Analytical study of Implementation issues of NTRU

Praveen Gauravaram
Tata Consultancy Services
Hyderabad

Harika Narumanchi
Jawaharlal Nehru Technological University
Hyderabad

Nitesh Emmadi
International Institute of Information Technology
Hyderabad

***Abstract*—N^{th}-Dimensional Truncated Polynomial Ring (NTRU) is a lattice-based public-key cryptosystem that offers encryption and digital signature solutions. It was designed by Silverman, Hoffstein and Pipher. The NTRU cryptosystem was patented by NTRU Cryptosystems Inc. (which was later acquired by Security Innovations) and available as IEEE 1363.1 and X9.98 standards. NTRU is resistant to attacks based on Quantum computing, to which the standard RSA and ECC public-key cryptosystems are vulnerable to. In addition, NTRU has higher performance advantages over these cryptosystems.**

Considering this importance of NTRU, it is highly recommended to adopt NTRU as part of a cipher suite along with widely used cryptosystems for internet security protocols and applications. In this paper, we present our analytical study on the implementation of NTRU encryption scheme which serves as a guideline for security practitioners who are novice to lattice-based cryptographic implementations. In particular, we show some non-trivial issues that should be addressed towards a secure and efficient NTRU implementation.

I. Introduction

Since the invention of public-key cryptography by Diffie-Hellman [17] and Merkle [28], significant scientific work has taken place in the field of cryptography that led to the invention of several public-key cryptosystems for various information security applications. The most notable ones are the invention of RSA cryptosystem [37] by Rivest, Shamir and Adelmann and Diffie-Hellman key-exchange protocol by Diffie and Hellman [17]. Since then many public-key cryptosystems such as RSA and digital signature algorithm (DSA) [31] have been standardized and deployed for various applications. Often (as is the case with RSA, ElGamal [18], Diffie-Hellman protocol), the security of these cryptosystems is based on solving some hard mathematical problem. Integer factorization and discrete log problem (DLP) are the two common hard mathematical problems to which the security of several public-key cryptosystems has been reduced to. For example, breaking RSA encryption scheme requires factoring a large integer into its prime factors and breaking ElGamal encryption scheme or DSA requires solving DLP.

While these hard problems (depending on the selected security parameters) are far too impractical to solve with modern-day computers, they were shown to be solvable in polynomial time with quantum computers as shown by Shor [38]. However, quantum computers that are efficient enough to solve these problems (to break the security levels of cryptosystems) are not ready yet. As pointed out in a recent article [15], commercial quantum computers that can solve complex problems are expected to be available in two decades.

Therefore it is essential for the industry and academic institutions to be proactive with well-studied designs that are resistant against quantum computing attacks and their implementations to minimize or completely eliminate the impact of any future quantum computing attacks on standard cryptosystems. In this direction, the science of cryptography has progressed towards designing cryptosystems based on lattice-based problems that are analysed to be immune against quantum computing attacks [42]. Several lattice-based cryptographic designs have been designed and analysed. One of the most important designs in this area is NTRU cryptosystem (both encryption and digital signing mechanism). NTRU was designed by Silverman, Hoffstein and Pipher and later patented by NTRU Cryptosystems Inc. (which was later acquired by Security Innovations). NTRU has also been standardized by IEEE in IEEE 1363.1 [42] and by ANSI in X9.98 [11].

Unlike RSA and DSA that are relatively easier to follow for security application developers, NTRU might be more complex to understand and implement for developers who are not experienced in lattice based cryptography. It is also interesting to note that while there are many open source implementations of RSA and DSA [36] cryptosystems(for example, OpenSSL [5], PuTTY [6], OpenPGP [4]), we came across only very few open source implementations of NTRU [3]. In addition, there have been several interesting implementation results for NTRU targeted towards specific platforms [21] and applications [30]. In contrast, our paper is targeted towards software security practitioners who are beginners to lattice-based cryptography implementations. Hence, our implementation has not been targeted towards any specific platform or application, such implementations would still benefit from our study.

Given this importance of NTRU, we have considered the study of design and analysis of NTRU cryptosystem. In this direction, we have implemented NTRU encryption scheme on a standard computer with 2 GHz Intel Core 2 Duo Processor running a linux operating system utilizing 2 GB Random Access Memory (RAM). In this paper, we show some non-trivial technical issues that a developer need to consider carefully during NTRU implementation. These issues include using a secure random generator to generate the private-key, using efficient algorithms for fast key generation, choosing appropriate and secure padding schemes and choosing security parameters that are crucial for implementing secure and efficient NTRU cryptosystem. Although the IEEE 1363.1 document [42] describes the security considerations of NTRU, we have pointed out a few additional technical issues that complement the description of the IEEE document.

