers and algorithms, the public key cryptosystems have been used more and more.

To explain the ideas of complexity theory we must introduce some definitions.

Definition 1.1: [44]

Let **P**$=O(f(n))$ be the set of all polynomials comprises all positive functions g(n) for which there exists constants n_0 and *c* such that $g(n) \leq cf(n)$ when $n > n_0$, where **P** is called also the set of problems that can be solved in polynomial time using a deterministi*c* algorithm. That is, those problems for which there is an integer *k* such that all instances of size *n* of the problem can be solved in time $O(n^k)$, such problems are called tractable; all other problems are called intractable.

Definition 1.2: [44]

Problems that have a polynomial time non-deterministic algorithm constitute the set **NP**. That is, for every problem in **NP** we can check in polynomial time whether a given candidate is a solution to the problem. $P \subseteq NP$ but it is not known whether $P = NP$.

Definition 1.3: [44]

A problem in **NP** is called **NP**-complete if finding a polynomial time algorithm for that problem would mean that there is a polynomial algorithm for all problems in **NP** and thus we would have **P** = **NP**. These are clearly the hardest problems in **NP**.

The idea of **NP**-completeness was introduced by Cook in 1971 [40].

1.2.1 The one way function

Definition 1.4: [44]

A function f is called one-way if f(x) can be efficiently computed when *x* is given.

The one-way functions is the main building blocks of public key cryptosystems, informally, but no efficient algorithm exists to find *x* such that f(x) = *y* when y is given. In other words, the problem of computing f(x) is in **P**, but the problem of inverting f is in **NP** \ **P**.

Despite several decades of intense research, it is not known whether one-way functions exist or not. But there are some good candidates which are widely used in practice. As there does not exist a proof of one wryness, it is possible that somebody someday will find a fast algorithm to invert them. One-way functions cannot be used as cryptosystems: it is impossible to decrypt. An additional property is needed. A trapdoor one-way function is a one-way function which can be inverted in polynomial time if some additional information, the trapdoor, is known. These functions suit public key systems perfectly: the trapdoor acts as the private decryption key. For more detailed definitions consult [33].

The first proposal [36] for a public key cryptosystem by Merkle and Hellman in 1978 was based on a knapsack problem. This problem is **NP**-complete so, at first glance, a cryptosystem based on it seemed sufficiently difficult to break. Unfortunately **NP**-completeness only means that the hardest instances of the problem are difficult. It turned out that on average the knapsack problem was relatively easy, which made it unsuitable for cryptographic applications.

The earliest public key system still in use is the RSA cryptosystem developed by Ronald Rivest, Adi Shamir and Leonard Adleman [36] the underlying hard problem in RSA is the integer factorization problem. It is relatively easy to multiply two large integers; even the “school algorithm” is efficient enough, as it works in time O (n^2). On the other hand, it is very difficult to determine the factorization of a large integer, especially if it is a product of two large prime numbers. Integers with 300 digits are well beyond the best factorization algorithms known today. Although the factoring problem is not known to be an **NP**-complete problem, it is commonly considered hard enough for cryptographic purposes.

Another popular candidate one-way function is modular exponentiation. Given a, e and n, the value a_e mod n can be computed in time O (n^3). For the inverse problem, also known as the discrete logarithm problem, no polynomial time algorithm is known. The famous El-Gamal cryptosystem is based on this fact [42]. The same system can also be applied in the elliptic curve group [9].
From a theoretical point of view it would be perfect if breaking a cryptosystem required solving an **NP**-complete problem. However, for practical reasons, all instances of the problem that may occur in cryptanalysis should be hard. Therefore a minimal requirement is that the problem in question is hard on average. In [28] Miklos Ajtai showed that certain lattice problems are hard on average, provided they are hard in the worst case. As a consequence, it seems a good idea to base a public key system on such problems.

The NTRU encryption system [12] from 1998 is based on the difficulty of finding a short vector in a lattice, or alternatively the closest lattice vector to a given vector. The main advantage of NTRU is its speed; which is comparable to the fastest symmetric systems.

In addition to ordinary message encryption, public key cryptography also has other applications. From a practical point of view, perhaps the most important one is the ability to sign digital documents. Some properties required from a digital signature are impossible to be fulfilled, or require a so-called trusted third party using symmetric cryptosystems.

The idea is that each signer again creates a pair of keys: one private key needed to sign documents, and one public key needed to verify the validity of signatures. There are signature protocols related to most public key cryptosystems, including RSA, El Gamal and NTRU; see table [1].

The main advantage of NTRU over other public key cryptosystems is its speed: it is comparable to the fastest symmetric cryptosystems available. As an example, consider table (1.1) taken from the official NTRU Cryptosystems website [46].

	NTRU 251	RSA 1024	ECC 163
Public key (bits)	2008	1024	164
secret key (bits)	251	1024	163
plaintext (block)	160	702	163
Cipher text (block)	2008	1024	163
Encryption speed(block/sec)	22727	1280	458
Decrypt speed (block/sec)	10869	110	702

Table (1.1) level comparison between NTRU, RSA and ECC

1.3 Mathematical Background for the NTRU Cryptosystem:

In this section we tried to collect some of basic material on the integer set, abstract algebra, polynomial ring and lattice.

1.3.1 Integer set:

The set of integers {…,-3,-2,-1, 0, 1, 2, 3 …} is denoted by the symbol Z.

Definition 1.5: [abstract]

Let a,b be integers , with $a \neq 0$. Then a divides b if there exists an integer c such that b=ac, if a divides b, then this is denoted by a/b .

Theorem1:-[5]

If $a, b \in Z$ with $b > 0$, then there exist unique integers q and r such that $a = qb + r, \quad 0 \leq r < b$.

Corollary 1:- **(Division Algorithm)** [5].

If $a, b \in Z$, with $b \neq 0$, then there exist unique integers q and r such that $a = qr + b, \ \ 0 \leq r < |b|$.

Moreover, q and r are unique the reminder of the division is denoted by a mod b, and the quotient is denoted by a div b.

Definition1.6: [5]

If $a, b \in Z$, we say that an integer d is common divisor of a and b if $d|a$ and $d|b$.

Definition1.7: [5]

Let a and b be an integers, not both of which are zero.

The greatest common divisor of a and b, denoted by $\gcd(a,b)$ is a positive integer d such that

1- d is common divisor

2- Whenever $c|a \text{ and } c|b, \text{ then } c|d.$

It's to observed that if a=b=0 , then every integers is a common divisor of a and b. The notion of greatest common divisor of a and make sense only when we require that at least one of a and b different from zero, with the expectation $\gcd(0,0)=0$.

<u>Definition1.8</u>: [5]

An integer p ≥2 is said to be prime if it's only positive divisors are 1 and p .other wise p is called composite.

<u>Definition1.9:</u>

For $N \geq 1$, let $\phi(n)$ denote the number of integers in the interval[1,n] which are relatively prime to n . The function ϕ is called the Euler phi function .

The Euler phi function is a map $\phi: N \to N$ given by

$$\phi(n)=\{m\in N|1\leq m\leq N \text{ and } \gcd(m,n)=1\}$$

<u>Definition 1.10</u>: [5]

For an arbitrary integer a, let [a] denote the set of all integers congruent to a modulo n:

$$[a]=\{x\in Z|x\equiv a\,(\bmod n)\}$$
$$=\{x\in Z|x=a+kn \text{ for some integer } k\}.$$

[a] is called the congruence class ,modulo n ,determined by a and refer to a as a representative of this class .

By way of illustration, that we are dealing with congruence modulo 3.then

$$[0]=\{x\in Z|x=3k \text{ for some } k\in Z]$$
$$=\{\ldots,-9,-6,-3,0,3,6,9,\ldots\}.$$

Also

$$[1]=\{x\in Z|x=1+3k \text{ for some } k\in Z\}$$
$$=\{\ldots,-8,-5,-2,1,4,7,10,\ldots\}.$$

Similarly

To return to the general case of congruence modulo n, let

$$Z_n=\{[0],[1],[2],\ldots,[n-1].$$

Several properties of the collection Z_n that we shall require later appear in the next theorem

Theorem 2:- **[5]**

Let n be a positive integer and Z_n be defined above then

1- for each $[a]\in Zn$, $[a]\neq\varnothing$

2- if $a\in Z_n$ and $b\in[a]$ then $[b]=[a]$; that is , any element f the congruent class [a] determine the class .

3-for any $[a],[b]\in Z_n$ *where* $[a]\neq[b],[a]\cap[b]=\varnothing$

4-$\cup\{[a]|[a]\in Z_n\}=Z$

Conclusion remark (1) (Properties of congruence)

(1) $a\equiv b$(mod n) if and only if a and b leave the same reminder when divided by n,

(2) $a\equiv a$(mod n) (reflexivity) .

(3)If $a\equiv b$ (mod n) then $b\equiv a$ (mod (n), (symmetric).

(4)If $a\equiv b$(mod n) and $b\equiv c$(mod n), then $a\equiv c$(mod n) ,(transitivity) .

(5)If $a\equiv a_1$(mod n) and $b\equiv b_1$mod(n), then $a+b\equiv a_1+b_1$(mod n), and $a*b\equiv a_1*b_1$(mod n).

The equivalence class of an integer is the set of all integers congruent to A modulo n . From properties (2), (3), and (4) above, it can be seen that for a fixed. The relation of congruence modulo n. Partitions the set *Z* into equivalence classes. Now, if $a=q_n+r$, where $0\leq r< n$, then $a\equiv r(\text{mod } n)$. Hence each integer a is congruent modulo thus a and r are in the same equivalence class, and so r may simply be used to represent this equivalence class.

<u>Definition 1.11</u>: [5]

The integers modulo n ,denote Z_n , is the set of(equivalence classes of) integers{0,1,2,…,n-1}.Addition, subtraction and multiplication in Z_n are performed modulo n.

<u>Definition 1.12</u>: [5]

Let $a\in Z_n$. The multiplicative inverse of a modulo n is an integer $x\in Z_n$ such that $ax\equiv 1(\text{mod } n)$ .if such an x exists, then it is unique and a is said to be invertible, or a unit, the inverse of a is denoted by a^{-1}.

<u>Definition 1.13</u>: [5]

Let a, $b\in Z_n$ .Division of a by b modulo n is the product of a and b^{-1} modulo n, and is only defined if b is invertible modulo n.

<u>Conclusion Remark (2)</u>:

(1)If $a\in Z_n$. then a is invertible if and only if gcd(a,n)=1 .

(2)Let d=gcd(a,n) .The congruence equation $ax\equiv b(\text{mod } n)$ has solution x if and only if d divided b , in which case there are exactly d solution between 0 and n-1, these solutions are all congruent modulo n/d .

(3) (Chinese remainder theorem, CRT) If the integers $n_1,n_2,\ldots.n_k$ are pair wise relatively prime , then the system of simultaneous congruence [30]

$$\begin{gathered} x \equiv a_1 (\bmod n_1) \\ x \equiv a_2 (\bmod n_2) \\ . \\ . \\ . \\ x \equiv a_k (\bmod n_k) \end{gathered}$$

Has a unique solution modulo $n=n_1,n_2,\dots n_k$.

1.3.2 Binary Operation:

Definition 1.14:

A Binary operation * on a set S is a mapping from S $*$ S to S. that is,* is a rule which assigns to each ordered pair of elements from S an element of S.

Definition 1.15: **[5]**

A group is a pair (G,*) consisting of a non empty set G and a binary operation * defined on G, satisfying the requirements

(1) G is closed under the operation *.

(2) The group operation is associative. That is a*(b*c) = (a*b)*c for a,b,c$\in$ G.

(3) G contains an identity element e for a group operation * , and

(4) Each element a of g has an inverse $a^{-1} \in G$ to*.

The group G is commutative if a*b=b*a for all a, b$\in$ G.

Note that multiplicative group notation has been used for the group operation .if the group operation is addition , then the group is said to be additive group , the identity element is denoted by 0, and the inverse of a is denoted –a .

Example (1.1):

(Z_4, $+_4$) is a commutative group

$+_4$	[o]	[1]	[2]	[3]
[0]	[0]	[1]	[2]	[3]
[1]	[1]	[2]	[3]	[0]
[2]	[2]	[3]	[0]	[1]
[3]	[3]	[0]	[1]	[2]

Definition 1.16: [5]

A group G is finite if $|G|$ is finite .The number of elements in a finite group represents its order.

Definition 1.17: [5]

A group G is cyclic with generator a if G=(a) for some a $\in$ G

Thus to say that a group is cyclic means that each of its member can be expressed as an integral power of some fixed element of group. Any cyclic group G= (a) is commutative since

$$a^n a^m = a^{n+m} = a^{m+n} = a^m a^n$$

For arbitrary integer n and m.

Definition 1.18: [5]

A ring is an order triple(R,+,*) consisting of a non empty set R and two binary operation + ,* defined on R such that

1- (R, +) is a commutative group

2- (R,*) is a semi group

3- The operation * is distributive over the operation +. That is

a * (b+c)= (a * b) + (a * c) and (b+c) * a = (b * a) + (c * a) for all a, b, c $\in$ R.

Definition1.19: [5]

An element a of a commutative ring with identity R called a unit or an invertible element if there is an element b $\in$ R such that a * b =e.

- The set of units in a ring R form a group under multiplication called the group of units of R.

Definition 1.20 [5]

Let $R \text{ and } R'$ be two rings .By a ring homomorphism from $R \text{ into } R'$ is meant a function $f : R \to R'$ such that

$$f(a+b) = f(a) + f(b), \quad f(a*b) = f(a)*f(b)$$

For every pairs of elements a,b$\in$ R . A homomorphism that is also is a one- to-one mapping is called an isomorphism

Definitin1.21: [5]

A ring F is said to be field provided that the set F-{0} is a commutative group under the multiplication of F (the identity of this group will be written as 1) .

Conclusion Remark (3):

R is a field (under the useful operation of addition and multiplication mod n) if and only if n is prime number.

1.3.3 (Polynomial Rings):

Definition 1.22 [5]

If R is commutative ring, then a polynomial in the indeterminate x over the ring R is an expression of the form

$$P(x) = p_n x^n + \dots\dots + p_2 x^2 + p_1 x + p_0$$

Where each $p_i \in R$ and $n \geq 0$. The element p_i is called the coefficient of x^i in P(x). The largest integer m for which $p_m \neq 0$. Is called the degree of P(x) denoted deg (P(x)), p_m is called the leading coefficient of P(x). if P(x) = p_0 (constant poly) and $p_0 \neq 0$, then P(x) has degree 0. If all coefficient of P (x) are 0, then P(x) is called the zero polynomial and its degree. For mathematical convenience is defined to be ∞. The polynomial f(x) is said to be monic if its leading coefficient is equal to 1 [8].

<u>Definition 1.23</u>: [13]

If R is commutative ring, the polynomial ring R(x) is the ring formed by the set of all polynomials in the indeterminate * having coefficients form R. the two operations are the standard polynomial addition and multiplication, with coefficient arithmetic performed in the ring R.

<u>Definition 1.24</u>: [8]

Let f(x) $\in$ P(x) be a polynomial of degree at least 1. Then f(x) is to be irreducible over P if it cannot be written as the product of two polynomials in P(x), each of positive degree.

<u>Theorem (2)</u>: (Division algorithm for polynomials) [5]

Let R be a commutative ring with identity and f(x) ,g(x) $\neq 0$be a polynomials in R[x] ,with leading coefficient of g(x) an invertible element . The unique polynomial q(x), r(x) $\in$ R[x] exist such that

$$f(x) = q(x)g(x) + r(x)$$

Where either $r(x) = 0$ or $\deg r(x) < \deg g(x)$.

Moreover, q(x) and r(x) are unique. The polynomial q(x) is called the quotient, while r(x) is called the reminder. The reminder of the division is some terms denoted g(x) mod f(x), and the quotient is some times denoted g(x) mod f(x).

<u>Definition 1.25</u>:

If g(x), f(x) $\in$ P(x) then f(x) divides g(x), written h(x) $\mid$ g(x). If g(x) mod f(x) = 0.

Let h(x) be a fixed in polynomial in F(x). as with the integers one can define congruencies of polynomial in F(x) based on division by h(x).

<u>Definition 1.26</u>:

If g(x), a(x) $\in$ P(x), then g(x) is said to be congruent to f(x) modulo h(x) if h(x) divides g(x) - f(x). This is denoted by g(x) =f(x) (mod h(x)).

<u>Conclusion Remark</u> (4): (properties of congruence)

For all g(x), f(x), g1(x), h_1(x), S(x) $\in$ P(x), the following are true

1. g(x)$\equiv$f(x) (mod h(x)) if and only if g(x) and f(x)leave the same reminder upon division by h(x).
2. (Reflexivity) g(x) $\equiv$ g(x) (mod h(x)).
3. (Symmetry) if g(x) $\equiv$ f(x) (mod h(x)), then f(x) $\equiv$g(x) (mod h(x)).
4. (Transitivity) if g(x) $\equiv$f(x) (mod h(x)) and f(x) $\equiv$S(x) (mod h(x)), then g(x) $\equiv$S(x) (mod h(x)).
5. If g(x)$\equiv$$g_1$(x) (mod g(x)) and f(x)$\equiv$f(x) (mod f(x)) then g(x)+f(x)=g_1(x)+f_1(x) (mod h(x)) and g(x) f(x)=g_1(x)f (x) (mod h(x)).

   Let f(x) be a fixed polynomial in F(x). the equivalence class of a polynomial g(x) $\in$ P(x) is the set of polynomials in P(x) congruent to g(x) modulo h(x), from properties 2 and 3 and 4 above, it can be seen that the relation of congruence modulo h(x) partitions P(x) into equivalence classes. If g(x)$\in$P(x), then long division by h(x) yields unique polynomials g(x), r(x) $\in$P(x). Such that g(x) =g(x) H(x)+r(x) where deg r(x)< deg h(x) . Hence every polynomial g(x) is congruent nodule h(x) to a unique polynomial of degree less then deg h(x), the polynomial r(x) will be used as representative of the equivalence class of polynomial containing g(x).

<u>Definition 1.27</u>: (Finite Fields)

A finite field is a field F(x) which contains a finite number of elements. The order of F(x) is the number of elements in F(x).

<u>Conclusion Remark (5)</u> (existence and uniqueness of finite fields)

1. If F(x) is a finite field then F (x) contains p^m elements for some prime p and integer $m \geq 1$.
2. For every prime power order p^m , there is a unique finite field of order p^m. This field is denoted by $F p^m$, or some times by GF (p^m).

In formally two fields are isomorphic if they are structurally the same, although the representation of their field elements may be different. Note that if p is a prime then p is a field, and hence every field of order p is isomorphic to p. unless other write stated, the finite field F_p will hence forth be identified with P [6].

<u>Definition1.28</u>:

The non-zero element of F_q form a group under multiplication called the multiplicative group of F_q, denoted F_q^*.

<u>Definition 1.29</u>:

A generator of the cyclic group F_q^* is called primitive element or generator of F_q.

1.3.3.1the Euclidean algorithm for polynomials

$Z_P[x]$ be the finite field of order p. the theory of generator common divisors and the Euclidean algorithm for integers carries over in a straight forward manner to the poly ring $Z_P[x]$ (and more generally to the polynomial ring F(x), where F(x) is any field).

<u>Definition 1.30</u>: [35]

Let g(x), h(x) $\in Z_P[x]$, where not both are 0. Then the greatest common divisor of g(x) and h(x) , denoted gcd(g(x), h(x)) is the monic polynomial of greatest degree in $Z_p[x]$ which divides both g(x) and h(x). By definition, gcd (0, 0) =0.

<u>Conclusion Remark</u> (6)

$Z_P[x]$ is a unique factorization domain. That is every non-zero polynomial f(x) $\in$ $Z_P[x]$ has factorization.

$$f(x) = af_1(x)^{e_1}\ f_2(x)^{e_2} \ldots\ldots\ldots\ldots f_k(x)^{e_k}$$

where the f_i (x) are distinct monic irreducible poly in $Z_P[x]$. The e_i are positive integers, and a $\in$ $Z_P[x]$. furthermore, the factorization is unique up to rearrangement of factors.

<u>1.3.4 Modular Arithmetic</u>:-

Modular arithmetic is simply division with reminder, where you keep the reminder and throw every think else away. for example the expression 141 (modulo 13) means to divide 141 by 13 and keep the reminder now 141 divided by 13 gives a quotient of 10 and a reminder of 11 (141 = 10*13+11), so 141 (module 13) is equal to 11. This is written as an equality called congruence 141=11 (module 13)

In general, the expression a (module m) means to divide a by m and keep the remind, similarly a congruence a=b (modulo m).
Simply means that a and b leave the same reminder when they are divided by m. this is the same as saying that the difference a-b is a multiple of m. the integer m is called the modulus of the congruence.
Numbers and congruence with the same modulus may be added subtracted, and multiplied just as is done with ordinary equation, for example
q (modulo 17)+5(modulo 17)=14(modulo 17), and

q (modulo 17)*5(modulo 17)=45(modulo 17)
=11(modulo 17)

if a and m have no common factors then is also possible to find an inverse for a (module m) , that is to find an integer b so that a*b= 1(modulo m).
For example, the inverse of 7(modulo 17) is 5, since 7*5=35=1 (modulo 17). There is a very fast algorithm called Euclidean algorithm, which can be used to check if a and m have common factors and also to compute the inverse of a (modulo m) , if they do not have common factors.

1.3.5 Truncated polynomials ring:

The principal objects used by the NTRU encrypt public key cryptosystem are polynomials of degree N-1 having integer coefficients:

$$P(x) = p_0 + p_1 x + p_2 x^2 +,...,+ p_{N-2} x^{N-2} + p_{N-1} x^{N-1}$$

The coefficients $a_0, a_1, ..., a_{N-1}$ are integers some of the coefficients are allowed to be 0.

The set of all such polynomials is denoted by R. the polynomials in R are added together in the usual way be simply adding their coefficients:

$$a(x) + b(x) = (a_0 + b_0) + (a_1 + b_1)x +,...,+ (a_{N-1} + b_{N-1})x^{N-1}.$$

They are also multiplied in almost the useful manner, with one change. After doing the multiplication the power x^N should be replaced by 1, the power x^{N+1} should be replaced by x; the power x^{N+2} should be replaced by x^2 and so on. [32].

Example (1.2):

Suppose N=3, and taken two polynomial

$a(x) = 1 + x + 3x^2$ and $b(x) = 2 - 3x + x^2$ then $a + b = 3 - 2x + 4x^2$ (mod 3)

and

$a(x) * b(x) = 2 - x + 4x^2 - 8x^3 + 3x^4 = 2 - x + 4x^2 - 8 + 3x = -6 + 2x + 4x^2$ (mod 3)

The following is the general formula for multiplying polynomials in R:

$a(x) * b(x) = c_o + c_1x + c_2x^2 +,...,+ c_{N-2}x^{N-2} + c_{N-1}x^{N-1}$

Where the c^{th} coefficients c_k is given by the formula

$c_k = a_o b_k + a_1 b_{k-1} +,...,+ a_k b_o + a_{k+1} b_{N-1} + a_{k+2} b_{N-2} +,...,+ a_{N-1} b_{k+1}$

this formula looks a little complicated but it really isn't the coefficient c_k is simply the dot product of the coefficient of a and the coefficient of b , except that first coefficient of b are listed in revrse order and are rotated around k positions. Using these addition and multiplication rules al of the familiar properties are true. For example the distributive law a*(b+c) =a*b+a*c is true in modern terminology, the above addition and multiplication rules make R into a ring. which its call the ring of truncated polynomials. In terms of modern abstract algebra, the ring R is isomorphic to the quotient ring ($Z(X)/X^n - 1$).

Example(1.3):

Let N=8 and $a(x) = 2 + x^2 - 3x^4 + x^7$ and

$b(x) = 1 + 3x + 2x^5 - x^7$ then $a + b = 3 + 3x + x^2 - 3x^4 2x^5 (\bmod 8)$

$a(x) * b(x) = 5 - x + x^2 + 6x^3 - x^4 + 4x^5 - x^6 - x^7 (\bmod 8)$

The NTRU encrypt public key cryptosystem uses the ring of truncated polynomials R combined with the modular arithmetic, these are combined by reducing the coefficient of a polynomials a modulo an integer q.
Thus the expression a (modulo q) means to reduce the coefficient of a modulo q. That is divided each coefficient by q and take the reminder similarly, the relation a=b (modulo q) means that every coefficients of the difference a-b is multiple of q.

Remark (1):

To make storage and computation easier it is convenient to just list the coefficient of polynomial without explicitly writing the powers of x. for example the polynomial $P(x) = p_o + p_1 x +, ..., + p_{N-2} x^{N-2} + p_{N-1} x^{N-1}$
Is conventionally written as the list of N numbers
$p = (p_0, p_1, .., p_{N-2}, p_{N-1})$

Be sure to include zero in the list of some of the powers of x are missing. for example when N=8 the polynomial $P(x) = 2 + x^2 - 3x^4 + x^7$ is stored as the list (2,0,1,0,-3,0,0,1) but if N=10 , then a stored as the list (2,0,1,0,-3,0,0,1,0,0).

1.3.6 Invariability in truncated polynomials rings:

Let $R_q = (Z / qZ)[x]/(x^N - 1)$ be the ring of truncated polynomials modulo q, we compute the probability that a random chosen polynomial $f(x) \in R_q$ is invertible in R_q, and also the probability if $f(x)$ is required to satisfy $f(1) = 1$.

- **Statements**: - [22]

Fix an integer N≥2, for any positive integer q, let R_q denote the ring of truncated polynomial modulo q .

$R_q = (Z/qZ)| \; [X^N - 1]$ .... (1.1)

In this note we shall describe the invertible elements

$R_q^* = \{f \in R_q = f_g = 1 \text{ for some } g \in R_q\}$ ....(1.2)

More that we are interested in the probability that an element of R_q is invertible so in the ratio $\#R^*q| \; \#R_q$

The first observation is that if $q = q_1 q_2$ with gcd $(q_1 q_2)$ =1, then the Chinese reminder theorem tells us that

$R_q = R_{q1} \times R_{12}$ and $R_q = R_{q1}^* \times R_{q2}^*$ ....(1.3)

So that suffices to look at the case that q is a power of a principle the following theorem handless this case.

Theorem 3:- [22]

Let p be a prime, let q = p^k be a power of p, and let N≥2 be an integer with gcd (P, N) =1.

Define n>1 to be the smallest positive integer

Such that $P^n = 1$ (mod N) .... (1.4)

And for each integer d/n. let

$$v_d = \frac{1}{d} \sum_{e/d} M(\frac{d}{e}) \gcd(N, P^e - 1) \text{ .... (1.5)}$$

Where $\sum_{e/d} M(\frac{d}{e}) = \begin{bmatrix} 1 & \text{if} & d=1 \\ 0 & \text{if} & 1<d<n \\ N-1 & \text{if} & d=n \end{bmatrix}$

$$\frac{\#R_g^{\ *}}{\#R_g} = \prod_{d/n} (1-\frac{1}{pd})^{vd} \text{ .... (1.6)}$$

In particular, if N is prime, then rd=0. For all 1<d<n so in this case

$$\frac{\#R_g^{\ *}}{\#R_g} = (1-\frac{1}{p})(1-\frac{1}{pn})^{(N-1)/n} \text{ .... (1.7)}$$

Remark (2):

There is a certain set of non-invertible elements which is easy to describe. To do this one can observe that the evaluation map

$R_q \to Z/qZ,\ f(x) \to f(1)$ .... (1.8)

Is a well defined homomorphism of rings, so it induces a group homomorphism $R_q^{\ *} \to (Z/qZ)^*$ it is well-known that [25]

$(Z/qZ)^* \cong \{a \in Z/qZ = \gcd(a,q) = 1\}$ .... (1.9)

So one can see that if $f(1)$ has a factor in common with q then in cannot be invertible. Thus in looking for invertible element of R_q we should make our random chosen intelligently by requiring that $\gcd(f(1), g) = 1$. In particular one must avoid polynomials with $f(1) = 0$.

For example, one might restrict attention to the subset of R_q and R_q* consisting of polynomials $f(x)$ satisfying $f(1) = 1$. These subset by $(R_q(1)$ and $R_q^{\ *}(1)$ can be denote respectively as f ranges over R_q, the values of f(1) are equidistributed in Z/qZ so that we see that $\#R_q(1) = q^{-1}\#R_q$.

Similarly as f ranges over R_q^*, the values of $f(1)$ are equi distributed in $(Z/qZ)^*$ so $\#R_Q^*(1) = \phi(q)^{-1}\#R_q^*$, where ϕ is the Euler phi function. In particular, if $q = P^k$ is a power of a prime, then $\phi(q) = P^k - P^{k-1}$ and one to find that the probability of intelligently chosen f being invertible is

$$\frac{\#R_q^*(1)}{\#R_q(1)} = (1-\frac{1}{p})^{-1}\frac{\#R_q^*}{\#R_q}$$

Since P tends to be small in application, this is a significant saving for example, if we also assume that N is prime, then

$$\frac{\#R_q^*(1)}{\#R\ \ (1)} = (1-\frac{1}{P^n})^{(N-1)/n} \approx 1-\frac{N-1}{nP^n}$$

<u>Remark</u> (3):

It is clear from theorem, that in order to maximize the probability of getting a unit in R_q, we want to chose N and q so that the order n of P in $(Z/NZ)^*$ is as large as possible. the value of n is easy to compute for specific values of N and P, but for cryptographic purpose n is wanted to be large for a single N and two different values of P(frequently P=2 and P=3). Notice that the possible orders of elements in $(Z/NZ)^*$ are the divisors of $\phi(N)$, so If take N to be prime, the possible orders are divisors of N-1, this suggests N to be prime such that N-1 has very few divisors.

For example, suppose that N is a prime of the form N=2M+1 with M is also prime (the prime M is called a sophic German prime). Then the corresponding n must be either M or 2M. .In practice, if N >100 is taken to be , then every p< 10 has corresponding n= M or 2M , and hence the probability that randomly chosen f satisfying $f(1)=1$ will be invertible is at least [20]

$$1-\frac{N-1}{Mp^{M}}=1-\frac{2}{p^{M}} \quad \ldots\ldots \qquad (1.12)$$

Since $p \geq 2$ and $M \geq 50$ the probability of choosing a non-invertible polynomial is virtually 0 [22].

Example (1.5)

Table 1.2 [43] gives some representative values of N and p, the column labeled np gives the smallest integer n such that

$p^{n} \equiv 1(\text{mod}\, N)$.

N	p	np	Prob_p	N	p	np	Prob_p
11	2	10	$10^{-3.01}$	11	3	5	$10^{-2.08}$
13	2	12	$10^{-3.6}$	13	3	6	$10^{-2.5}$
17	2	8	$10^{-2.1}$	17	3	16	$10^{-7.6}$
19	2	18	$10^{-5.4}$	19	3	18	$10^{-8.5}$
23	2	11	$10^{-3.01}$	23	3	11	$10^{-4.4}$
47	2	23	$10^{-7.22}$	47	3	23	$10^{-11.2}$
59	2	58	$10^{-17.46}$	59	3	29	$10^{-14.1}$
71	2	35	$10^{10.8}$	71	3	35	10^{-17}
107	2	106	$10^{-31.9}$	107	3	53	$10^{-25.5}$
127	2	7	$10^{-3.3}$	127	3	126	$10^{-60.1}$
167	2	83	$10^{-25.2}$	167	3	83	$10^{-39.3}$
229	2	57	$10^{-23.3}$	229	3	57	$10^{-27.8}$
349	2	348	$10^{-104.7}$	349	3	174	$10^{-83.3}$
503	2	251	$10^{-75.8}$	503	3	251	10^{-120}
1019	2	1018	$10^{-306.4}$	1019	3	509	$10^{-243.1}$
1093	2	364	$10^{-110.05}$	1093	3	7	$10^{-5.53}$

Table 1.2 Probability $f(x)$ is not Invertible in R_{p^k}

The column labeled "Prob p" is the probability that a random chosen in R_q satisfying $f(1)=1$ will fail to be invertible in R_q , where $q=p^q$ is any power of p.

(Theorem 1 and Remark (2) show that these probabilities are independent of the exponent k). The values N=47,59,107,167,503,1019 correspond to the sophic Germain prime $(N-1)/2 = 23{,}29.35.83{,}251{,}509$ and thus have especially small probability of failure for all (small) prime p. Conversely, $p=2$ has order 7modulo N=27 , and $p=3$ has order 7 modulo N=1093 , so for these values of p and , the ring R_p has a comparatively large number of non-units.

1.4 Lattices:

This section started by defining the inner product of two vector $v=(v_1,v_2,...,v_m)$ and $u=(u_1,u_2,...,u_m)$,

$$(v,u)=\sum_{i=1}^{m} v_i u_i \ .$$

The inner product is commutative and distributive.

The Euclidian norm or the length of a vector $v=(v_1,v_2,...,v_m)$ is defined as

$$|v|=\sqrt{(v,v)}=\sqrt{\sum_{i=1}^{m} {v_i}^2}$$

Let $b_1, b_2, ..., b_n$ be a linearly independent vectors in R^m and let B be the $n \times m$ matrix with these vectors as rows .The lattice generated by vectors $b_1, b_2, ..., b_n$ or the basis matrix B is the set

$$L(b_1, b_2, ..., b_n) = L(B) = \left\{ \sum_{i=1}^{n} a_i b_i \middle| \ a_i \in Z \right\}$$

Of all linear combinations over integers of the basic vectors .The dimension of this lattice is $\dim(L(B)) = n \le m$.if $m = n$ the lattice is called full dimension, Informally, a lattice is a set of intersection point of a regular, infinite n-dimension grid.

The vector space generated by vectors $b_1, b_2, ..., b_n$ or B is

$$\mathrm{Span}(b_1, b_2, ..., b_n) = \mathrm{Span}(B) = \left\{ \sum_{i=1}^{n} a_i b_i \middle| \ a_i \in R \right\},$$

Where it will be have all linear combination of the basis vectors.

Note that if B' is the result of applying of the following operation to B then $L(B) = L(B')$:

1- Swap the order of two rows in B.

2- Multiply a row of B by -1.

3- Add an integer multiple of a row to another row of B.

The first two cases are trivial, also $L(B') \subseteq L(B)$ in the third case, let $b_j' = b_j + cb_k . j \ne k$ and $b_i' = b_i$, for all $i \ne j$, Now for all $a_i \in Z, \sum a_i b_i = \sum_{i \ne k} a_i b_i' + (a_k - a_j c) b_k'$ and thus $L(B) \subseteq L(B')$.

Definition 1.31: [44]

Let $b_1, b_2, ..., b_n \in R^m$ be linearly independent vectors

The additive subgroup $L(b_1, b_2, ..., b_n) := \sum b_i Z = \{\sum t_i b_i | \ t_1, t_2, ..., t_m \in Z\}$ of R^m lattice with basis $\{b_1, b_2, ..., b_n\}$, the Rank or the dimension of the lattice is rank(L):=n .

For example Z^m is a lattice of rank m, the standard unit vector $e_1, e_2, ..., e_m$ forming a basis.

The determinant of a lattice $L(B)$ with basis $b_1, b_2, ..., b_n$ is defined as

$$\det(L(B)) = \sqrt{\det((b_i, b_j)_{1\le i,j\le n})} = \sqrt{\det(BB^T)}.$$

If $n = m$ we have

$$\det(L(B)) = \sqrt{\det(BB^T)} = |\det(B)|.$$

Because basis vectors are linearly independent, two bases of the same lattice have the same number of vectors. In the following we show that the determinant of a lattice does not depend on the selection of the basis.

<u>Theorem</u> 4:-[44]

Let B and B' be $n \times m$ real matrices and $L(B) = L(B')$. then $\det(L(B) = \det(L(B'))$.

Proof: the rows of both matrices are two bases for the same lattice so we have $B = UB'$ and $B' = VB$ for some $U, V \in Z^{n\times m}$. From these it will be obtained $| \ \det(U) \ | = | \ \det(V) \ | = 1$, then we have

$$\det(L(B)) = \sqrt{\det(BB^T)}$$

$$= \sqrt{\det(VBB^T V^T)}$$

$$= \sqrt{\det(VB(VB)^T}$$

$$= \sqrt{\det(L(B'))}$$

The fundamental parallelepiped associated with B is the set of points We see that $v + P(B)$, $v \in L(B)$, form a partition of the space $\mathrm{Span}(B)$. In other words, for any $u \in \mathrm{Span}(B)$ there exist a unique lattice point $v \in L(B)$ such that $u \in v + P(B)$. If $L = L(B)$ also write $P(L) = P(B)$.

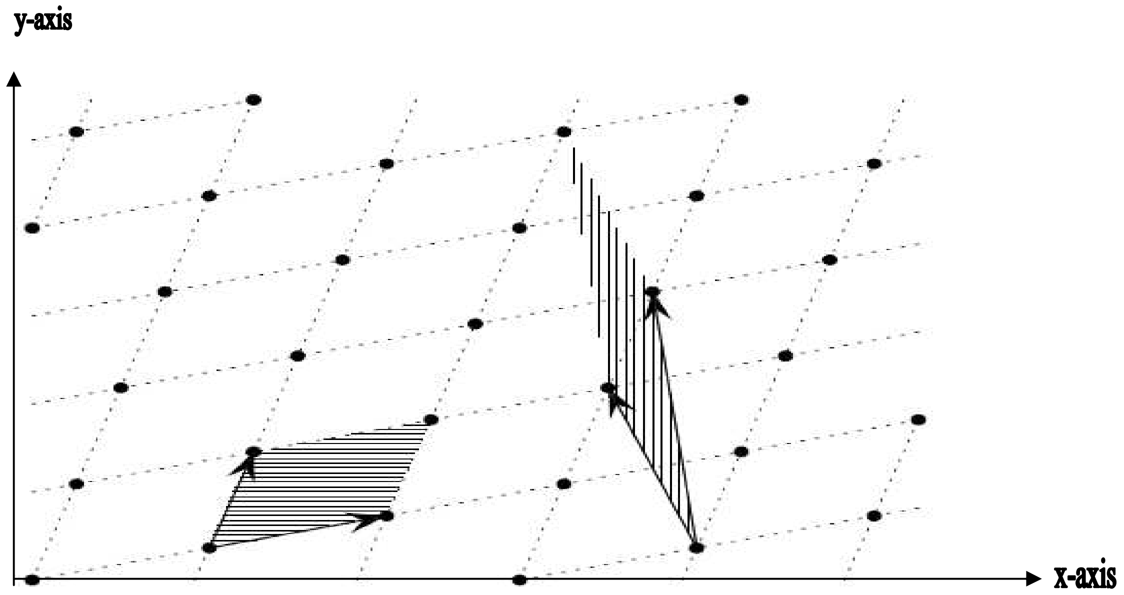


Figure 1.1: A lattice of dimension 2 with two bases and two fundamental parallelepipeds. A fundamental parallelepiped forms a partitioning of the space.

There are two famous computational problems on lattice :the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP) , In SVP one is given a basis $\{b_1, b_2, ..., b_n\}$, In CVP one is given a basis $\{b_1, b_2, ..., b_n\}$ and target vector v and must find the lattice vector in $L(b_1, b_2, ..., b_n)$ closest to v.

<u>Definition 1.32</u> :(Shortest Vector Problem) [28]

The problem of the shortest lattice vector in the ℓ_2-norm is

$\ell_2 - \mathrm{SVP} :=$

$$\left\{ k, m, n, b_1, \ldots, b_n \,\middle|\, k, m, n \in N, b_1, b_2, \ldots, b_n \in Z^m, \exists x \in L(b_1, b_2, \ldots b_n) \backslash \{0\} \|x\|_2^2 \le k \right\}$$

Where $\ell_2 -$ norm means general Euclidean norm.

The complexity of this SVP problem is unresolved .Although some efforts to show that $\ell_2 - \mathrm{SVP}$ is NP-hard problem have failed; this problem is known to be NP-hard with respect to randomized reduction by Ajitai [28]. However the CVP problem is known to be NP-complete for any norm.

Definition 1.33: (Closest Vector Problem)

The problem of the closest lattice vector in the ℓ_2-norm is defined as $\ell_2 - \mathrm{CVP} :=$

$$\left\{ (k, m, n, b_1, \ldots\ldots, b_n, z) / k, m, n \in N, b_1, b_2, \ldots\ldots, b_n, z \in Z^m, \exists x \in L(b_1, b_2, \ldots\ldots, b_n) : \|z - x\|_2^2 \le k \right\}$$

Given a lattice basis $\{b_1, b_2, \ldots, b_n\} \in Z^m$, the following tasks are thought to be hard lattice problem:

- Find a short non- trivial lattice vector.
- Find a basis comprised of short lattice vector.
- Find for a given $z \in \mathrm{Span}(b_1, b_2, \ldots b_n)$ the closest lattice vector

In contrast, given a system of generators $b_1, b_2, \ldots, b_n \in Z^m$ for a lattice L, $n \ge \mathrm{rank}(L)$, it is possible to construct a basis for L in polynomial time.

1.5 Digital Signature:

The notion of a digital signature may prove to be one of the most fundamental and useful inventions of modern cryptography. A signature scheme provides a way for each user to sign messages so that the signatures can later be verified by anyone else. More specifically, each user can create a matched pair of private and public keys so that only he can create a signature for a message (using his private key), but anyone can verify the signature for the message (using the signer's public key). The verifier can convince himself that the message contents have not been altered since the message was signed. Also, the signer can not later repudiate having signed the message, since no one but the signer possesses his private key.

A digital signature scheme within the public key framework, is defined as a triple algorithm (G, δ, V) such that

* Key generation algorithm *G* is a probabilistic, polynomial-time algorithm which on input a security parameter 1^k, produces pairs (P,S) where *P* is called a public key and *S* a secret key. (It will be used the notation $(P,S) \in G(1^k)$ to indicate that the pair (P,S) is produced by the algorithm G).

* Signing algorithm δ is a probabilistic polynomial time algorithm which is given a security parameter $1^{k,}$ a secret key *S* in range $G(1^{k)}$, and a message $m \in \{0,1\}^k$ and produces as output string *s* which we call the signature of *m*. (We use notation $s \in \delta(1^k, S, m)$ if the signing algorithm is probabilistic, otherwise $s = \delta(1^k), S, m)$. As shorthand when the context is clear, the secret key may be omitted and we will write $s \in \delta(S, m)$ to mean meaning that s is the signature of message m.).

* Verification algorithm *V* is a probabilistic polynomial time algorithm which given a public key *P*, a digital signature *s*, and a message *m*, re-turns 1 (i.e.….”true”) or 0 (i.e.… “False”) to indicate whether the signature is valid. It will be required that $V(P,s,m)$ as $V(s,m)$, to indicate verifying signature *s* of message *m* when the context is clear).

Note that if V is probabilistic, we can the requirement on V to accept valid signatures and reject invalid signatures with high probability for all messages m, all sufficiently large security parameter k, and all pairs of keys $(P,S) \in G(1^k)$. The probability is taken over the coins of V and S. Note also that the signed message may be plaintext or encrypted, because the message space of the digital signature system can be any subset of $\{0,1\}$.

3.1 Introduction:

In this chapter improvements are introduced to enhance the NTRU cryptosystem, Matrix NTRU leads to respectable speed improvements at large public key by comparing the results with the classical NTRU. NTRU sign is defined in the same EESS standard as NTRUEncrypt [4]. In the following we outline the characteristics of NTRUSign.

3.2 Matrix NTRU Cryptosystem:

In this section, a new variant of the NTRU public key cryptosystem is proposed, the Matrix cryptosystem. Matrix NTRU works under the same general principles as the NTRU cryptosystem, except that it operates in a different ring with a different linear transformation for encryption and decryption. In particular, it operates in the ring of k by k matrices of polynomials in $\Re = Z[X]/(X^N - 1)$ whereas NTRU operates in the ring $Z[X]/(X^N - 1)$.

The improved efficiency of the linear transformation in Matrix NTRU leads to respectable speed improvements by a factor of $O(K)$ over NTRU at the cost of a somewhat larger public key [29].

3.2.1 Overview:

The Matrix NTRU cryptosystem operates in the ring M of k by k matrices of elements in the ring $\Re = z[x]/(x^N - 1)$. The ring $\Re$ consists of polynomial with degree n-1 having integer coefficients. Multiplication and addition of polynomial in R is done in the usual manner, but exponents of X are reduced modulo n. Matrix multiplication in M is denoted by the use of the symbol *. Besides n and k, Matrix NTRU also uses the parameters p, q 2 N. The numbers p and q may or may not be prime, but they must be relatively prime. In general, p is much smaller than q; in this thesis, for ease of explanation, we stick to p = 2 or p = 3 and q in the range of 2^8 to 2^{11}.

When it is said perform a matrix multiplication modulo p (or q), it means that we reduce the coefficients of the polynomials in the matrices modulo p (or q).

The width of an element $M \in \mathrm{M}$ can be defined to be

$$|M|_\infty = (\max_{\text{ploy m in M}} \text{coff in m}) - (\min_{\text{poly m in M}} \text{coff in m}) \ldots\ldots\ldots\ldots\ldots\ldots (3.1)$$

[11].The width of M is the maximum coefficient in any of its k^2 polynomials minus the minimum coefficient in any of its polynomials. A matrix $M \in \mathrm{M}$ is short if $|M|_\infty \leq P$. When short matrices are multiplied together, we get a matrix whose width may be greater than p but is still almost certainly smaller than q; we call this matrix pretty short. The definitions for width and shortness apply similarly to polynomials in R. For $r \in \Re$ $|r|_\infty = (\max$ $\text{coff in r}) - (\min \text{ coff in r})$. The polynomial r is said to be short if $|r|_\infty \leq P$.

It the size of an element $M \in \mathrm{M}$ will be defined to be

$$|M| = \sqrt{\sum_{\text{polys. min m}} \sum (\text{coff.inM})^2} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.2)$$

When defining some of the sets of short matrices below, the following notation used

$$L(d) = \left\{ M \in \mathrm{M} \,\middle|\, \begin{array}{l} \text{for } i = \left\lceil -\frac{p-1}{2} \right\rceil \ldots\ldots \left\lceil \frac{p-1}{2} \right\rceil, i \neq 0, \text{each polynomial in M} \\ \text{has on average d coefficients equal to i, with reset} \\ \text{coefficient equal to 0} \end{array} \right\} \ldots\ldots\ldots (3.3)$$

For example, if p = 3 and n = 5, then L(2) consists of all matrices of polynomials, where on average each polynomial has 2 coefficients equal to 1, 2 coefficients equal to −1, and 1 coefficient equal to zero. Or, if we have p = 2 and n = 5, then L(2) consists of all matrices of polynomials where on average each polynomial has 2 coefficients equal to 1, and 3 coefficients equal to zero. The parameters for Matrix NTRU consist of the four integers (n, k, p, q), described above and the five sets of matrices (L_f, L_{Φ}, L_A, L_w, L_m) $\subset M$.

These sets have the following meanings and compositions:

1. L_A consists of all matrices $C \in M$, such that $C_0, C_1, \ldots, C_{k-1}$ are linearly independent modulo q; and for short $c_0, \ldots, c_{k-1} \in \Re$, $\sum_{I=0}^{K-1} c_i C^i$ is short , L_A have elements A , B and are used to construct f, g, ϕ, φ .

2. L_f and L_{Φ} consist of all matrices $D \in M$ constructed such that, for $C \in L_A$ and short $c_0, \ldots, c_{k-1} \in \Re$, $D = \sum_{i=0}^{k-1} c_i C^i$. Additionally, matrices in L_f must satisfy the requirement that they have inverses modulo p and modulo q. L_f have the element f, g and it is used to compose the private key, L_{Φ} have ϕ, φ and it is used to generalize random matrices applied for each encryption.

3. The set of messages L_m consists of all matrices of polynomials with coefficients modulo p. therefore it will be expressed as

$$L(m) = \left\{ M \in M \left| \begin{array}{c} \text{polynomial in M have coefficents} \\ \text{between} \left[-\frac{p-1}{2} \right] \text{and} \left[\frac{p-1}{2} \right] \end{array} \right. \right\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.4)$$

This means that each message contains $nk^2 \log_2 p$ bits of information.

4. L_W has elements W and used to construct the public key.

3.2.2 Key Generation:

To create a public/private key pair, the sender chooses two k by k matrices A, B $\in L_A$. Next, the sender randomly selects short polynomials $\alpha_0, \alpha_1, \ldots, \alpha_{k-1} \in R$ and $\beta_0, \beta_1, \ldots, \beta_{K-1} \in R$ then the sender constructs the matrices $f, g \in L_f$ by taking:

$$f = \sum_{i=0}^{k-1} \alpha_i A^i \text{ and } g = \sum_{i=0}^{k-1} \beta_i B^i \ .$$

The matrices f and g must have inverses modulo p and modulo q F_p, $F_{q,}$ and G_p G_q where

$$F_g * f \equiv 1(\bmod q) \text{ and } F_p * f \equiv 1(\bmod q)$$

$$G_p * g \equiv 1(\bmod q) \text{ and } G_q * g \equiv 1(\bmod q)$$

The sender now has his private key, (f, g), although in practice he will want to store the inverses F_p and G_p as well. Now the sender selects a random matrix $W \in L_w$, and constructs the matrix $h \in M$ by taking

$h \equiv F_q * W * G_q (\bmod q)$……………………………………..……………. (3.5)

The sender's public key consists of three matrices (h, A, B).

3.2.3 Encryption:

To encrypt a message to send to the sender, the receiver randomly generates the short polynomials $\phi_0, \phi_1, ..., \phi_{k-1} \in R$. The receiver then constructs the matrices $\phi, \varphi \in L_\phi$ by taking

$$\phi = \sum_{i=0}^{k-1} \phi_i A^i \text{ and } \varphi = \sum_{i=0}^{k-1} \varphi_i B^i .$$

The receiver then takes his message $m \in L_m$, and computes the encrypted message

$e \equiv p(\phi * h * \varphi) + m(\bmod q)$……………………………………………. (3.6)

The receiver sends e to the receiver.

3.2.4 Decryption:

To decrypt, the sender computes

$a \equiv f * e * g(\bmod q)$………………………………………….……………. (3.7)

The sender translates the coefficients of the polynomials in the matrix a to the range −q/2 to q/2 using the centering techniques as in the original NTRU paper [12]. Then, treating these coefficients as integers, the receiver recovers the message by computing

$$d \equiv F_p * a * G_p \,(\mathrm{mod}\, p) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.8)$$

3.2.5 Decryption Analysis:

In decryption, from eq (3.4), the sender has

$$a \equiv f * (p(\phi * h * \varphi) + m) * g) \,(\mathrm{mod}\, q)$$

$$\equiv p(f * \phi * F_q * W * G_p * \varphi * g) + f * m * g \,(\mathrm{mod}\, q)$$

Although matrix multiplication is not generally commutative, f and φ here do indeed commute:

$$f * \phi \equiv \left(\sum_{i=0}^{k-1} \alpha_i A^i\right) * \left(\sum_{i=0}^{k-1} \varphi_i A^i\right) \qquad (\mathrm{mod}\ q)$$

$$\equiv \sum_{i=0}^{k-1} \sum_{i \equiv j+l (\mathrm{mod}\, q)} \alpha_j A^j \phi_l A^l \qquad (\mathrm{mod}\ q)$$

$$\equiv \sum_{i=0}^{k-1} \sum_{i = j+l (\mathrm{mod}\, k)} \varphi_l A^{j+l} \alpha_j \qquad (\mathrm{mod}\ q)$$

$$\equiv \sum_{i=0}^{k-1} \sum_{i \equiv j+l (\mathrm{mod}\, k)} \phi_l A^l \alpha_j A^j \qquad (\mathrm{mod}\ q)$$

$$\equiv \left(\sum_{i=0}^{k-1} \phi_i A^i\right) * \left(\sum_{i=0}^{k-1} \alpha_i A^i\right) \equiv \phi * f \quad (\mathrm{mod}\ q)$$

Similarly $g * \varphi \equiv \varphi * g (\mathrm{mod}\, q)$, so the sender now has that

$$a \equiv p(\phi * w * \varphi) + f * m * g \quad (\mathrm{mod}\ q)$$

For appropriate choices, $|a|_\infty \le q.$, then treating polynomials in this matrix, as having coefficients in Z. The sender can take those coefficients modulo p, leaving $f * m * g (\mathrm{mod}\, p)$. The original message is then recovered by left – multiplying by F_p and right – multiplying by G_p.

<u>Remark</u> (1):-

A matrix f in the ring M will be invertible modulo p and q , only if the corresponding matrix department $\det_f$, which is the ring R is also invertible modulo p and q. In practice, this impossible if $\det_f(1) = 0$, (the sum of the coefficient values of the determinant polynomial is equal to 0), so we must re-select one or more of the polynomial elements in f if this condition was not fulfilled.

3.2.6 Parameter Selection:

- Selection of pairs (f, g) and (ϕ, φ).

We define d_f and d_ϕ such that $L_f = L(d_f)$ and $L_\phi = L(d_\phi)$.
Since the matrices A and B are public, the security of f, g, ϕ and φ necessarily depends on the difficulty of discovering the short polynomials $\alpha_i, \beta_i, \phi_i$ and φ_i. For this reason, and for maximizing the number of possible choices for these polynomials, therefore, one commonly selects.

$$d_f \approx \frac{n}{p} \text{ and } d_\phi \approx \frac{n}{p}.$$

- Selection of A and B

A main concern in generating the matrices f and φ (and likewise, g and φ) is that they must not only commute but they should also be short. Shorter matrices ensure that $\left|p(\phi * w * \varphi) + f * m * g\right|_\infty$ will be smaller, which will allow us to reduce q and valid cipher text will be decipherable. To achieve this, we select A and B to be permutation matrices. A permutation matrix is a binary matrix (i.e. consisting of only the scalars 0 and 1) such that there is exactly one 1 in each row and column with all 0's elsewhere. Since a and b have the additional requirement that the sets $A^o,\dots,A^{k-1}$ and $B^o,\dots,B^{k-1}$ are both linear, we must have that:

$$\sum_{i=0}^{k-1} A^i = \sum_{i=0}^{k-1} B^i = \begin{pmatrix} 1\dots.1 \\ . \\ . \\ .\dots\dots \\ 1\dots.1 \\ \end{pmatrix}$$

This implies that each row and column of f will contain some permutation of $\alpha_0,\dots,\alpha_{k-1}$, meaning that each α_i will appear k- times in f .

Using the column choice of $d_f \approx d_\phi \approx \frac{n}{p}$, we have that

$$|f| \approx \sqrt{k^2|\alpha_i|} \approx \sqrt{\frac{(p-1)nk^2}{p}} \approx |g| \approx |\phi| \approx |\varphi|.$$

-Selection of W

Likewise, f and g should also be chosen to be short in order to keep $|p(\phi * w * \varphi) + f * m * g|_\infty$ small. For security season, it important that w remains secret from the attacker. Therefore, in order to maximize the space of w, make

$$L_w = l\left(\left\lfloor \frac{n}{p} \right\rfloor\right).$$

The size of w is given by

$$|w| = \sqrt{\frac{(p-1)nk^2}{p}}.$$

Remark (2):- Note that when w is chosen in this manner, on average $|w| \approx |m|$. this means that $|\phi * w * \varphi| \approx |f * m * g|$.

Example (3.1)

N=4, k=4, p=2, q=16, d_f =1, d_ϕ =1

$$H = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} ,$$

Let $A, B \in L_H$

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad , \; B = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Chooses short polynomials

$\alpha_0 = 0$, $\alpha_1 = 0$, $\alpha_2 = 0$, $\alpha_3 = 1$ and $\beta_0 = 0$, $\beta_1 = 0$, $\beta_2 = 0$, $\beta_3 = 1$

Constructing f where $f = \sum_{i=0}^{3} \alpha_i A^i$

$$f = \alpha_0 A^0 + \alpha_1 A^1 + \alpha_2 A^2 + \alpha_3 A^3$$

$$f = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

And constructing g where $g = \sum_{i=0}^{3} \beta_i B^i$

$$g = \beta_0 B^0 + \beta_1 B^1 + \beta_2 B^2 + \beta_3 B^3$$

$$g = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$F_p = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$ and $F_q = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$

$G_p \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$ and $G_q = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

Select a random matrix $W \in L_w$

Let $W = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Construct the matrix h, where $h = f_q * w * g_q \pmod{q}$

$h = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$ (Public key consist of three matrices (h, A, B)

For Encryption, we choose short polynomial

$\phi_0 = 0,\ \phi_1 = 1\ ,\ \phi_2 = 1\ ,\ \phi_3 = 0$ And $\varphi_0 = 1\ ,\ \varphi_1 = 1\ ,\ \varphi_2 = 0\ ,\ \varphi_3 = 0$

Where $\varphi = \sum_{i=0}^{3} \varphi_i A^i = \varphi_0 A^0 + \varphi_1 A^1 + \varphi_2 A^2 + \varphi_3 A^3$

$\varphi = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$

And $\varphi = \sum_{i=0}^{3} \varphi_i B^i = \varphi_0 B^0 + \varphi_1 B^1 + \varphi_2 B^2 + \varphi_3 B^3$

$$\varphi = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 0 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix}$$

The message $m \in L_m$

$$\text{Let } m = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$e(m) = p(\phi.h.\varphi) + m(\bmod q)$$

$$e(m) = \begin{pmatrix} 6 & 4 & 5 & 7 \\ 7 & 7 & 5 & 5 \\ 4 & 7 & 7 & 4 \\ 5 & 4 & 7 & 6 \end{pmatrix}$$

To decrypt

$$a = f.e(m).g(\bmod q)$$

$$a = \begin{pmatrix} 7 & 7 & 4 & 4 \\ 4 & 7 & 6 & 5 \\ 4 & 5 & 7 & 6 \\ 7 & 5 & 5 & 7 \end{pmatrix}$$

$$d(m) = f_p * a * g_p(\bmod q)$$

$$d(m) = \begin{pmatrix} 6 & 4 & 5 & 7 \\ 7 & 7 & 5 & 5 \\ 4 & 7 & 7 & 4 \\ 5 & 4 & 7 & 6 \end{pmatrix} (\text{mod } 2)$$

$$d(m) = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

d (m) =m.

3.2.7 Comparison between basic NTRU and Matrix NTRU

Here the theoretical operating characteristics of Matrix NTRU will be compared with those of NTRU, as shown in Table 4.1. The properties are listed in terms of the parameters (N, p, q) for NTRU and the parameters (n, k, p, q) for Matrix NTRU. These should be compared by setting $N = nk^2$ since this equates to plain text message blocks of the same size.

As indicated in the table, the total time for encryption and decryption is $\frac{k}{2}$ times faster for Matrix NTRU than for basic NTRU. Matrix NTRU has a larger public key length as a result of the need to store the matrices A and B, but a smaller private key length due to the particular nature of the private keys f and g.

For example, compare the NTRU "high" security level of (N, p, q) = (107, 3, 128) with the Matrix NTRU parameter choices of (n, k, p, q) = (10, 5, 2, 256).

Characteristic	NTRU	Matrix NTRU
Plain Text Block	N $\log_2$ p bits	nk^2 log2 p bits
Encrypted Text Block	N $\log_2$ p bits	nk^2 log2 p bits
Encryption Speed	$O(N^2)$ operations	O(n2k3) operations
Decryption Speed	$O(N^2)$ operations	O(n2k3) operations
Message Expansion	$\log_p$ q-to-1	$\log_p$ q-to-1
Private Key Length	2N $\log_2$ p bits	$2nk^2$ log2 p bits[2]
Public Key Length	N $\log_2$ q bits	$3nk^2$ log2 p bits[3]

Table 4.1 Comparison between Matrix NTRU and NTRU public key cryptography

Conclusion Remark (1):

1-Since Matrix NTRU performs two-sided multiplications, the constant factor will be about twice that of standard NTRU.

2- A key length of 2nk $\log_2$ p+2k^2 $\log_2$ k bits can be achieved by storing f and g not as matrices but as the 2k polynomials found in the matrices along with their positions in the matrices.

3-A key length of nk2 $\log_2$ q + 2k $\log_2$ k bits can be achieved by storing A and B not as matrices but as the positions of each of the k in the two matrices.

4- For message security, dg is replaced by d for NTRU whereas d_f is replaced by d_g for Matrix NTRU. For ease of comparison, we fix p = 3. The definition of d_g and d used in NTRU is elaborated in [11].

3.3 NTRU Sign Signature Scheme:

The NTRU cryptosystem was first presented by Hoffstein, Pipher and Silverman at CRYPTO'96 It is a ring-based cryptosystem operating in the polynomial ring $Z_q[x] /(x^N - 1)$, where *N* is the security parameter. NTRU has received considerable attention because of its encryption and decryption speed and the ease of creating public-key/secret-key pairs, which makes it practical to change keys frequently. Its security is based on the hard mathematical problem of finding short and/or closed vectors in a certain class of lattices, called NTRU lattices. Since the advent of NTRU encryption scheme, several related signature schemes such as NSS [15] and R-NSS [16] have been proposed. A fast authentication and digital signature schemes called NSS, based on the same underlying hard problem and using keys of the same form, was presented at EuroCrypt 2001 [3]. However, this scheme was broken by Mironov and Gentry et al., [8, 4]. In their Euro crypt presentation, the authors of NSS sketched a revised version of NSS (called R-NSS) and published it in the preliminary cryptographic standard document EESS [3]. Although R-NSS was significantly stronger than the previous version (NSS), Gentry and Szydlo proved that key recovery attack could be mounted [3]. The source of these weaknesses of NSS and R-NSS was an incomplete linking of the NSS method with the approximate closest vector problem in the NTRU lattice. In other words, the weaknesses of NSS and R-NSS arose [12] from the fact that the signer did not possess a complete basis of short vectors for the NTRU lattice $L^{NT}{}_{h}$. Later on, Hofstein et al. proposed a new NTRU based signature scheme called NTRU Sign. Unlike previous signature schemes, the link in NTRU Sign between the signature and the underlying approximate closest vector problem is clear and direct: the signer must solve an approximate CVP problem in the lattice i.e., produce a lattice point that is sufficiently close to a message digest point.

3.3.1 Overview of NTRU Sign:

In this section, we will describe the NTRU Sign digital signature scheme.

In NTRU encryption scheme, basic operations in the ring

$R = Z[x] / (x^N - 1)$, where N is the security parameter. A polynomial

$v(x) \in R$ can be represented by a vector v of its coefficients as follows:

$$v = \sum_{i=0}^{N-1} a_i x^i = (v_0, v_1, \ldots, v_{N-1})$$

For the sake of simplicity, the same notation for the polynomial $v(x)$ and the vector v it will be used. The product of two polynomials v and u in R is simply calculated by v*u=c, where the kth coefficients c_k is

$$c_k = \sum_{i=0}^{k} v_i u_{k-i} + \sum_{i=k+1}^{N-1} v_i u_{N+k-i} = \sum_{i+j \equiv k \pmod N} v_i u_j .$$

Hereafter, we sometimes write a polynomial $v(x)$ as simply a. In some steps, NTRU Sign uses the quotient ring $R_q = Z_q[x] / (x^N - 1)$, where the coefficients are reduced by modulo q, where q is typically a power of 2, for example 128. The multiplicative group of units in R_q is denoted by R^*_q. The inverse polynomial of $v \in R^*_q$ is denoted by a^{-1}. If a polynomial *v* has all coefficients chosen from the set {0, 1}, we call this binary polynomial. The security of NTRU Sign scheme is based on the approximately closest vector problem in a certain lattice, called NTRU lattice. In this scheme, this is based on the approximately closest vector problem in a certain lattice, called NTRU lattice. In this scheme, the signer can sign a message by demonstrating the ability to solve the approximately closest vector problem reasonably well for the point generated from a hashed message in a given space. The basic idea is as follows: The signer's private key is a short basis for an NTRU lattice and his public key is a much longer basis for the same lattice. The signature on a digital document is a vector in the lattice with two properties:

*-The signature is attached to the digital document being signed.

*-The signature demonstrates an ability to solve a general closest vector problem in the lattice.

The way in which NTRU Sign achieves these two properties may be briefly summarized as follows:

3.3.2Key Generation:

The private key includes a short 2*n*-dimensional vector denoted by (f; g). The public key is the large *n*-dimensional vector *h* that specifies the NTRU $L^{NT}{}_h$ that is, h is generated from *f* and *g* by

$h \equiv f^{-1} * g(\bmod q)$ …………………………………………………………... (3.9)

The private key also includes a complementary short vector (F, G) that is chosen so that (f, g) can generate the full NTRU lattice $L^{NT}{}_h$.

3.3.3Signing:

The digital document to be signed is hashed to create a random vector (m_1, m_2) modulo q. The signer uses the secret short generating vectors to find a lattice vector (s, t) that is close to (m_1, m_2).

Verification: The verifier uses the public key h to verify that (s; t) is indeed the lattice L^{NT}_h, and he verifies that (s, t) is appropriately close to (m_1, m_2).

NTRU Sign algorithm uses the centered norm concept instead of Euclidean norm in the verification step to measure the size of an element $a \in R$.

Definition 3.1:

Let v(x) be a polynomial in ring R = Z[x] / (x^N -1). Then the centered norm of v(x) is defined by

$$\|v(x)\|^2 = \sum_{i=0}^{N-1}(v_i - m_0)^2 = \sum_{i=0}^{N-1} v_i^{\,2} - \frac{1}{N}(\sum_{i=0}^{N-1} a_i)^2$$

Where $m_0 = \frac{1}{N}\sum_{i=1}^{N-1} a_i$ is the average of the coefficients of v(x)

The centered norm of an n-tuple (v_1, v_2,…,v_n) with v_1,v_2….,$v_n \in$ R can be defined by the formula

$$(\|v_1, v_2, \ldots\ldots\ldots, v_n\|)^2 = \|v_1\|^2 + \|v_2\|^2 + ,\ldots, + \|v_n\|^2$$

Note that the signature on a document *D* is a vector (s, t) in NTRU Lattice L^{NT}_h, which is very close to *m*. To solve an approximately closest vector problem in the lattice, a signer uses a secret \short basis" defined as shown below:

Definition 3.2: A basis {(f; g), (F; G)} is called a short basis in L^{NT}_h if

$$\|f\|, \|g\| = O(\sqrt{N}), \text{and} \|F\|, \|G\| = O(N)$$

Where N is half dimension of NTRU Lattice L^{NT}_h .

Remark (3):-

$$\|v + u\|_c^{\,2} \approx \|v\|^2_c + \|u\|^2_c$$

Let $v=(v_1,v_2,...,v_n)$ and $u=(u_1,u_2,...,u_n)$. , it will be had

$$\|v+u\|^2=\sum_{i=0}^{n}\left(v_i+u_i-\frac{1}{n}\sum_{j=0}^{n}(v_j+u_j)\right)^2$$

$$=\sum_{i=0}^{n}\left(v_i-\frac{1}{n}\sum_{j=1}^{n}v_j\right)^2+\sum_{i=0}^{n}\left(u_i-\frac{1}{n}\sum_{j=0}^{n}u_j\right)^2+2\sum_{i=1}^{n}\left(\left(v_i-\frac{1}{n}\sum_{i=1}^{n}v_j\right)\left(u_i-\frac{1}{n}\sum_{i=1}^{n}u_j\right)\right)$$

$$=\|v\|^2{}_c+\|u\|^2{}_c+2s \text{ for some } s.$$

On the other hand

$$\sum_{i=0}^{n}\left(v_i-\frac{1}{n}\sum_{j=0}^{n}v_j\right)=\sum_{i=1}^{n}v_i-n\frac{1}{n}\sum_{j=1}^{n}v_j=0.$$

If the vectors are random enough, the quantity *s* is very close to 0.

3.4 NTRU Sign scheme:

In this section we describe NTRU Sign key generation and the NTRU Sign signing and verification protocols.

The system parameters of NTRU Sign include

N: a (prime) dimension.

q: a power of 2.

d_f; $d_{g:}$ key size parameters.

Norm Bound: a bound parameter of verification.

3.4.1 Key generation:

Let f be a polynomial in **R** with d_f randomly selected coefficients set to 1 and the rest is set to 0. Let f be such that there exist an inverse f^{-1} such that $f*f^{-1}\equiv 1(\text{mod}\, q)$. Let g be a polynomial in **R** with d_g arbitrary selected coefficients set to 1 and the reset to 0 and let $h\equiv f^{-1}*g \bmod q$.

First we find polynomial $F_1, G_1 \in$ **R** such that

$f*G_1-g*F_1=1$ .... (3.10)

First note there are polynomials $u, v, k_1, k_2 \in Z[X]$ such that

$$\begin{aligned} f * v + k_1 * (x^N - 1) &= R_f, \\ g * u + k_2 * (x^N - 1) &= R_g, \end{aligned} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.11)$$

In $Z[x]$, R_f and R_g are integers. If R_f and R_g are co-primes we can apply the Extended Euclidian algorithm to obtain integers α and β satisfying

$$\alpha R_f + \beta R_g = 1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.12)$$

Putting above three quantities we have

$$(\alpha v) * f + (\beta u) * g \equiv 1 \bmod x^N - 1 \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.13)$$

Now we have found the polynomial $F_1 = -\beta u$ and $G_1 = \alpha v$.

Denoting that $F' = qF_1$ and $G' = qG_1$. Let

$$B_{fg} = \begin{pmatrix} f^{\rightarrow} & g^{\rightarrow} \\ F' & G'' \end{pmatrix} = \begin{bmatrix} f_0 \quad f_1 \ldots \quad f_{N-1} & g_0 \ g_1 \ldots\ldots g_{N-1} \\ f_{N-1} \ f_0 \ldots \quad f_{N-2} & g_{N-1} \ g_0 \cdots g_{N-1} \\ . \quad . \quad . & . \quad . \quad . \\ . \quad . \quad . & . \quad . \quad . \\ f_1 \quad f_2 \ldots f_0 & . \ g_1 \quad g_2 \ldots\ldots g_0 \\ & \\ F'_0 \quad F'_1 \ldots F'_{N-1} & G'_0 \ G'_1 \ldots\ldots .G'_{N-1} \\ F'_{N-1} \ F'_0 \ldots F'_{N-1} & G'_{N-1} \ G'_0 \ldots G'_{N-1} \\ . \quad . \quad . & . \quad . \\ . \quad . \quad . & . \quad . \quad . \\ F'_1 \quad F'_2 \ldots F'_0 & G'_1 \quad G'_2 \ldots G'_0 \end{bmatrix}$$

And $B_h = \begin{pmatrix} I & g^{\rightarrow} \\ o & q^{I} \end{pmatrix} = \begin{bmatrix} 1 & & & & h_0 & h_1 \ldots\ldots h_{N-1} & \\ & 1 & & & h_{N-1} & h_0 & h_{N-1} \\ & & . & & . & . & . \\ & & & 1 & h_1 & h_2 & h_0 \\ & & & & & & \\ & & & & q & & \\ & & & & & q & \\ & 0 & & & & & . \\ & & & & & & q \end{bmatrix}$

Theorem 3.1 [44]

Matrices B_{fg} and B_h generate the same lattice, i.e. $L(B_{fg}) = L(B_h)$.

Proof: Using previous notation, let

$$U = \begin{pmatrix} U_1^{\rightarrow} & U_2^{\rightarrow} \\ U_3^{\rightarrow} & U_4^{\rightarrow} \end{pmatrix}$$

Where

$U_1 = G_1 - F_1 * h,$

$U_2 = \dfrac{-g + f * h}{q},$

$U_3 = -qF_1$ and

$U_4 = f.$

We have

$U_1 * f + U_2 * F' = G_1 * f - F_1 * h * f - g * F_1 + f * h * F_1 = 1,$

$U_1 * g + U_2 * G' = G_1 * g - f_1 * h * g - g * G_1 + f * h * G_1 = h,$

$U_3 * f + U_4 * F'' = -qF_1 * f + f * gF_1 = 0$ and

$U_3 * g + U_4 * G' = -qF_1 * g + f * gG_1 = g.$

Thus

$UB_{fg} = B_h$.

Furthermore, $\det(U) = (G_1 - F_1 * h) * f + (-g + f * h) * F_1 = 1$.

These equations together imply that $L(B_{gf}) = L(B_h) = L^{NT}{}_h$.

The private signing key will consist of small basis for the lattice B_{fg}, however, the polynomial F' and G' are not small enough for this purpose.

3.4.2 Signing:

The signature of a message m will be a lattice vector close to $(m_1\ m_2)$, where (m_2) is the message reprehensive of m. Here "close to" means that the distance is at most some predefined limit, namely *NormBound.*

The process of signing m starts by computing, using some selected hash function. This method is public as it needs to be done by verifiers of the signature as well.

Let

$$B = \left[\frac{-F * m_1}{q}\right] \text{ and } b = \left[\frac{f * m_1}{q}\right]$$

The signature s of the message is

$s \equiv f * B + F * b \mod p$ ………………………………………………. (3.14)

3.4.3 Verification:

Suppose that we want to verify that s is the signature of message m, and that the public verification key of the signer, h is known.

The first task is to compute

$t \equiv h * s \mod q$ ……………………………………………………. (3.15)

Where $(s\ t)$ is the lattice point. It only remains to be checked that the centered norm of $(s\ t) - (m_1\ m_2)$

Is at most the size of the *NormBound* constant.

3.4.4 Verification Analysis:

The signer wants to find a lattice point near the vector

$J = (m_1 \;\; m_2)$

It can be seen that the secret short basis of the lattice is derived directly from $f, g, F, G)$. If the secret basis is short, it is also orthogonal and the lattice point

$$(s \;\; t) = (B \;\; b)\begin{pmatrix} f & g \\ F & G \end{pmatrix}$$

$$= [(m_1 \;\; m_2)\begin{pmatrix} G/q & -g/q \\ -F/q & f/q \end{pmatrix}]\begin{pmatrix} f & g \\ F & G \end{pmatrix}$$

$$= [(m_1 \;\; m_2)\begin{pmatrix} f & g \\ F & G \end{pmatrix}^{-1}]\begin{pmatrix} f & g \\ F & G \end{pmatrix} \quad \text{.......................................} \; (3.16)$$

Is near J.

Next we will approximate the centered norm of F and G. The polynomials f and g were chosen to satisfy $// f //_c \approx // g //_c \approx c\sqrt{N}$, that it will have

$$F(x) = F'(x) - f(x) * \left[\frac{f(x^{-1}) * F'(x) + g(x^{-1}) * G'(x)}{f(x^{-1}) * f(x) + g(x^{-1}) * g(x)} \right]$$

$$= F'(x) - f(x) * \frac{f(x^{-1}) * F'(x) + g(x^{-1}) * G'(x)}{f(x^{-1}) * f(x) + g(x^{-1}) * g(x)} + f(x) + A(x)$$

$$= F'(x) - \frac{f(x^{-1}) * f(x) * F'(x) + g(x^{-1}) * f(x) * G'(x)}{f(x^{-1}) * f(x) + g(x^{-1}) * g(x)} + f(x) * A(x), \ldots \; (3.17)$$

Where the coefficients of A are in interval $[-\frac{1}{2}, \frac{1}{2}]$. Substituting

$f * G' = g * F' + q$ Into the equation above we get

$$F(x) = \frac{qg(x^{-1})}{f(x^{-1}) * f(x) + g(x^{-1}) * g(x)} + f(x) * A(x) \text{......................} \; (3.18)$$

Because $\|f(x)\|_c = \|f(x^{-1})\|_c \approx \|g(x)\|_c = \|g(x^{-1})\|_c$ Remark (3) gives

$$\left\|f(x^{-1})*f(x)+g(x^{-1})*g(x)\right\|_c \approx \sqrt{\left\|f(x^{-1})*f(x)//_c^2+//g(x^{-1})*g(x)\right\|_c^2}$$

$$\approx \sqrt{2\left\|g(x^{-1})\right\|_c^2}\,.$$

In addition, if the coefficients of A are uniformly distributed in the interval $\left[\frac{-1}{2},\frac{1}{2}\right]$, the centered norm of A is approximately $\sqrt{\frac{N}{12}}$.

This is due to the fact that the square of the average coefficients is

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} x^2 dx = \frac{1}{12}.$$

The centered norm of $F(x)$ has an upper limit which is approximately

$$\frac{q\left\|g(x^{-1})\right\|_c}{\sqrt{2}\left\|g(x^{-1})\right\|_c^2}+\left\|f(x)\right\|_c\left\|A(x)\right\|_c \approx \frac{q}{c\sqrt{2N}}+\frac{cN}{\sqrt{12}} \approx \frac{cN}{\sqrt{12}}.$$

The same approximation holds for G.

It’s shown that $\|f\|_c \approx \|g\|_c \approx c\sqrt{N}$ and $\|F\|_c \approx \|G\|_c \approx \frac{cN}{\sqrt{12}}$.

that have $(s\ \ t)-(m_1\ \ m_2)=(A\,a)\begin{pmatrix} f & g \\ F & G \end{pmatrix}$

Where the coefficients of a and A are in interval $\left[-\frac{1}{2},\frac{1}{2}\right]$, assuming that these coefficients are uniformly distributed on this interval. The centered norms of a and A are approximately $\sqrt{\frac{N}{12}}$. Such that it will be had

$$\left\|(s\ t)-(m_1\ m_2)\right\|_c^2 = \left\|A*f+a*F\right\|_c^2+\left\|A*g+a*G\right\|_c^2$$

$$\approx \left\|A*f\right\|_c^2+\left\|a*F\right\|_c^2+\left\|A*g\right\|^2{}_2+\left\|a*G\right\|^2{}_c$$

$$\approx 2(\frac{N}{12}c^2N + \frac{N}{12}\frac{c^2N^2}{12})$$

$$\approx \frac{c^2N^2}{6} + \frac{c^2N^3}{72}.$$

With these parameter sets, the square root of this quantity is equal to 216 and $c = 0.45$ and *NormBound* is equal to 300.

<u>Example</u> (3.3):

Given a short basis (4, 2), (-1, 2), and using this basis, we can calculate a lattice point (s, t) close to $(0,5)$:

Let

$$U = \begin{bmatrix} 4 & 2 \\ -1 & 2 \end{bmatrix}$$

$$U^{-1} = \begin{bmatrix} \frac{2}{10} & \frac{-2}{10} \\ \frac{1}{10} & \frac{4}{10} \end{bmatrix}$$

$$(0 \quad 5)\begin{bmatrix} \frac{2}{10} & \frac{-2}{10} \\ \frac{1}{10} & \frac{4}{10} \end{bmatrix} = (\frac{5}{10} \quad \frac{2}{10}) \approx (1 \quad 2) \quad \text{and}$$

$$(1 \quad 2)\begin{bmatrix} 4 & 2 \\ -1 & 2 \end{bmatrix} = (2 \quad 5)$$

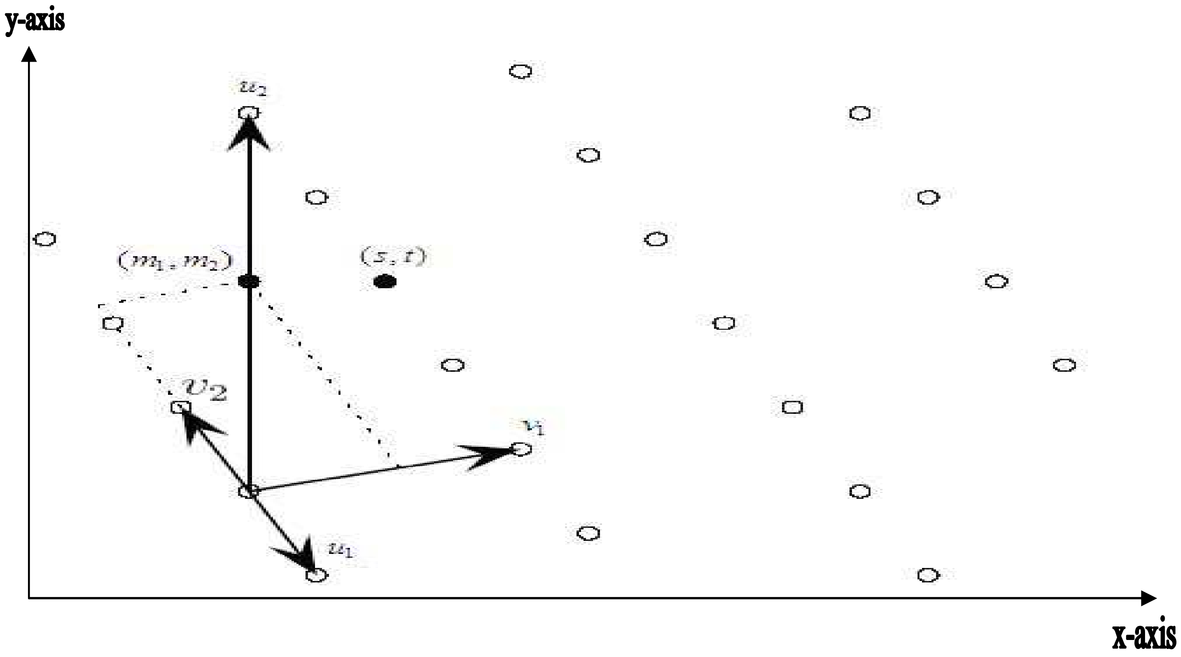


Figure 3.1: A Lattice with two bases $(v_1 = (4 \quad 2), v_2 = (-1 \quad 2)$ **and** $u_1 = (1 \quad -2)\,, u_2 = (0 \quad 9)$. **The message representative** $m_1 = 5$ **and** $s = 2$.

With the same method and a longer basis $(1 \quad -2), (0 \quad 9)$, we can only find a point which is farther away from $(0 \quad 5)$:

$$(0 \quad 5)\begin{bmatrix} 1 & \frac{2}{8} \\ 0 & \frac{1}{8} \end{bmatrix} = (0 \quad \frac{5}{8}) \approx (0 \quad 1)$$

$$(0 \quad 1)\begin{bmatrix} 1 & -2 \\ 0 & 8 \end{bmatrix} = (0 \quad 8)$$

However, any body with this kind of basis can verify that $(s \quad t)$ belong to the lattice: $s * h = 2 * -2 \equiv 5 = t \mod 9$.

<u>3.5 NTRUSign Algorithm</u>:

The NTRUSign domain parameters *N*, *q*

<u>Input</u>:

— The signer's NTRUSign private key vector *f*

— The signer's NTRUSign basis completion vector *F*

— The message representative, which is a polynomial m

Output**:** the signature, which is a polynomial s,

Operation: The signature *s* shall be computed by the following or an equivalent sequence of steps:

```
1. Compute or retrieve the basis completion polynomial F
2. Compute the polynomial B = –F*m1 in $\mathbf{Z}[X]/(X^N – 1)$ (only need to
   store $2*\log_2 q$ bits per coefficient)
3. Set integer $j := 0$
4. While $j < N$ do
    a. Set $B_j := \text{floor}[B_j/q + .5]$
    b. Set $j := j + 1$
5. Compute the polynomial $b = f*m_1$ in $\mathbf{Z}[X]/(X^N – 1)$ (only need to store
   $2\log_2 q$ bits per coefficient)
6. Set $j := 0$
7. While $j < N$ do
    a. Set $b_j := \text{floor}[b_j/q + .5]$
    b. Set $j := j + 1$
8. Set polynomial s := b *F + B*f in $(\mathbf{Z}/q\mathbf{Z})[X]/(X^N – 1)$
9. Output $s$
```

3.6NTRU Verification Algorithm:

The NTRUSign verification is used to indicate if a signature on a message representative satisfies the appropriate verification conditions or not. There is only one NTRUSign verification primitive specified in this standard.

— The NTRUSign parameters *N*, *q*

— The NTRUSign security parameter *NormBound*

Input:

— The signer's NTRUSign public key h

— The signature to be verified, which is a polynomial s

— The message representative *i* for which *s* is alleged to be a signature

Output: A message indicating that the signature is either "valid" or "invalid"

<u>Operation</u>**:** A signature s shall be verified by the following or an equivalent sequence of steps:

```
1. Compute the polynomial t := h*s in $(\mathbf{Z}/q\mathbf{Z})[X]/(X^N - 1)$
2. Compute the polynomial e2 := i - t in $(\mathbf{Z}/q\mathbf{Z})[X]/(X^N - 1)$ (setting coefficients in the range 0 to q – 1)
3. Let maxrange be the largest integer such that $e2_j - e2_k$ = maxrange for some j, k in the range 0 to q – 1 and no coefficient of e2 has values between $e2_j$ and $e2_k$
4. Let $e2_l$ be the largest coefficient of e2 and $e2_m$ be the smallest coefficient of e2
5. Set integer j := $q - e2_l + e2_m$
6. If j > maxrange
    a. Set integer shift := m
7. Else
    a. Set integer shift := j
8. Set j := 0
9. While j < N do
    a. Set $e2_j := e2_j$ – shift (mod q)
    b. Set j := j + 1
10. Let maxrange be the largest integer such that $s_j - s_k$ = maxrange for some j, k in the range 0 to q – 1 and no coefficient of s has values between $s_j$ and $s_k$
```

```
11. Let $s_l$ be the largest coefficient of s and $s_m$ be the smallest coefficient
    of s
12. Set $j := q – s_l + s_m$
13. If j > maxrange
        a. Set shift := m
14. Else
        a. Set shift := j
15. Set j := 0
16. While j < N do
        a. Set $s_j := s_j – \text{shift} \pmod q$
        b. Set j := j + 1
17. Set j := 0
18. Set integers ssum, e2sum, squaresum := 0
19. While j < N do
        a. Set ssum := ssum + $s_j$
        b. Set e2sum := e2sum + $e2_j$
        c. Set squaresum := squaresum + $s_j^2 + e2_j^2$
        d. Set j := j + 1
20. Compute the value CenteredNorm := sqrt((N*squaresum – $\text{ssum}^2$ –
    $\text{e2sum}^2$)/N)
21. If CenteredNorm > NormBound
        a. Output "invalid"
22. Else
        a. Output "valid"
```

Example (3.4):

Here is an example of how to generate another signature from a given message-signature pair. Let parameters be as defined in Efficient Embedded Security Standards (EESS) [4]; N = 251, q = 128, df = 73, dg = 71, and *NormBound* = 300. The binary private key *f; g* and complementary private key *F; G* satisfying

$f * G - g * F = q$ are as following:

$f \ (d_f = 73)$

0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 1 , 1
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0
0 , 0 , 0 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1 , 1 , 1 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 1 , 1 , 0 , 1 , 0
0 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 1 , 0 , 1 , 1 , 1
0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 1 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0
0 , 1 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 0
0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 0
0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 1
0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 0

$g \ (d_g) = 73$

1 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 0
1 , 0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 0 , 1 , 1 , 0 , 0 , 1 , 1 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
0 , 0 , 0 , 1 , 1 , 1 , 1 , 0 , 1 , 1 , 0 , 0 , 1 , 1 , 0 , 1 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0
0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 1 , 1 , 1 , 0 , 0 , 0 , 1
0 , 1 , 1 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0
0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0
0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0
0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 1 , 1 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0

F

-1 , 4 , -1 , 1 , -1 , 0 , -1 , 1 , -4 , 5 , -3 , 3 , 1 , 1 , 0 , -1 , 0 , 3 , 5 , 2 , 0 , 2 , -3 , 1 , -1 , 0
, 3 , -2 , 2 , -2 , 2 , 2 , 2 , 3 , 1 , -2 , 5 , 0 , 1 , 1 , 4 , 2 , -3 , 0 , 1 , 2 , 2 , 0 , 1 , -1 , 0 , 2 ,

3 , 0 , -1 , 1 , 1 , 3 , 2 , 0 , -1 , 1 , -3 , 1 , 1 , 2 , -5 , 0 , 0 , -4 , 2 , -1 , 2 , -2 , 1 , 2 , 5 , 1 ,
0 , 4 , 0 , 1 , -1 , 1 , 0 , 3 , 0 , 5 , 4 , -1 , 3 , -1 , 1 , 0 , 1 , 0 , 2 , 2 , -1 , 0 , -1 , 3 , 2 , -2 ,
-2 , -1 , 0 , 2 , 0 , 0 , 3 , 1 , 5 , -3 , 1 , 3 , 3 , 0 , -2 , 0 , -2 , 2 , -3 , -3 , -1 , 2 , 1 , 0 , 0 , 7
2 , -1 , 3 , -4 , 3 , -1 , 4 , -3 , 3 , 4 , 3 , 3 , 1 , -1 , 1 , -2 , 0 , -2 , 0 , 2 , 2 , 3 , 3 , 3 , 3 , 0 ,
1 , 2 , 1 , 3 , -3 , 0 , -7 , 0 , 0 , -2 , 0 , 0 , 1 , 2 , 2 , 3 , -1 , 3 , 1 , -3 , 3 , 1 , 2 , 2 , 1 , -1 , 4
, -3 , 1 , 2 , -1 , -2 , 5 , 0 , 3 , 1 , 0 , 4 , 3 , 0 , 2 , 2 , 4 , 1 , 1 , -1 , 1 , 2 , 1 , -3 , 2 , 3 , 3 ,
0 , 3 , 0 , -2 , 0 , -1 , 0 , -1 , -2 , 3 , -3 , 1 , -3 , 3 , -1 , -1 , -1 , 1 , -1 , 1 , 0 , -1 , 0 , -1 , 5
1 , 3 , -1 , 0 , 6 , 5 , 0 , -2 , 1 , 2 , 3 , 0 , 0 , 1 , 1 , 1 , 2 , 0

G

1 , -2 , -3 , 2 , 1 , 2 , 2 , -3 , 1 , 0 , -1 , -1 , 2 , 4 , -3 , 2 , 0 , -1 , 2 , 1 , 0 , -1 , -1 , 1 , -2 , 0
, -2 , 2 , 1 , 0 , 4 , 0 , 0 , 1 , -1 , 1 , 2 , 7 , 3 , -1 , 3 , -3 , 2 , 2 , -2 , 1 , 1 , 4 , -2 , 0 , 3 , -1
, 3 , 0 , 2 , 2 , -4 , -2 , 1 , -1 , 2 , 1 , 0 , -1 , -2 , 1 , 4 , 3 , 0 , -1 , -2 , -2 , 1 , 4 , -1 , 1 , 0 ,
3 , -1 , 2 , 1 , 2 , 4 , 1 , 3 , 0 , 0 , 1 , 0 , -1 , -3 , 4 , 4 , 3 , -2 , -2 , -2 , 1 , -2 , 0 , 1 , 1 , -3 ,
-3 , 2 , 1 , 1 , 4 , -1 , 2 , 1 , 3 , 1 , 1 , 0 , 0 , -3 , 1 , 2 , 3 , 2 , 3 , 0 , 5 , 0 , 2 , 3 , 3 , -2 , 2 ,
1 , 2 , 0 , 1 , -3 , 2 , 0 , 0 , -2 , -1 , -1 , 4 , 1 , 3 , -2 , 4 , 1 , 2 , 0 , 2 , 0 , 4 , 2 , 5 , 1 , 0 , 1
, -1 , -1 , -1 , 0 , 1 , 3 , 0 , 0 , 2 , 0 , 2 , 3 , 5 , 1 , 2 , -1 , 3 , 2 , 5 , 2 , 0 , 1 , 0 , 0 , -1 , 1 ,
1 , -1 , -3 , -4 , 3 , 2 , 0 , -1 , 4 , 2 , 3 , -1 , 1 , -1 , -1 , -2 , 0 , 2 , 2 , 4 , 0 , 0 , 2 , 1 , 3 , -3
, -1 , 0 , 2 , 4 , -1 , 0 , 1 , -1 , 1 , 2 , 0 , 4 , -2 , 0 , -4 , 0 , 2 , 0 , -1 , 4 , 0 , 0 , -3 , 1 , 0 , 1
, 2 , 3 , -3 , 2 , 2 , 2 , 2 , 3 , -1 , 4 , 4 , 1 , 0 , 5 , 2 , 2 , 0 ,

The public key $h = f^{-1} * g \pmod{q}$ is

-23 , 36 , -50 , -28 , -4 , -17 , 14 , -16 , -40 , -4 , 40 , -39 , 1 , 14 , -55 , 8 , -62 , -42 , -21 , 6
-49 , 64 , -63 , 9 , 35 , 18 , -44 , -14 , -2 , -17 , 5 , -4 , -7 , -30 , 49 , 27 , 62 , -28 , 46 , -15
-16 , 41 , 42 , -53 , -22 , -42 , -29 , 15 , -24 , 37 , -52 , 39 , -23 , 56 , 43 , 53 , -22 , 50 ,
37 , -51 , 60 , -31 , 52 , -16 , -34 , -5 , 37 , -61 , -5 , -50 , -3 , 61 , 40 , -42 , 25 , -57 , 20
-45 , -1 , 36 , -6 , 62 , 17 , 54 , 32 , -55 , 52 , 16 , 12 , -49 , -30 , 2 , -30 , -62 , -34 , -27
15 , 25 , 22 , -37 , 31 , 64 , 49 , 56 , -10 , -15 , 1 , -43 , 18 , -63 , -16 , -29 , 6 , -4 , 11 ,
34 , -61 , -47 , 22 , 15 , 47 , 14 , -18 , 6 , -36 , 43 , 26 , 34 , -39 , 19 , 25 , -60 , 28 , -16 ,
-12 , 39 , -35 , 38 , -43 , 2 , 8 , 24 , -18 , 12 , 20 , 26 , -16 , 3 , 15 , -7 , 32 , -38 , -28 , 41
45 , 8 , 0 , 57 , 29 , 1 , 6 , 23 , -18 , 24 , 48 , 38 , -36 , 17 , -33 , 60 , 30 , 43 , -38 , -56
38 , -33 , -24 , 3 , 58 , -10 , 56 , -37 , 4 , -17 , 62 , 23 , 57 , -52 , 5 , 19 , 64 , -41 , 34 ,
45 , -23 , 21 , 55 , -29 , -7 , 49 , 19 , 9 , -41 , -14 , 10 , -46 , 57 , -49 , 17 , -22 , -31 , -25 ,
36 , -12 , -9 , 10 , -31 , 58 , -20 , 13 , 55 , 25 , 47 , -36 , 44 , -61 , -25 , 11 , -21 , -6 , 8 ,

-61, -45 , 48,-52, 12 , 52 , 30 , -12 , -2 , -59 , -22 , 48 ,-58 ,-26 ,-52 , -22 ,1 ,-49 ,19 , 29 , 0

Let the message m_1, m_2 be signed to be

m_1

26 , 8 , 30 , -48 , 64 , -10 , 3 , 41 , -41 , 14 , 51 , -31 , 62 , 19 , 40 , -14 , 49 , -12 , -59 , -24 , 7 , -47 , -37 , 22 , -61 , -29 , -48 , 17 , 41 , 64 , 2 , 2 , 8 , -32 , 18 , 7 , 22 , -43 , -16 , 46 , 36 , -29 , -50 , 33 , 54 , 54 , -46 , 39 , -22 , -40 , -50 , 50 , -22 , -22 , 8 , -18 , 13 , 24 , 63 , -10 , 24 , 1 , 56 , -33 , 33 , 10 , 39 , -10 , 32 , -42 , -28 , 4 , -7 , -14 , -28 , -17 , -24 , -9 , -42 , 19 , 16 , -27 , 5 , 58 , 15 , -51 , -25 , -36 , 37 , -26 , 18 , -3 , 40 , 10 , 28 , 8 , -44 , 2 , 63 , 53 , 25 , -29 , -8 , -46 , 21 , 28 , 1 , 62 , -45 , 24 , 17 , 36 , 61 , -43 , 30 , 12 , -29 , -60 , 40 , -57 , -21 , -6 , 4 , -45 , -61 , -32 , 27 , -40 , 35 , 26 , -52 , -5 , 61 , 4 , 13 , 18 , -32 , -50 , 16 , -12 , 38 , -31 , -41 , 34 , -9 , 53 , -19 , 26 , 58 , -43 , 33 , -27 , 15 , -27 , -8 , 19 , 5 , -45 , 43 , -25 , 46 , 55 , 35 , 42 , -5 , -17 , -4 , 27 , -3 , -52 , -50 , -30 , -19 , -26 , -60 , 36 , -38 , -15 , -3 , -44 , 7 , -35 , -7 , -43 , 3 , 50 , 40 , -56 , -60 , 19 , -17 , 50 , 9 , -47 , 28 , -61 , 1 , -41 , 31 , 62 , -28 , 45 , -32 , 17 , -45 , -28 , -12 , -19 , 22 , 49 , 2 , -36 , -50 , 59 , -14 , 18 , 45 , -39 , 26 , 49 , 44 , -56 , 35 , -11 , -38 , -2 , -7 , 28 , 22 , -41 , 26 , 58 , -60 , 58 , 10 , -41 , -34 , 63 , 5 , 53 , 47 , -58 , -47 , 62 , -63 , 3 , 15 , 46 , 29 , -24 , 31 , 0

m_2

9 , -15 , 1 , 63 , 12 , 64 , -9 , -25 , 21 , 15 , -64 , 15 , 20 , 59 , -40 , 43 , -40 , -41 , -16 , -51 , -58 , -9 , -34 , -61 , -7 , 34 , 19 , -26 , -1 , 60 , -59 , -57 , -20 , 6 , -59 , 56 , 5 , -3 , -33 , -38 , -53 , -33 , 41 , 31 , -39 , -63 , 10 , -14 , -40 , 59 , 0 , -34 , -15 , 30 , -30 , 42 , 0 , 53 , -48 , 63 , 48 , -43 , -58 , -36 , 28 , -53 , -45 , -32 , 9 , -13 , -6 , 21 , 18 , -29 , -12 , 44 , -28 , 63 , -35 , -4 , 57 , 29 , 27 , -22 , -5 , 61 , -44 , 60 , 50 , -28 , 58 , 33 , -6 , 64 , 62 , -43 , -53 , -48 , -10 , 21 , 4 , 49 , -23 , -43 , -45 , 29 , -64 , -9 , 27 , -34 , 52 , 20 , 60 , 14 , 63 , -9 , 10 , -46 , -14 , -5 , -9 , -20 , -36 , 49 , -21 , -39 , -58 , -9 , -22 , -3 , -53 , 46 , -19 , -11 , -61 , 1 , -46 , -60 , 56 , 45 , -30 , 44 , 1 , -34 , -7 , -1 , 21 , -61 , 17 , -58 , -1 , -56 , -14 , 28 , 57 , 30 , 53 , 64 , -43 , -33 , -4 , -31 , -51 , 42 , 22 , -48 , -22 , 40 , -44 , -30 , 21 , -9 , -51 , -43 , 21 , 6 , 21 , -23 , 10 , -26 , -16 , -56 , -18 , 35 , 36 , -25 , 0 , 25 , -26 , 21 , 56 , 35 , 55 , -59 , 12 , 12 , -43 , 54 , -12 , -22 , -40 , -56 , 33 , -27 , -34 , -10 , 44 , 51 , 32 , -11 , -39 , -49 , -3 , 7 , 50 , -31 , 46 , -14 , 58 , -45 , -57 , 50 , 55 , 62 , 55 , 2 , 9 , -52 , -8 , 61 , -10 , 16 , -59 , -41 , 54 , -29,13 ,33,-42, -20 , -43 , -17 , -4 , 19 , 55,-18 , 52 , 36 , 32 , 45 , 56 , 0

Now observe a valid signature (s,t) , which is made by a valid signer

s

26 , 26 , 40 , -43 , -52 , 0 , 16 , 38 , -37 , 29 , 47 , -9 , -41 , 43 , -56 , 4 , -60 , -12 , -51 , -12 , 21 , -40 , -34 , 45 , -43 , -16 , -9 , 28 , 53 , -51 , 8 , 0 , 3 , -12 , 24 , 1 , 33 , -44 , -4 , 59 , 52 , -25 , -51 , 36 , 58 , 57 , -33 , 29 , -13 , -50 , -42 , -59 , 2 , 0 , 18 , -16 , 28 , 32 , -52 , -4 , 36 , -4 , 58 , -20 , 55 , 39 , 41 , 8 , 46 , -37 , -4 , 6 , 14 , 6 , -22 , 3 , -17 , -1 , -19 , 20 , 37 , -19 , 11 , -53 , 36 , -52 , -36 , -27 , 45 , -17 , 44 , -3 , 61 , 28 , 30 , 14 , -42 , 14 , -60 , 61 , 16 , -34 , 12 , -41 , 40 , 36 , -11 , -54 , -34 , 30 , 49 , 37 , -59 , -48 , 55 , 29 , -11 , -45 , 50 , -41 , -16 , 9 , 21 , -46 , -37 , -48 , 46 , -34 , 47 , 56 , -34 , -9 , -30 , 23 , 39 , 22 , -29 , -36 , 7 , 5 , 33 , -24 , -33 , 40 , 0 , 41 , -6 , 30 , -60 , -45 , 27 , -15 , 31 , -12 , 11 , 23 , 15 , -25 , 32 , -6 , 43 , -55 , 32 , 42 , 17 , -10 , 15 , 34 , 21 , -44 , -38 , -10 , 0 , -9 , -61 , 54 , -26 , -9 , 6 , -33 , 14 , -26 , -3 , -29 , 35 , 53 , 60 , 63 , -40 , -5 , -5 , -63 , 16 , -26 , 28 , -43 , 2 , -22 , 47 , -52 , -33 , 56 , -32 , 18 , -36 , -20 , 7 , -9 , 48 , 55 , 17 , -14 , -27 , -32 , -14 , 29 , 49 , -26 , 36 , 53 , 53 , -38 , 52 , 6 , -18 , 20 , 19 , 37 , 33 , -28 , 32 , 64 , -49 , -53 , 10 , -21 , -30 , -57 , 15 , 47 , 57 , -58 , -43 , 54 , -61 , 6 , 25 , 54 , 35 , -16 , 56 , 0

$t = s * h (\mathrm{mod}\, q)$

12 , 5 , 9 , -48 , -14 , -38 , 6 , -16 , 52 , 31 , 59 , 23 , 17 , -58 , -27 , -56 , -25 , -21 , 6 , -50 , -54 , 7 , -29 , -28 , -5 , 46 , 20 , -17 , 5 , -62 , -40 , -60 , -22 , 22 , -63 , -62 , 20 , 3 , -30 , -37 , -33 , -19 , 46 , 41 , -44 , -40 , 8 , 6 , -20 , -50 , 15 , -27 , 1 , 45 , -23 , 58 , 15 , -57 , -41 , -62 , 61 , -23 , -37 , -11 , 34 , -39 , -31 , -15 , 14 , 2 , 8 , 17 , 34 , -29 , 8 , 57 , -29 , -52 , -27 , 2 , 45 , 30 , 46 , -18 , 5 , -55 , -27 , -52 , 52 , -18 , -58 , 37 , 21 , -57 , -39 , -29 , -53 , -56 , -9 , 33 , 21 , -60 , -7 , -40 , -20 , 58 , -44 , -3 , 46 , -20 , 62 , 33 , -62 , 40 , -56 , -3 , 24 , -44 , 3 , 10 , -3 , -13 , -45 , 62 , -10 , -32 , -47 , -6 , -14 , 7 , -50 , -60 , -2 , 0 , -51 , 7 , -29 , -46 , -48 , 50 , -21 , 54 , 8 , 1 , 9 , 4 , 37 , -60 , 16 , -41 , 21 , -37 , -1 , 25 , 59 , 34 , -52 , -58 , -29 , -30 , -7 , -29 , -38 , -59 , 50 , -17 , -21 , 44 , -29 , -20 , 45 , 3 , -47 , -19 , 38 , 10 , 30 , 8 , 36 , -17 , 9 , -40 , -4 , 60 , 44 , -9 , 10 , 53 , -3 , 53 , -61 , 36 , -59 , -35 , 23 , 21 , -34 , -63 , -4 , -14 , -20 , -48 , 40 , -36 , -24 , 2 , 44 , -54 , 49 , -6 , -23 , -49 , 0 , 11 , -56 , -23 , 54 , 5 , -46 , -27 , -22 , 52 , -56 , -47 , 54 , 16 , 25 , -28 , 20 , -56 , 11 , 18 -25 , -41,-57 , -31,13 , 24 ,-20 ,-11 , -41 , -13 , 10 , 34 ,-62 , -5 , -51 , 60 , 33 , 47 , -56 , 0

Obviously, the above signature (s, t) is valid and its norm value

$\|s - m_1\|^2 + \|t - m_2\|^2 = 48203 \leq 90000$,

Where $\|s - m_1\|^2 = 25335$ and $\|t - m_2\|^2$=22868 respectively.

2.1 Introduction:

In this chapter we shall present the original NTRU cryptosystem as described in [12]. In the past couple of years, the standards (such as recommended parameters, message generation and scrambling, encryption / decryption processes) were changed several times and will probably change again, but the underlying principles remain the same. It will concentrate on those. First, it would be discussed how the system works, describe the key generation / encoding / decrypting processes with algorithms, then, analysis of NTRU system and security which show a few successful attacks and the way NTRU was changed to circumvent them. The concentrated on attacks according to some circumstances show some proposed results.

2.2Inverse truncated Polynomial Rings:

The inverse modulo q of polynomial (a) is a polynomial (A) with the property that a*A=1(modulo q).

Not every polynomial has an inverse modulo q, but it is easy to determine the inverse, if a has an inverse, we explain how to used “Almost Inverse Algorithm” [38] to compute the NTRU public key pairs [22].

Gives an efficial way to compute the inverse of the polynomial a(x) in the ring (z/2z) (x)/ (m(x)) provided that gcd (a(x), m(x)) =1 and m (0) =-1.

The following IOP describe the necessary steps for performing the inverse of truncated polynomial, the almost inverse algorithm works for the polynomial m(x) =x^N-1 by the NTRU public key cryptosystem.

Algorithm IOP (1): Inverse in (z/2z) (x) / (x^N-1)

```
Input: a(x)
Output: b(x) =$a(x)^{-1}$ in (z/2z)(x)/($x^N$-1)
Step1 : initialization k:0,b(x)=1,c(x)=0 , f(x):=0,g(x):=$x^N$-1
Step2 : do while $f_0$=0
```

```
Step3: $f(x):=F(X)/X$, $C(X):=C(X)*X$, $K=K+1$
Step4 : if $f(x)=1$ then return $x^{N-k}b(x)(\text{mod } x^N-1)$
Step5:    if $\deg(f)<\deg(g)$ then
Step6:      exchange f and g and exchange b and c
Step7: $f(x):=f(x)+g(x)$ (mod 2)
Step8: $b(x):=f(x)+c(x)$ (mod2)
Step9: End while
```

Note that the number f_0 in step 2 is the constant coefficients of f, and that the return value $X^{N-K}b(x)$ (mod (x^N-1), in step 4 is simply b(x) with its coefficients cyclically shifted k places. We also note that the speed of the inversion Procedure can be significantly enhanced by a number of implementation tricks, such as expanding the operation on b,c,f,g into inline loop-unrolled code.

In order to create NTRU public/private key pairs, one needs to compute the inverse of a polynomial modulo p for primes other than 2. Here is adaptation of the almost inverse algorithm or the prime p=3, since the other value required for the standard NTRU parameter sets.

Algorithm IOP (2): Inversion in $(z/3z)(x)/(x^N-1)$

```
Input: $a(x)$
Output: $b(x)=a(x)^{-1}$ in $(Z/3Z)(x)/(x^N-1)$
Step 1: Initialization $k:=0$, $b(x):=1$, $c(x):=0$, $f(x):=a(x)$, $g(x):=x^N-1$
Step2: do while $f_0=0$
Step 3:      $f(x):=f(x)/x$ , $c(X):C(x)*x$ , $k=k+1$
Step 4:      if $f(x)=\pm 1$ then return $\pm x^{N-K}b(x)$ (mod $x^N-1$)
Step 5:          if $\deg(f)<\deg(g)$ then
Step6 :                exchange f and g and exchange b and c
Step7 :              if $f_0=g_0$
```

```
Step8 :     $f(x):=f(x)-g(x)$ (mod3)
Step9 :     $b(x):=b(x)-c(x)$ (mod3)
Step10:     else
Step11:     $f(x): f(x) +g(x)$ (mod3)
Step12:     $b(x): b(x) +c(x)$ (mod3)
Step13: end while
```

In this algorithm, all computation are done modulo (3), so all coefficients are chosen from the set {-1, 0, 1}.Also, the ± 1's is in step4 are chosen to have the same sign.

The creation of NTRU public / private key pairs often requires finding the inverse of a polynomial f(x) modulo not only a prime , but also a prime power , in particular a power of 2.However , once an inverse is determined modulo a prime p , a simple method based on Newton iteration allows one to rapidly compute the inverse modulo p^r.

The following algorithm converges doubly exponentially, in the sense that it requires only about $\log_2(r)$ steps to find the inverse a(x) modulo (p).

Algorithm IOP (3): Inversion in (Z/pZ) (x) / (x^N-1)

```
Input: a(x), p (a prime)
Output: $B(x)=a(x)^{-1}$ in $(Z/pZ)(x) /(x^N-1)$
Step1: Initialization $k:=0 ,b(x):=1,c(x):=0, f(x):=a(x),g(x):=x^N-1$
Step2: do while $f_0=0$
Step3: if deg(f)=0 then
Step5:    $b(x):=f_0^{-1}b(x)$ (mod p)
Step6: return $x^{N-k}b(x)$ (mod $X^N-1$)
Step7:  if deg(f)<deg(g) then
Step8:   exchange f and g and exchange b and c
Step9: u: $=f_0g_0^{-1}$ (mod (p)
```

Step 10: f(x):=f(x)-u*g(x0 (mod p)

Step11: b(x):=b(x)-u*c(x) (mod p)

Step12: end while

The idea of "Almost inverse algorithm "[38] inverse algorithm " stars with the vector (f, g) = (a, m). Then multiplies (on the right) by the following matrices:

$$\alpha=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix},\quad \beta=\begin{bmatrix}x^{-1} & 0\\ 0 & 1\end{bmatrix},\quad \gamma_u=\begin{bmatrix}1 & 0\\ -u & 1\end{bmatrix}$$

Note that the effect of these transformations is

$(f , g)\alpha =(g ,f)$, $(f,g)\beta =(X^{-1} f,g)$, $(f ,g)\gamma_u =(f - u g , g)$

So step 4 is matrix β , step 9 is the matrix α , and step11 is the matrix γ_u, Note that in step11 , the value of u is chosen so that f-u g is divisible by X , so that its constant term is 0) . In step4 divides by X until us constant term is non-zero appear, also, in step 9 makes sure that deg (f) $\geq$deg (g).

The net effect is that each time through the loop the total degree deg(f)+deg(g) is reduced by at least 1, so eventually f becomes a constant (provided gcd (f,g) =1). Hence the algorithm terminates in at most deg (a) +deg (m) iterations.

Thus the algorithm produces a sequence of transformation $D_1D_2.......D_r$, where each D_i (i=1, 2,…,r) is one of α, β, or γ_u, so that

$(a, m)D_1D_2....D_{r-1}D_r = (\chi, *)$

Where χ is a non-zero number modulo p, unfortunately, the coefficients of the product $D_1D_2........D_r$ are not polynomials, because the matrix β has X^{-1} as an entry.

Let k be the number of times that β appears in the product $D_1D_2.....D_r$.

$X^kD_1D_2........D_r$ has coefficients that are polynomial, say

$$X^kD_1D_2.......D_r=\begin{bmatrix}a' & *\\ m' & *\end{bmatrix}$$

Now multiplying on the left by (a, m) yields

$$(a\,a' + m\,m') = (a\,,\,m)\begin{bmatrix} a' & * \\ m' & * \end{bmatrix}$$

$$= (a, m)X^kD_1D_2\ldots\ldots D_r$$

$$= X^k(\alpha, *)$$

So

$$a\,a' = \alpha X^k \pmod{m}$$

The almost inverse algorithm construct this value a' while it is applying the transformation $D_1D_2\ldots D_r$ starting from (a, m), then applying the same transformation starting from (b , c) =(1,0) , except that in place of $\beta=\begin{bmatrix} X^{-1} & 0 \\ 0 & 1 \end{bmatrix}$, we apply $X\,\beta = \begin{bmatrix} 1 & 0 \\ 0 & X \end{bmatrix}$.

Since β has been used k time at the end of the algorithm the value of(b , c) is

$$(b, c) = (1,0)X^kD_1D_2\ldots\ldots D_r = (1,0)\begin{bmatrix} a' & * \\ m' & * \end{bmatrix} = (a',*) .$$

At the end of the algorithm, b has a value satisfying

$$ab = \alpha\, X^k (\bmod\, m)$$

Since the value of α is simply f_0 (the constant term of f, which actually equals f at this stage of the algorithm), so that $a^{-1}=f_0=f_0^{-1}X^{N-k}b$.

see Example (2.1)

Example (2.1):

Let N =4, p=2, $a=1+x^2-x^3$

The inverse of s modulo 2 in $Z(x)/(X^N-1)$

$a1=1+x^2-x^3$ then a1= [1, 0, 1,-1, 0]

$G1=-1+x^4$ then G1= [-1, 0, 0, 0, 1]

Let B1=1 thenB1= [1, 0, 0, 0, 0]

C1=0 then C1= [0, 0, 0, 0, 0]

A1can not be divided by X because of the presence of number one.

A2 = a1+G1 = x^2-x^3+x^4 then a2= [0, 0, 1,-1, 1]

G2=a1=1+x^2-x^3 then G2 = [1, 0, 1,-1, 0]

B2=B1+C1=1 then B2= [1, 0, 0, 0, 0]

A2 can't be divided by X

A3=a2/x = x-x^2+x^3 then a3= [0, 1,-1, 1, 0]

G3=G2=1+x^2-x^3 then G3 = [1, 0,1,-1, 0]

B3=B2=1 then B3= [1, 0, 0, 0, 0]

C3=C2.X=X then C3= [0, 1, 0, 0, 0]

A4 can’t be divided by X because the presence of number one.

A5=a4+G4 = -x-x^3 then a5= [0,-1, 0,-1, 0]

G5=a4=1-x+x^2 then G5 = [1,-1, 1, 0, 0]

B5=B4+C4=1+x^2 then B5 = [1, 0, 1, 0, 0]

C5=B4=1 then C5 = [1, 0, 0, 0, 0]

A5 can't divide by X.

A6= a5/x =-1-x^2 then a6= [-1, 0,-1, 0, 0]

G6=G5=1-x+x^2 then G6= [1,-1, 1, 0, 0]

B6=B5=1+x^2 then B6 = [1, 0, 1, 0, 0, 0]

C6=C5.X=X then C6= [0, 1, 0, 0, 0]

A6 can’t be divided by X because of the presence of number one.

A7=a6+G6 =-x then a7= [0,-1, 0, 0, 0]

G7=a6=-1+x+x^2 then G7= [-1, 0,-1, 0, 0]

B7=B6+C6=1+x+x^2 then B7= [1, 1, 1, 0, 0]

C7=B6=1+x^2 then C7 = [1, 0, 1, 0, 0]

A7 can't be divided by X

A8=a7/x = -1 then a8= [-1, 0, 0, 0, 0]

G8=G7=-1-x^2 then G8= [-1, 0,-1, 0, 0]

B8=B7=1+x+x^2 then B8= [1, 1, 1, 0, 0]

$C8=C7.X=x+x^3$ the $C8=[0, 1, 0, 1, 0]$

Because a8 equal to constant the evaluation stops and the inverse a is

$A=X^{N-K}*B8$ (k equal to No of division times by x)

And for k=4

$A=X^{4-4}*B8=B8=1+x+x^2$ then $A=[1, 1, 1, 0, 0]$

2.3 NTRU PKCS Parameters:-

The basic collection of objects used by the NTRU public key cryptosystem is the ring R [12] that consist of all truncated polynomials of degree N-1 having integer coefficients :

$$P(x) = p_0 + p_1 x + p_2 x^2 + \ldots\ldots + p_{N-2} x^{N-1} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(2.1)}$$

Polynomials are added in the usual way.

They are also multiplied more or else as usual except that X^N are replaced by 1, X^{N-1} is replaced by X and X^{N+2} are replaced by X^2.

A Full implementation of the NTRU public key cryptosystem is specified by a number of parameter. The NTRU PKCS have three parameters:

N the polynomial in the truncated polynomial ring has degree N-1.

P small modulus .As the final step in decryption, the coefficient of the message is reduced modulo p.

q Large modulus, the coefficient of truncated polynomials will be reduced Modulo q.

2.4 Star Multiply:

This function performs the polynomial multiplication of a*b mod X^N-1. As a note, the M in Step 9 is either p or q depending upon which one is passed into the function. In contrast to the guideline in [14], the algorithm executes Step 9 if the current coefficients of a[i] and b[j] are both non-zero. This, therefore, eliminates approximately a third of the operations, which are unnecessary.

Also, for the case M = q, the algorithm assumes q = 2^w so the reduction is performed by extracting the lower w bits.

2.4.1Star Multiply Algorithm: (a; b; c;N;M)

Input: N, the coefficient modulus, M, and the two polynomials to be multiplied, a and b.

Output: (a;b;c;N;M)

```
step1: for k = 0 to N-1 do
step2: c[k] = 0
Step 3: for i=0 to k do
Step 4:      c(k) =c(k)+[a(i)*b(j)] mod M
Step 5: End for
Step 6: For i= k+1 to N-1 do
Step 7:       c(k) =c(k)+[a(i)*b(j)] mod M
Step 8: End for.
```

2.5 Rand Polynomial:

The Random Polynomial function generates a random polynomial, r, whose coefficients are in the subset {-1, 0, 1}.

The user specifies the number of ones (NumOnes) and the number of negative ones (Num Neg Ones) that will make up the random polynomial, r. Basically, the algorithm works by randomly selecting a location (position) between 0 and N in the random polynomial vector, r. For each selected location, if the value is zero the algorithm replaces the zero by 1 or -1 until all the specified number of ones and negative ones have been entered into the vector.

2.5.1 Algorithm Random Polynomial (r;N, NumOnes, NumNegOnes)

Input: N, NumOnes, NumNegOnes, and polynomial vector to be made random r.

Output: random polynomial r.

```
Step 1: r=0
Step 2: while NumOnes <>0 or NumNegOnes <> 0  do
Step: position = rand () mod N
Step 4: if r(position)= 0 then
Step 5:     if NumOnes > 0 then
Step 6:        r[position] = 1
Step 7:        NumOnes = NumOnes - 1
Step 8:     else if NumNegOnes > 0 then
Step 9: r (position) = -1
Step 10: NumNegOnes = NumNegOnes- 1
Step 11: end if
Step 12: end if
Step 13: end while.
```

2.6 Inverse Polynomial Fq:

The Inverse Polynomial Fq function in is responsible for generating the inverse polynomial of the secret key, f, modulo q.

2.6.1 Algorithm Inverse Polynomial Fq(a; Fq;N; q)

Input: the polynomial to invert a(x), N, and q.

Output: Fq (a;F:N;q)

```
Step 1: k = 0
Step 2: for i=o to N
Step 3: $d_g$=-1, $d_f$ =-1
Step 4: b (i) =0, c (i) =0
Step 5: f (i) =a (i)
Step 6: if a (i) <>0 then
Step 7: $d_f$=i
Step 8: End if
Step 9: g (i) =0
Step 10: End for
Step 11: g (0) =-1, g (N) = 1, b(0) = 1, $d_g$=N
 Step 12: loop
Step 13: while f (0) =0 do
Step 14: temp2=c (n-1)
Step 15: for i=1 to N-1 do
Step 16: f(i-1)=f(i)
Step 17: c(N+1-i)=c(N-i)
Step 18: end for
Step 19: f(n)=0 , c(0)=temp2, $d_f$=$d_f$-1, k=k+1
Step 20: end while
Step 21: if $d_f$=0
Step 22: go to 46
Step 23: end if
Step 24: if $d_f$<$d_g$ then
Step 25: temp2=$d_f$
Step 26: $d_f$=$d_g$ , $d_g$=temp2 , temp=f , temp=f , f=g , f=g , g=temp , temp=b ,
         b=c, c=temp
Step 27: end
Step 28: if f=f+g mod 2
Step 29: b=b+c mod 2
Step 30: end loop
Step 31: j=0
Step 32: k=k mod N
Step 33: for i = N - 1 downto 0 do
Step 34: j=i-k
Step 35: if j<0 then j=j+n
Step 36: end if
Step 37: $F_q$(i)=b(i)
 Step 38: end for
Step 39: v=2
Step 40: while v<q do
Step 41: v=v*2
Step 42: star multiply (a,$F_q$, temp,$F_q$,N,v
Step 43: star multiply (a,$F_q$, temp,$F_q$,N,v)
Step 44: end while
Step 45: for i =N-1 downto 0 do
Step 46: if $F_q$(i) <0 then
Step 47: $F_q$(i)=$F_q$(i)+q
Step 48: end if
Step 49: end for
```

2.7 Inverse Polynomial Fp:

The Inverse Polynomial Fp function is responsible for generating the inverse polynomial of the secret key, f, modulo p. Inverse polynomial F_p is based o R the pseudo-code for \Inversion in (Z=pZ)[X]=(X^N-1)" provided in [15].

2.7.1 Algorithm Inverse Polynomial Fp (a; Fp;N; p)

Input: the polynomial to invert a(x), N, and p.

Output: F_p(a;F_p;N;p)

```
step 1: k = 0
step 2: for i=o to N
step 3: $d_g$=-1 , $d_f$ =-1, b(i)=0 c(i)=0
step 4: f(i)=a(i)
step 5: if a(i)<>0 then $d_f$=i
step 6: End if
 step 7: g(i)=0
step 8:  end for
step 9: g(N) = 1 , $d_g$=N, b(0) = 1
step 10: loop
step 11:  while f(0)=0 do
step 12: for i=1 to N-1 do
step 13: f(i-1)=f(i) , c(N+1-i)=c(N-i)
step 14: end for
step 15: c(0)=temp2 , $d_f$=$d_f$-1 , k=k+1
step 16: end while
step 17: if $d_f$=0 go to 55
step 18: end if
step 19: if $d_f$<$d_g$ then
step 20: temp2=$d_f$ , $d_f$=$d_g$ , $d_g$=temp2 , temp=f , f=g , g=temp , temp=b , b=c ,
c=temp
step 21:  end if
step 23: if f(0)+3)mod3= (g(0)+3) mod 3 then
step 24: for i=0 to N-1 do
step 25: f(i)=(f(i)-g(i)) mod3
step 26: b(i)=(b(i)-c(i))mod 3
step 27: end for
step 28: else
step 29: for i=0toN-1 do
```

```
step30: f(i)=(f(i)-g(i)) mod3
step31:  b(i)=(b(i)-c(i))mod 3
step 32: end for
step 33: end if
step 34: end loop
step 35: j=0  , k=k mod N
step 36: if f(0)+3)mod3=1 then
step 37: for i=0 to N-K-1 do
step 38: end for
step 39: for i=N-k to N-1 do
step 40: if f(0)+3)mod3
step 41: end for
step 42: end if
step 43 : for i=0 to N-k-1 do
step 44: result(i)=-1*(b(k+i-N)+3 mod 3
step 45: end for
step 46: for i=N-k to N-1 do
step 47: result(i)=-1*(b(k+i-N)+3 ) mod 3
step 48: end for
step 49:  end if
```

Algorithms finds the inverse polynomial modulo a power of 2, which is q. is based off the pseudo-code for \Inversion in (Z/2Z) [X] = (X^N-1)" and inversion in (Z=^{pr}z) [X] = (X^N-1)" provided in [15].

2.8 Key Generating: - [12]

For generation the public key we must choose a random secret key of binary Numbers (0, 1, 1, 0…) and as a polynomial, $f \in R$ with coefficient reduced modulo p.

2.8.1 Overview

The sender wants to create a public/private key pair for the NTRU Public Key Cryptosystem.

First randomly chooses two "small" polynomials f and g in the ring of truncated polynomials R.

A "small" polynomial is small relative to a random polynomial mod q. In a random polynomial, the coefficients will in general be randomly distributed mod q; in a small polynomial, the coefficients are much smaller than q.

The sender must keep the values of the polynomials f and g private, since anyone who knows the value of either one will be able to decrypt messages sent to the sender.

The sender next step is to compute the inverse of f modulo q and the inverse of **f** modulo p.

Thus computes polynomials fq and fp with the property that

$f * f_q = 1 (\mod q)$ and $f * f_p = 1 (\mod p)$. (If by some chance these inverses do not exist, the sender will need to go back and choose another f.

Now the sender computes the product

$h = p * f_q * g (\mod q)$ ………………………………………………….. (2.2)

The sender's private key is the pair of polynomials f and f_p. the sender's public key is the polynomial h.

2.8.2 Create Key Algorithm (N; q; p; f; g; h; F_p; F_q)

Input: p, q, N and random polynomials, f and g.

Output: $h = p * f_q * g (\mod q)$

step1: Inverse Poly Fq(f; F_q;N; q)

step 2: Inverse Poly F_p(f; F_p;N; p)

step 3: Star Multiply (F_q; g; h;N; q)

```
step 4: for i = 0 to N - 1 do
step 5: if h[i] < 0 then
step 6: h[i] = h[i] + q
step 7: end if
step 8: h[i] = h[i] * p mod q
step 9: end for
```

See example (2.2)

Example (2.2):

As an illustration of the above discussion, the considered parameters are:

N=11 q=32 p=3

Also needed to define a "small" polynomial more precisely (less than N=11). For the purposes of this example, the key generation can be done by using the quantities d_f and d_g. The polynomial f has d_f coefficients equal to +1, (d_{f-1}) coefficients equal to -1, and the rest equal to 0.

Also, polynomial g has d_g coefficients equal to +1, d_g coefficients equal to -1, and the rest equal to 0.

(The reason for the slight difference in form between f and g is that f has to be invertible, while g doesn't). For the purposes of this section, it will be taken as **Case (I):**

$d_f = 4$ $d_g = 3$.

The sender needs to choose a polynomial f of degree 10 with four 1's and three -1's, and he needs to choose a polynomial g of degree 10 with three 1's and three -1's.

The chosen polynomial f and g are:

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

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

Next computes the inverse fp of f modulo p and the inverse fq of f modulo q.

The sender finds that:

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

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

$$+ 20x^8 + 18x^9 + 30x^{10}$$

The final step in key creation is to compute the product

$$h = pf_q * g = 8 + 25x + 22x^2 + 20x^3 + 12x^4 + 24x^5 + 15x^6 +$$

$$19x^7 + 12x^8 + 19x^9 + 16x^{10} (\text{mod}32).$$

So the sender's private key, in this case is the pair of polynomials f and fp, and his public key is the polynomial h.

For the purpose of high security level we increase the degree N of the chosen polynomial

Case (II): Taken d_f=4, $dg_{=3}$, N=12

Now, the sender need to choose a polynomial of degree 12 with fourth's 1 and three's 1 and three's -1

Suppose the sender chooses:

$$f = -1 + x + x^2 - x^6 + x^9 + x^{11} - x^{12}$$

$$g = -1 + x^2 + x^5 + x^8 - x^{10} - x^{12}$$

Next he computes the inverse F_p of F modulo P and the inverse F_q of F modulo q.

$$f_p = 2x + 2x^5 + x^6 + x^8 2x^{12}$$

$$f_q = 9 + 5x + 6x^2 + 21x^3 + 27x^4 + 19x^5 + 8x^6 + 3x^7 + 28x^9 + 17x^{10} + 11x^{12}$$

Finally $h = pf_q * g(\text{mod}\,32)$

$$h = 21 + 28x + 27x^2 + 30x^3 + 20x^4 + 19x^6 + 31x^7 + 14x^9 + 23x^{10}$$
$$+ 19x^{11} + 12x^{12} \pmod{32})$$

The sender's private key is the pair of polynomial f and f_p and his public key is the polynomial h.

2.9 Encryption: [12]

The encrypted message is computed as

$$e = pr * h + m \pmod q) \quad \text{……………………………………………….. (2.3)}$$

Where the message $m \in R$, and the random polynomial $r \in R$ has coefficients reduced modulo P.

2.9.1 Overview

The receiver wants to send a message to the sender using the sender's public key h. first puts the message in the form of a polynomial m whose coefficients are chosen modulo p, say between -p/2 and p/2 (in other words, m is a small polynomial mod q). Next it should be randomly chooses another small polynomial, r. This is the "blinding value", which is used to obscure the message (random value when encrypting).

The sender uses the message m, her randomly chosen polynomial **r**, and the receiver's public key h to compute the polynomial

$$e = r * h + m \pmod q).$$

The polynomial e is the encrypted message which the receiver sends to the sender.

2.9.2 Encoding Algorithm (N; q; r; m; h; e)

Input: N, q, Public Key h, message m, and random polynomial r.

Output: $e = pr * h + m \pmod q)$

Step1: Star Multiply(r; h; e;N; q)

```
Step 2: for i = 0 to N -1 do
Step 3: e[i] = e[i] + m[i] mod q
Step 4: end for.
```

Example (2.3):

In this example, we shall consider the tow cases of example (2.2)

We need to specify what we mean by saying r is "small" polynomial. We do this using the quantity d_r.

Where r has d_r of coefficients:

$$d_r = \begin{bmatrix} 1\,, & \text{Coefficients are} + \text{ve} \\ -1\,, & \text{Coefficients are} - \text{ve} \\ 0\,, & \text{Coefficients are zero} \end{bmatrix}$$

For the purposes of case of example (2.2)⁝, it will be taken $d_r = 3$.

Now, suppose the receiver wants to send the message

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

To the sender using the sender's public key

$$h = 8 + 25x + 22x^2 + 20x^3 + 12x^4 + 24x^5 + 15x^6 + 19x^7 + 12x^8 + 19x^9 + 16x^{10}$$

For case (Π) of example (2.2), the receiver wants to send the message

$m = -1 + x^3 - x^4 - x^8 + x^9 + x^{10} + x^{12}$ to the sender using the sender public key

$$h = 21 + 28x + 27x^2 + 30x^3 + 20x^4 + 11x^5 + 19x^6 + 31x^7 + 14x^9 + 23x^{10} \\ + 19x^{11} + 12x^{12}$$

First chooses a random polynomial r of degree 12 with three's 1 and three's -1, say she chooses

$$r = -1 + x^2 + x^4 - x^8 - x^{11}$$

Then the encrypted message e is:

$e = r * h + m(\mathrm{mod}\, 32)$

$e = 5 + 3x + 17x^2 + 20x^3 + 6x^4 + 15x^5 + 6x^6 + 11x^7 + 24x^7 + x^9 + x^{10}$
$\quad + 26x^{11} + 26x^{12}(\mathrm{mod}\, 32).$

Then send the encrypted message e to the sender.

2.10 Decryption: [12]

The decryption procedure requires three steps:

(i) Constructing polynomial $a = f * e(\mathrm{mod}\, q)$ ………………………… (2.4)

(ii)Shift the coefficients of a to the range (-q/2, q/2).

(iii) Calculating the polynomial $d = f_p * a(\mathrm{mod}\, p)$ …………………… (2.5)

The final step of decryption requires the user to compute the inverse polynomial F_p of the secret key f modulo p. The decryption process outlined above will recover message (d = m).

2.10.1 Overview

The sender has received the receiver's encrypted message e and wants to decrypt it. It will be begin by using his private polynomial f to compute the polynomial

$a = f * e(\mathrm{mod}\, q)$.

Since the sender is computing a modulo q, he can choose the coefficients of a to lie between -q/2 and q/2. (In general, the sender will choose the coefficients of a to lie in an interval of length q. The specific interval depends on the form of the small polynomials.

It is very important that the sender does this before performing the next step, the sender next computes the polynomial

$b = a(\mathrm{mod}\, p)$.

That is, he reduces each of the coefficients of a modulo p. Finally the sender uses his other private polynomial f_p to compute

$c = f_p * b(\mathrm{mod}\, p).$

The polynomial **c** will be the receiver's original message m**.**

2.10.2 Decoding Algorithm (N; q; p; f; F$_p$; e; d)

Input: N, q, p, secret key f, inverse polynomial F$_p$, and encrypted message e .

Output: $d = f_p * a(\mathrm{mod}\, p)$

```
Step 1: Star Multiply (f; e; a;N; q)
Step 2: for i = 0 to N - 1 do
Step 3: if a[i] < 0 then
Step 4: a[i] = a[i] + q
Step 5: end if
Step 6: if a[i] > q=2 then
step 7: a[i] = a[i] + q
Step 8: end if
Step9: end for
Step 10: Star Multiply (a; F_p; d;N; p).
```

Example (2.4):

In this example we shall considered the result of examples (2.2) and (2.3) are valid, the sender has received the encrypted message

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

from the receiver. The sender uses his private key f to compute

$a = f * e(\mathrm{mod}\, 32).$

$$a = 3 - 7x - 10x^2 - 11x^3 + 10x^4 + 7x^5 + 6x^6 + 7x^7 + 5x^8 - 3x^9 - 7x^{10}(\mathrm{mod}\, 32).$$

Note that when the sender reduces the coefficients of f*e modulo 32, he chooses values lying between -15 and 16, not between 0 and 31. It is very important that he choose the coefficients in this way. Next the sender reduces the coefficients of a modulo 3 to get

$$b = a = -x - x^2 + x^3 + x^4 + x^5 + x^7 - x^8 - x^{10} (\mathrm{mod}\, 3).$$

Finally the sender uses f_p, the other part of his private key, to compute

$$c = f_p * b = -1 + x^3 - x^4 - x^8 + x^9 + x^{10} (\mathrm{mod}\, 3).$$

The polynomial **c** is the receiver's message m, so the sender has successfully decrypted the receiver's message.

-**Now, if** the sender received the encrypted message

$$e = 5 + 3x + 17x^2 + 20x^3 + 6x^4 + 15x^5 + 6x^6 + 11x^7 + 24x^8 + x^9 + x^{10} + 26x^{11} + 26x^{12}$$

From the receiver he uses his private key f to compute

$$a = f * e = 1 - 4x - 13x^2 - 10x^3 - 2x^4 - 3x^5 + 11x^7 + 5x^8 - 2x^9 - 10x^{10} + 3x^{11} + 5x^{12} (\mathrm{mod}\, 32)$$

Next the sender reduce the coefficients of a mod 3 to get

$$b = a = 1 - x - x^2 - x^3 - 2x^4 + 2x^7 + 2x^8 - 2x^9 + x^{10} + 2x^{12} (\mathrm{mod} 3)$$

Finally the sender uses f_p the other part of his private key to compute

$$d = f_p * b = -1 + x^3 - x^4 - x^8 + x^9 + x^{10} + x^{12} (\mathrm{mod}\, 3).$$

The polynomial d is the receiver message m so the sender has successfully decrypted the receiver's message.

2.10.3 Decryption Analyses:

The receiver's encrypted message e of the form e = r*h + m (mod q), but the sender doesn't initially know the values of r and m. The sender's first step is to compute f*e and reduce the coefficients modulo q. Remember that the sender's public key h was actually formed by Multiplying pf_q*g and reducing its coefficients modulo q. So although the sender doesn't know r and m, when he computes a = f*e (modulo q), he is actually performing the following computation:

a = f*e (modulo q)

= f*(r*h + m) (modulo q) [since e = r*h + m (modulo q)]

= $f*(r*pf_q*g + m)$ **(**modulo q) [since h = pfq*g (modulo q)]

= pr*g + f*m (modulo q) [since f*fq = 1 (modulo q)] ……… (2.6)

Now look back at the sizes of the various parameters. The polynomials r, g, f, and m all have coefficients that are quite small. This means that the coefficients of the products r*g and f*m are also quite small, at least in comparison to q. Since the prime p is also small compared to q, this means (assuming that the parameters have been properly chosen) that the coefficients of the polynomial pr*g + f*m already lie between -q/2 and q/2, so reducing the coefficients modulo q has no effect at all!

In other words, when the sender computes a by first multiplying f*e and then reducing the coefficients modulo q, the polynomial a he ends up with is exactly equal to the polynomial pr*g + f*m. When the sender next reduces the coefficients of a modulo p to form the polynomial b, he is really reducing the coefficients of pr*g + f*m modulo p, so the b that he ends up with is equal to b = f*m (modulo p).

Remember that the sender still doesn't know the value of m, but he now knows the value of b. So his final step is to multiply b by f_p and use the fact that $f_p*f = 1$ (modulo p) to compute

$c = f_p* b = f_p* f*m = m$ (modulo p),

which allows him to recover the receiver’s message m.

2.11 Parameter Choice:

The recommended parameters for different levels of security are presented in Table 2.1.

Also the sizes of the resulting keys are presented. The size of the public key is computed as N $\log_2 q$, the number of coefficients times the number of bits needed to store each coefficient. The size of the private key is computed as 2N $\log_2$ p, where the 2 comes from the fact that we should keep not only f, but also f−1 p to make decryption efficient. The parameters are suggested by NTRU Cryptosystems [46].

Table 2.1. Recommended parameter choices for different levels of security . Also, the resulting key sizes (in bits) are shown.

Security level	N	p	q	d_f	d_g	d_r	$/f/+/f^{-1}_p/$	/h/
Moderate	167	3	128	61	20	18	530	1169
Standard	263	3	128	50	24	16	834	1841
Highest	503	3	256	216	72	55	1595	3521

2.11.1 Successful Decryption:

We need to choose the parameters carefully in order to be able to decrypt: the following condition needs to be verified often enough:

$$|pr * g + f + *m|_{\propto} < q \text{ ………………………… (2.7)}$$

(Recall that for $f \in R$, $|f|_{\infty}$ denotes its width, (i.e. the maximum deference between two coefficients of f). The coefficients of f, r and g should be small enough for the previous inequality to hold with high probability. The authors of NTRU [30] define the following set:

L(d1, d2) = {F $\in$ R| F has d_1 coefficients equal to 1, d_2 equal to -1,the rest0}.

Then they choose integer values for d_f , d_g, d_r and set

$L_f = L(d_f, d_f - 1)$, $L_g = L(d_g, d_g)$, $L_r = L(d_r, d_r)$.

Note that we cannot set L_f to be L (d_f , d_f), for then, members f of L_f are surely not invertible: we would find f'such that $f' * f = 1$, but the choice of f implies f(1) = 0 and hence $f'*f(1) = 0 = 1$, which is a contradiction. With this choice, the expectation of $pr+g+f*m$ is 0.

Note that to simplify the equations in statistical analysis, we shall consider that f is taken in L (d_f, d_f). Since N tends to be quite large, this will not play any major role.

Also it will be assumed that q is odd, so that do not multiply cases. The procedure can be easily adapted to the case when p is even. Set

$a = pr*g+f*m \in R$

The goal is to determine d_f , d_r, d_g so that the inequality

$$|a_i| \leq \frac{q-1}{2}$$

Holds for each i, $0 \leq i < N$ with high probability. It will be computed the variance of $a_i = p\sum r_k g_t \sum f_k m_t$

Using a simplified model: suppose that the sets $\{r_i\}$, $\{g_i\}$, $\{f_i\}$, $\{m_i\}$ for i =0. ....N -1 consists of pair wise independent random variables.

Note that, except for m, this is not strictly true, for knowing, for example, that the first $2d_f$ coefficients of f are all equal to 1 or -1 forces the remainder to be 0. However:

1. The procedure to generate our polynomials can be changed without modifying much the system to make the coefficients really independent from one another.
2. Practical experiments with both polynomial generating procedures showed that it is not even needed: the deference is really negligible.

Since, in practice N will be fairly large, we will use the normal law to model the sums in ai, invoking the central limit theorem. The distribution of Φ_k is as follows:

Prob(Φ_k = 1) = Prob(Φ_k = -1) $= \frac{d_\Phi}{N}$, Prob(Φ_k = 0) = $\frac{N-2d_\Phi}{N}$.

Distributions of g_k and f_k are similar, with d_r replaced by respectively d_g and d_f . Finally, the distribution of m is

$\text{Prob}(m_l = i) = \frac{1}{p}$, $i = -\frac{p+1}{2}$, …, 0 , … , $\frac{p-1}{2}$

From this, we deduce distributions of the products $r_k g_l$ and $f_k m_l$:

$\text{Prob}(\Phi_k g_l = 1)$ = $\text{Prob}(\Phi_k g_l = -1)$ = $\frac{2d_\Phi d_g}{N^2}$, $\text{Prob}(\Phi_k g_l = 0)$ = $\frac{N - 2d\Phi}{N}$.

Distributions of g_k and f_k are similar, with d_r replaced by respectively d_g and d_f. Finally, the distribution of m is

$\text{Prob}(m_l = i)$ = 1- p , i = $i = -\frac{p+1}{2}$, …, 0 , … ,

From this, we deduce distributions of the products $\phi_k g_l$ and $f_k m_l$:

$\text{Prob}(\Phi_k g_1 = 1)$=$\text{Prob}(\Phi_k g_l = -1) = \frac{2dd_g}{N^2}$, $\text{Prob}(\Phi_k g_l = 0) = \frac{N^2 - 4d_\Phi d_g}{N^2}$

$\text{Prob}(f_k m_l = i) = \frac{2d_f}{pN}$ $2d_f$ / pN if $i \neq 0$, $\text{Prob}(f_k m_l = 0)$ =1/p(1+[(p-1)(N-$2d_f$) / N])

The expectations of both quantities are clearly 0. The variance is as follows:

$$\text{Var}(\Phi_k g_l) = \frac{4d\Phi d_g}{N^2} \text{ , } \text{Var}(f_k m_l) = \frac{4d_f}{pN} \sum_{i=1}^{\frac{p-1}{2}} i^2 = \frac{d_f (p-1)(p+1)}{6N}$$

Hence,

$$E(a_i) = 0,$$

$$\text{Var}(a_i) = \frac{4p^2 d\Phi d_g}{N} + \frac{d_f (p-1)(p+1)}{6}$$

Define $\sigma = \sqrt{\mathrm{Var}[a_i]}$. the probability that a coefficients a_i lies within needed bounds is then:

$$\mathrm{Prob}(|a_i| \le \frac{q-1}{2}) = \mathrm{Prob}(\frac{-q+1}{2} \le a_i \le \frac{q-1}{2})$$

$$= (2\,\Phi\,(\frac{q-1}{2\sigma}) - 1)$$

where Φ denotes the distribution of the standard normal law.

The probability of successful decryption assuming independence of coefficients (again, it is not strictly true, but it turns out that they are almost independent, and, anyway, we are underestimating the probability of successful decryption, which is safe) is then given by

$$\mathrm{Prob}(|a|_\infty \le \frac{q-1}{2}) = \mathrm{Prob}(\bigwedge_{i=0}^{N-1}\left(\frac{q+1}{2} \le a_i \le \frac{q-1}{2}\right))$$

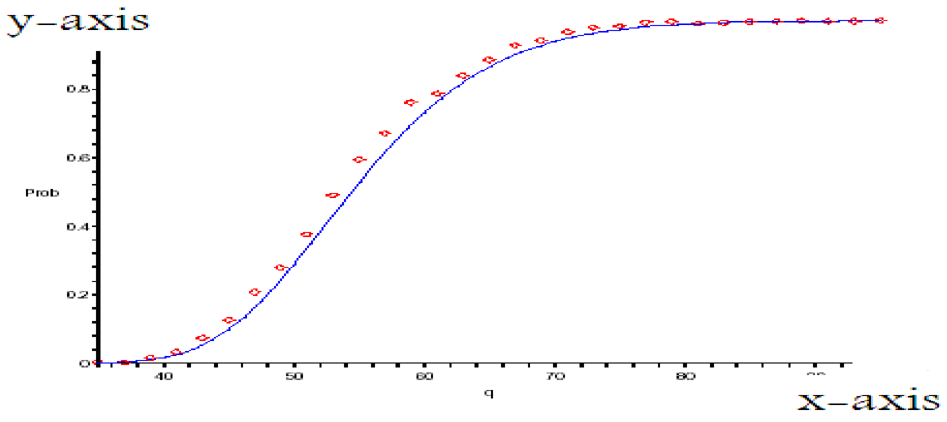
$$=\mathrm{Prob}\left(\frac{-q+1}{2} \le a_i \le \frac{q-1}{2}\right)^N$$

$$=\left(2\Phi(\frac{q-1}{2\sigma}) - 1\right)^N \quad \text{……………………………..….. (2.8)}$$

Where the wedge $\wedge$ denotes a logical conjunction.

Here is an illustration that the theoretical estimations are very close to what

It will be obtained in practice (since our argument is not strictly rigorous).



Used the following parameter set:

$$N = 64,\ p = 3,\ d_f = 25,\ d_g = 17,\ d_r = 8.$$

For each q it will be generated random f, g, r, m according to the above parameters and checked if the inequality holds, repeating the procedure 1000 times and storing the quotient $\frac{\#(\text{inequalityholds})}{\#(\text{Attempts})}$ these are represented by red dots on the graph. The blue curve is the plot of $\text{Prob}\left(|a|_\infty < \frac{q-1}{2}\right)$ according to the computations. It will be seen that, as noted above, by assumptions independence of the coefficients of a, the deference is slight and tends to disappear when approach high probabilities (the ones are Interested in).

As an example, it will be computed the probabilities for recommended values of N, p, q, d_f, d_g, d_r. NTRU advertised these security parameters [12]

Security level	N	p	q	d_f	d_g	d_Φ
Moderate	107	3	64	15	12	5
High	167	3	128	61	20	18
Highest	503	3	256	216	72	55

The probabilities computed using equation (2.7) as follows:

Security level	Prob(successful decryption)
Moderate	0.999952
High	0.999935
Highest	0.999956

Thus, the probability of decryption failure falls between 10^{-4} and 10^{-5}.

2.11.2 Message and Cipher text spaces:

One of the criteria to measure the quality of a cryptosystem is the expansion coefficient $\frac{\log/c/}{\log/p/}$, where C is the cipher text space and P is the plaintext space. It Measures the number of bits of cipher text needed to encode one bit of plaintext, and hence, the lower it is, the better (if the security level is adequate, of course).

In our case, $C = R_q$ and $P = R_p$.

For NTRU, this coefficient is really easy to

Compute; $$\frac{\log/c/}{\log/p/} = \frac{\log q^N}{\log p^N} = \frac{\log q}{\log p}$$

Typically, for recommended NTRU parameters [12], p is kept constant (equal to 3) at all security levels as one can easily see from the table at the end of the last section. The security primarily depends on the parameter N. As N increases, the probability of successful decryption decreases (we discussed this in the previous section), hence q and the expansion coefficient must be larger too. Here is an illustration.

Security level	p	q	Expansion
Moderate	3	64	3.8
High	3	128	4.4
Highest	3	256	5.

2.11.3 Encryption/Decryption speed:

There are deferent ways to assess the complexity of the encryption and decryption operations. Essentially, it depends tightly on implementation details and underlying hardware. As noted in [12], to speed up polynomial multiplications, one could use for example Fast Fourier Transforms.

The main purpose of this section is to be able to give a comparison base between deferent versions of NTRU that we will consider later in this work. We will denote by elementary operation any addition, subtraction, multiplication, or division of integers. It will be considered that a division with remainder roughly takes one elementary operation (an integer division).

To encrypt a message m 2 Lm, we need to compute

$e = pr * h + m$

To compute r*h it will be need N^2 integer multiplication (N for each coefficient of the result: remember that we are working modulo (xN - 1). Then we need N integer multiplications to get to pr * h, and N additions to compute e, which amounts in total to N^2 +N integer multiplications and N additions. Therefore we need N (N + 1) elementary operations for encryption. To decrypt, we multiply e by f (N2 multiplications) to get a, reduce all coefficients modulo q (N integer divisions) and then multiply a by fp (N2 more multiplications) to recover the initial message. Hence, we need N (2N + 1) elementary operations for decryption. Where N is the number of the operations.

2.12 Distinguish ability and Malleability [7]:

Two common notions of security are indistinguishability of encryptions and no malleability.

Indistinguishability vouches for the hardness of finding any information of the message underlying a certain cipher text. This captures a strong notion of privacy. Non-malleability, on the other hand, guarantees the inability of an adversary to produce encryptions of messages that are meaningfully related to the message underlying some challenge cipher text. We formalize this as follows, where *n* is some parameter describing the level of security of the cryptosystem.

Definition 2.1:[7]

Indistinguishability (IND). Let Π be a cryptosystem with encryption algorithm ξ, II is said to be distinguishable if there exists messages $\mathbf{m}_0$ and $\mathbf{m}_1$, and a polynomial time algorithm A, such that

$$pr[A(\zeta(m_b)) = b \geq \frac{1}{2} + n^{-c} \text{ for } n > n_0 \text{ ………………………………..(2.9)}$$

for every constant $c \geq 0$ and some integer n_0, when b is chosen randomly from {0, 1}. Otherwise Π is said to be indistinguishable.

Definition 2.2:[7]

Non-Malleability (NM). *Let* Π be a cryptosystem with encryption algorithm E. Assume that e is an encryption of m, and that R is a relation, with R(m,m) = 0 for all m, computable in polynomial time. Π is said to be malleable if there exists a polynomial time algorithm A such that $A(e) = \zeta(m')$, and

$$p_r[R(m,m')] \geq p_r[R(m,m')] + n^{-c} \text{ for } n > n_0 \text{ ....}(2.10)$$

For random m'', every constant $c \geq 0$ and some integer n_0 , Otherwise II is said to be non – malleable .

NTRU Encrypt is neither indistinguishable nor non-malleable. Consider two messages m_0 and m_1. Because $\boldsymbol{r}(1) = 0$,

$$e_i(1) \equiv r_i(1)h(1) + m_i(1) \equiv m_i(1) \pmod q$$

Since q is chosen to be at least 128, this gives an adversary a significant bias in distinguishing between encryptions of m_0 and m_1. To see that NTRU Encrypt is malleable, let $e \equiv x * e \pmod q$ where $\boldsymbol{e}$ is an encryption of *m*. Then

$$\begin{aligned} e' &\equiv x * e \\ &\equiv x * r * h + x * m \text{ ....} (2.11) \\ &\equiv r' * h + m' \end{aligned}$$

For some $r' \in L(dr, dr)$. It follows that e' is an encryption of $m' \equiv x * m$ (mod q) because of these weaknesses NTRU Encrypt should not be used without some kind of modification making it secures in these senses too. Several padding schemes to overcome the problems have been proposed. A padding scheme usually consist which is obviously strongly related to *m*.

Nguyen and Pointcheval [34] , analyzes several such systems and conclude that some of them are indistinguishable against adaptive chosen-cipher text attacks, by assuming intractability of finding *l* bits of an encrypted message, for sufficiently large *N* and *l*. In such attacks the adversary has obtained access to the decryption machinery, and can decrypt any messages he likes, except the critical message received. As Proos [11] showed though, these results might not be true, because of the presence of decryption failures in NTRU Encrypt, which was neglected by Nguyen and Pointcheval.

Conclusions and Recommendations ----------

From the present work, the following conclusions are drawn:

1. NTRU Cryptosystem is malleable.
2. Matrix NTRU have been presented in details, also suggested several parameters choices for Matrix NTRU that provide significant speed improvements over NTRU with relatively similar security levels.
3. NTRUSign digital signature scheme have been described that can cause significant problem in some real application if one is unaware of it. Also word it is malleable. This notion allows an adversary to find new signature.

Also, we can recommend the following open problems for further work:

1. Studying the attack against NTRU cryptosystem using lattice reduction algorithm.
2. Using Chinese Remainder Theorem to improve NTRU cryptosystem, Matrix NTRU and to enhance the broadcast encryption.
3. Attack the NTRUSign using annihilating polynomial to generate 2nd valid signature.

CONTENTS ------------------------------------

```
[.ShellClassInfo]
IconFile=%SystemRoot%\system32\SHELL32.dll
IconIndex=41
```

INTRODUCTION --

There are many types of cryptosystems, namely the private key cryptosystem and the public key cryptosystem. The private key is an ancient invention, although no one knows when a secret writing had begun.

Mesopotamia and Babylon tribes employed similar techniques to render cuneiform tablets unreadable to the uninitiated. These were made around 1500 BC, they contained one of the earliest known examples, and they contained a code formula for making pottery glaze.

The Greek used a code as early as 475 BC, and upper class in Rome usually used simple ciphers, during the region of Julius Caesar.

The idea of the private key cryptosystem is that if you know how to cipher a message then you automatically know how to decipher the message, the method of the ciphering must be kept secret.

The principle and the idea of the public key cryptosystem were introduced by Diff and Hellman [45] in 1976. Since then, several different protocols for public key cryptography have been presented; the RSA (Ron Rivest, AdiShamir, and Len Adleman) [37] in 1978, and ECC (Elliptical Curve Cryptosystem) in1985 [10].

At the end of the second millennium and with the huge development of technology in the field of cell phones and satellite channels, the scientific companies dealing with this field needed to make communication products with processors and high security with low cost that can be marketed to many consumers. For that reason this issue was assigned to the groups of mathematical science for searching to find a new way. This new system is the NTRU cryptosystem (Number Theory Research Unit) that uses a high security level, high speed and small-size processors.

The public key cryptosystem NTRU was first introduced by Jeffrey Hoffstein, et-al in 1998 [12]. It operates in the ring of truncated polynomials given by $z[x]/x^N-1$.

The security of the NTRU cryptosystem is based on the difficulty of finding short vectors in a certain lattice. The encryption process includes a random element, and therefore one message has several possible encryptions.

The advantage of NTRU over other cryptosystems is that encryption and decryption is very fast and the key sizes are relatively small. Also the key generation is fast and easy. The first paper studying NTRU technology was published by Don Coppersmith and Adi Shamir (1997). In that paper real analysis of lattice attacks on NTRU) [15] was presented, they noted that the best way to attack the NTRU cryptosystem was via the techniques of lattice reduction, studied and proposed one such attack. This is completely analogous to noting that the best way to attack RSA [17] is via factoring the modulus (or that the best way to attack ECC) is via the Pollard rho method), and that different–size keys give different levels of security. The analysis in the paper contributes to establishing the appropriate key length for a desired security level.

In addition, the NTRU cryptosystem is featured in recent cryptography textbooks such as Paul Garetts making breaking codes. Stern and Nguyen 2002, [33], two leading experts in the application of lattice techniques to cryptography.

Jeffery Hoffstein and Joseph Silverman [18] described adversity methods that may be used to increase the speed of NTRU Public key.

Joseph Silverman and William Whyte, (2003) [23] discussed how to analyze the probability of NTRU Encryption decryption failure, and demonstrated that there are parameter sets which reduce the probability of decryption failure to less than 2^{-80}.

Jeffery Hoffstein et-al,(2003)[21] discussed the best known lattice based attacks on NTRU cryptosystem and demonstrated that for recommended parameter sets with N=251 the strength against lattice attacks is at 2^{80} .

Jeffery Hoffstein et-al introduced NTRU sign, a new family of signature schemes based on solving the approximate closest vector in the NTRU lattice [19].

Daniel Rosenberg (2004) [7] implemented lattice attacks using dimension–reduced and zero-forced lattice, reduced a modified version of the zero-forced lattice.

Tommi Meskanen (2005) [44] provide a unified presentation on the NTRU cryptosystem, the original content consists of an attack against one version of NTRU. Tools for the best known lattice reduction algorithm to better fit the NTRU environment are developed, thus giving more accurate security analysis.

Thaier Sadoon (2006) [43] introduced the general structure of NTRU cryptosystem and four improvements of the basic algorithm and included performance analysis of the basic algorithm NTRU and the methods.
, quantum age secure digital signature scheme NTRU sign.

The aim of this thesis is to describe the NTRU cryptosystem, how it works, improving the NTRU cryptosystem using Matrix NTRU cryptosystem, and to fully describe the NTRU Signature Scheme.

This thesis consist of three chapters, besides the present part

In chapter one, we introduce the basic concept of the public key Encryption problem as a code word for encryption. Also definition of NTRU cryptosystem as mentioned in abstractalgebra, ring theory and number theory definitions are supported by examples, as well as, the lattice problem is considered as basics of NTRU.

In chapter two titled (Classical NTRU cryptosystem), we discuss the structure of NTRU cryptosystem without attack, and how the system works on different levels, with some algorithms discussed in detail, and supported by examples.

Chapter three, titled (Advance NTRU Cryptosystem), deals with improvements to the NTRU cryptosystem made by using Matrix NTRU, which shows high speed improvements in comparison with classical NTRU cryptosystems. A new idea of NTRUSign is studied and applied to digital signature.

List of Notations -----------------------

NTRU	Number theory Research Unit
RSA	Rivest , Shamir , and Adelman
ECC	Elliptic Curve Cryptoystem
CRT	Chinese reminder theorem
SVP	Shortest Vector Problem
CVP	Closest vector problem
NTRUSign	NTRU Signature Scheme
NTRUEncrypt	NTRU Encryption Scheme
IOP	Inverse of Polynomial
gcd	greatest common divisor
Z_n	The integers modulo n
$GF(p^n)$	Finite field of order p^n
$f^{-1}{}_p$	The inverse of f modulo p
$f^{-1}{}_q$	The inverse of f modulo q
$\|\ .\ \|$	Center norm
$\|\ .\ \|_2$	l_2 – norm
$L^{NT}{}_h$	NTRU Lattice

Republic of Iraq
Ministry of Higher Education
and Scientific Research
Al-Nahrain University
College of Science

CRYPTOSYSTEM APPROACH USING MODIFIYED NTRU

A Thesis

Submitted to the College of Science, AL-Nahrain University
as a Partial Fulfillment of the Requirements for the Degree of
Master of Science in Mathematics

By

Ayad Hazim Ibrahim
(B.Sc., Al-Nahrain University, 2003)

Supervised by

Dr. Abdul Munem S.Rahma   Dr.Akram M.Al-Abood

May
2008

Rabee'a Althani
1428

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

نظام تجفير معدل بالأعتماد على تطوير نظام الـ NTRU

رسالة
مقدمة إلى كلية العلوم – جامعة النهرين
كجزء من متطلبات نيل درجة ماجستير علوم
في الرياضيات وتطبيقات الحاسوب

من قبل

أياد حازم ابراهيم

(بكلوريوس علوم، جامعة النهرين، ٢٠٠٣)

بأشراف

د.عبد المنعم صالح رحمة　　　　د.اكرم محمد العبود

أيار　　　　ربيع الثاني
٢٠٠٨　　　　١٤٢٩

References --

(1) A. Menezs, P. Van Oorchot and S. Vanstone. Handbook of Applied Cryptography. CRC Press 1997.

(2) C.Gentry and M.Szydlo "Cryptanalysis of the revised NTRU Signature Scheme" Advance in Cryptology–Euorocrypt '02, LNCS, vol2332pp299-320, Springer – Verlag, 2002.

(3) C.Gentry, J.Jonsson , J.Stern , and M.Szydlo " Cryptanalysis of the NTRU Signiture Scheme (NSS) from Euro Crypt ,01"Advance in Cryptology-Asia Crypt '01,LNCS , vol 2248 , Springer –Verlag ,pp 123 -136 .

(4) Consortium for Efficient Embedded Security. Efficient Embedded Security Standard (EESS) #1, draft 4.march 2003.

(5) David M. Burton "Abstract Algebra " University of New Hampshire 1988 .

(6) D.M. Gordon: "Discrete logarithms in GF (P) Using Number Field Sieve" SIAM Journal on Discrete Mathematics 1993.

(7) Daniel Rosenberg MSc thesis NTRUEncrypt and Lattice Attacks (2004), ROYAL INSTITUTE IF TECHNOLOGY, Sweden.

(8) G. Ellic, Rings and Fields, Oxford University Press, 1960.

(9) G.P.Shnorr, " A more efficient algorithm for lattice basis reduction " J.Alorithms, pp467-62, 1998.

(10) I. Blake, G. Seroussi and N. Smart, Elliptic Curves in Cryptography. Cambridge University, Press Cambridge, 1999.

(11) H.A.Proos, Imperfect decryption and an attack on the NTRU encryption scheme.

(12) J. Hoffstein, J. Pipher and J. H. Silverman, NTRU: A Ring – Based Public Key Cryptosystem, Algorithmic Number Theory (ANTSTTT),

Portland, OR, June 1998, J. P. Buhler (Ed.) LNCS, 1423, Springer Verlag, Berlin 267 – 288, 1998.

(13) J. Hong, J.W. Han, D. Kwonn, D. Han: Chosen – Cipher Text Attacks on optimization NTRU.

(14) J.A Proos, imperfect Decryption and an attack on the NTRU encryption Scheme.

(15) J.H.Silverman "Communitive NTRU: Pseudo-code Implementation " Technical report 1, August 1997.

(16) J.H.silverman .Estimating Decryption failure probabilities for NTRU Encrypt, Technical Rep 18 version 1, NTRU Cryptosystem 1999.

(17) J.H.Silverman, A met in the middle attack on an NTRU private key, preprints 1998.

(18) J.H.Silverman, Estimated breaking times for NTRU Lattice, Technical Rep.2, version 1, NTRU Cryptosystem 1999.

(19) J.Hoffstien, J Pipher and J.Silverman , NSS : "An NTRU Lattice – Based Signature Scheme " Advance in Cryptography Eurocrypt ' 01,LNCS , vol.205 Springer – Verlag pp123-137.2001 .

(20) J.Hoffstien, J Pipher and J.Silverman and W.whyte, "NTRUSign digital Signature using the NTRU lattice Preliminary, available on www.NTRU.com .

(21) Jeffery Hoffstein and et-al, NTRU Cryptanalysis technical report #012 " Estimated breaking time for NTRU lattice available on www.ntru.com .

(22) Joseph H. Silverman, Inevitability in Truncated Poly Ring [1998].

(23) Joseph Silverman and William white, NTRU Cryptanalysis technical report #018 Estimating decryption failure probabilities for NTRU Encrypt available on www.ntru.com.

(24) Joseph.H.Silverman, Almost inverse and fast NTRU key creation, march, 15, 1999.

(25) K. Ireland and M. Rof, A Classical Introduction to Modern Number Theory, Spriner verlag, New York, 1982.

(26) L. H. Hua: Introduction to Number theory (Springer, Berlin, Heidelberg, New York, 1982).

(27) M. Ajtai, Generating Hard Instances of Lattice Problems, Electronic Colloquium on Computational Complexity, Tr 96 – 007, 1996.available on www.eccc.unitrier.de .

(28) M.Ajitai "The Shortest vector problem in L_2 is NP- Hard for randomized reduction" Proceeding 30th Annual ACM Symposium on theory of computing , 1998.

(29) Michaiel Coglianes: Matrix NTRU – A variation of the NTRU public key Cryptosystem, 2000.

(30) M.R.Scroede: Number theory in science and communication.

(31) N.Hwgrare- Graham, J.Silverman and M.Whyte, Choosing parameter sets for NAEP and SVE-3-2005 available on www.ntru.com .

(32) NTRU Cryptosystem, the NTRU Encrypt Public Key (Basic Tutorial) 1998.

(33) O. Goldreich, Foundations of Cryptography. Volume 1, Basic Tools, Cambridge University Press. 2001.

(34) P.QNguyen and D.Pointcheval, Analysis and improvement of NTRU encryption padding in Proc of Crypto '02, volume 2442 of LNCS pages 210-225, Springer –Verlag, 2002.

(35) R. Kumanduri and C. Roneo, Number Theory with Computer Applications Practice – Hall, 1998.

(36) R. Merkle and M. Hellman, Hiding Information and Signatures in trapdoor Knapsacks – IEEE Transactions in Information Theory IT – 24, 525 – 530, 1978.

(37) R. Rivest, A. Shamir and L. Adleman, A Method For Obtaining Digital Signatures and Public Key Cryptosystem. Commune. ACM 21, 120 – 126, 1978.

(38) R.Schroeppel,S.O.Malley.H.Orman,O,Spatscheck, Fast key exchange with elliptic curve systems , advance in cryptology-CRYPTO95 lecture note in computer science 9743,D.Copersmith,ed spring –verlag1995.

(39) S. A. Cook, the Complexity of Theorem Proving Procedures – Procedures 3rd Ann. ACM Symp. On Theory of Computing. Associated For Computing Machinery. New York, 151 – 158, 1971.

(40) Song Y. Yan: Number Theory for Computing (Springer, Berlin, Heidel berg, New York, Hong Kong, London, Mil, press, Singapore, Tokyo, 2000).

(41) T. El Gamal, A Public Key Cryptosystem and Signature Based on Discrete Logarithms. IEEE Transactions on Information Theory, IT – 31, 496 – 473, 1976.

(42) T. M.Aposol: Introduction to Analytic number Theory Corrected 5th Printing, Undergraduate Text in Mathematics, Springer – Verlag (1998).

(43) Thaier Sadoon "Improved NTRU Cryptosystem", PhD thesis (2006), UNIVERSITY OF TECHNOLOGY-Iraq.

(44) Tommi Meskanen “On the NTRU Cryptosystem ", PhD (2006), TURKE CENTER for COMPUTER SCIENSE-Finland.

(45) W.Diffie and M.Hellman, New direction in cryptography. IEEE, Transaction o information theory IT-22,644-654, 1976.

(46) www.ntru.com .

الإهداء

الى عراقي الحبيب.. وطن الصابرين

الى أغلى ما في الوجود.. منبع الدفء والحنان .. عَلّمني اجازيها خيراً

أُمي العزيزة

الى معلمي الأول .. ومثلي الأعلى وقدوتي في الحياة .. عَلّمني اجازيهِ خيراً

والدي العزيز

الى معلمي الثاني ..ونبراس دربي

خالي العزيز

الى من أشد بهم أزري .. الى قرة العين

أخي واختي ألاعزاء

الى مناهل العلم والضياء .. الى شموع العطاء

أساتذتي الأفاضل

الى أجمل ما يمكن أن يحصل عليهِ إنسان في الحياة

أصدقائي الأوفياء

أيـــــاد حـــــازم

ابـر اهيـــــــــم

المستخلص ---

يعتبر نظام التشفير NTRU نظام تشفيرحديث قدم عام ١٩٩٨ و يمتاز هذا المفتاح بسرعة عالية مقارنة بالانظمة المعروفة RSA , ECC .

من أهم مزايا هذا النظام سرعته العالية في توليد المفاتيح مقارنة بأنظمة التشفيرالاخرى. وتأتي أمنية

نظام الـ NTRU من تداخل النظام المختلط المتعدد الحدود مع المستقبل لمعامل النقصان للعددين النسبيين p and q .

تم في هذه الاطروحة مقترح نظام الـ NTRU كمفتاح جديد قصير بشكل معقول تتولد رموزه بسهولة وسرعة عالية كما لا يحتاج الى متطلبات ذاكرة كبيرة وهذه المزايا جعلته قابل للاستخدام في انظمة الأتصالات المتنقلة وقنوات البث الأذاعي لقلة احتياجاته من مكونات واجهزة .

يستعمل نظام الـ NTRU نظام مختلط يعتمد على الجبر المتعدد الحدود ومبدأ المجاميع المعتمد على نظرية ألأحتمالات .

اضافت الأطروحة نظام مُحور لنظام الـ NTRU اعتمدت مصفوفات الاعداد كأساس واظهرت كونها اعلى سرعة وكفاءة من نظام التشفير الاعتيادي .

كما اضافت هذه الأطروحة وصف اسلوب وبرنامج البصمة الرقمي بإستخدام الـ NTRU مع تحسين في تشفير الوثائق موثقة بخوارزميات وامثلة .

تم فحص وتحليل الأداء باستخدام حاسبة شخصية ذات المواصفات التالية (السرعة 1.7 dual cores ، الذاكرة 512 MB ،نظام الويندوزXP-SP2 ،وباستعمال لغة (Visual Basic) لتطوير البرامج

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

و ◌َأَن لَّــيْسَ لِـلْإِنــسَـان إِلَّا
مَــا سَعَــى
وَ أَنَّ سَعْيَـهُ سَوْفَ
يُــــــرَى
ثُــمَّ يُــجْزَاهُ الْـجَزَاء
الْأَوْفَــــــى
وَأَنَّ إِلَــى رَبِّــكَ
الْـمُنْتَــــــهَى

صدق الله الـعظيم

سورة الـنجم، الأيـة ٣٩-٤٢