We expect that this analytical study would be a useful guideline for the security practitioners (especially who are novice to cryptographic implementations and NTRU) in the industries that wish to make NTRU as part of a cipher suite

TABLE I: Notation used in the paper

Notation Symbol	Description
$\mathbb{R}$	set of all real numbers
$\mathbb{Z}$	set of all integers
$\mathbb{R}^n$	set of all n-tuples of real numbers
$\|\|u\|\|$	Euclidean Norm/Length of vector u
B	set of Basis vectors
$L(B)$	Lattice generated by basis B
$\equiv$	Congruence
$\circledast$	Convolution product
MHz	megahertz
GB	gigabyte
c	ciphertext
m	plaintext/message

in their protocols and applications.

The rest of the paper is organised as follows: In section II, we describe the notation used in this paper. In section III, we introduce lattice-based cryptography. In section IV, we outline NTRU encryption scheme. In sectionV, we illustrate the methodology and various prototypes we have used while implementing NTRU. In section VI, we describe technical issues that a developer needs to consider while developing NTRU to yield a easy to understand as well as efficient implementation.

II. Notation and Definitions

This section describes the notation used in this paper and the definitions required for understanding the basics of lattice based cryptography. Table I describes the general notation used in the paper.

Lattices are mathematical objects that have many interesting properties for constructing cryptosystems or for cryptanalysis. Before going into the issues related to lattices, we recall some important definitions.

A. Basis

A basis is a set of linearly independent vectors that in a linear combination represents every vector in a given vector space. Given a basis of a vector space, every element of the vector space can be expressed uniquely as a finite linear combination of basis vectors.

Example: Let $\mathbb{R}^2$ be the vector space of all coordinates (a, b) where a and b are real numbers. Then the basis vectors are $e_1 = (1, 0)$ and $e_2 = (0, 1)$. Suppose $v = (a, b)$ be a vector in $\mathbb{R}^2$, then $v = a(1, 0)+b(0, 1)$. Any two linearly independent vectors like $(1, 1)$ and $(-1, 2)$ will also form a basis for $\mathbb{R}^2$.

B. Lattice

A lattice L is a set of points in n-dimensional space with periodicity property and L is defined as follows:

$$L = \{a_1v_1 + ... + a_nv_n \mid a_i \ integers\}.$$

Also denoted by $L(B)$ where B is an $n \times n$ matrix with columns as basis vectors $v_1, ..., v_n$. Equivalently, a lattice is a discrete additive subgroup of $\mathbb{R}^n$. Any point in a lattice can be represented by a linear combination of its basis vectors.

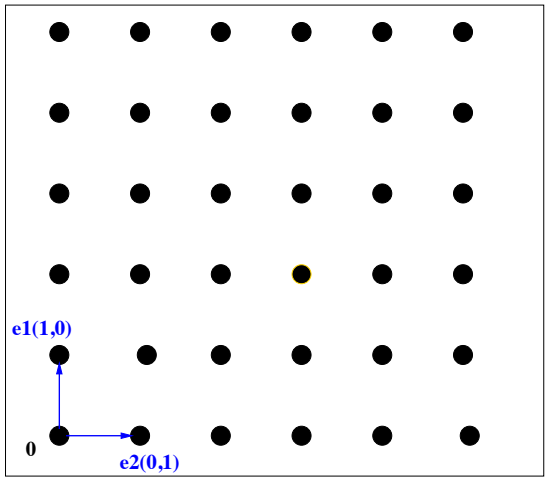

Fig. 1: 2-D Lattice with Basis vectors $e1$ and $e2$

Figure 1 shows a 2-Dimensional lattice with two basis vectors $e1$ and $e2$. The basis vectors $e1$ and $e2$ can be used to generate any point in this lattice. A lattice can have different sets of basis vectors. These sets of basis are equivalent to each other. Lattice can be of any dimension and as the dimension increases there are some interesting properties that can be used in cryptographic constructions.

III. Lattice Based Cryptography

The cryptosystems built on lattices are based on hardness of lattice problems[See section III-A] which are average-case hard problems. These cryptosystems are provably secure based on these hard problems. Furthermore, easier computational complexity adds to the efficiency of lattice-based cryptosystems [22].

A. Lattice Problems

Lattice problems are a class of optimization problems on lattices. In an optimization problem, we find the best solution from all possible solutions. In this section, we define two lattice problems called Shortest Vector Problem (SVP) [9] and Closest Vector Problem (CVP) [29]. SVP and CVP are worst-case hard problems. Approximating lattice problems to run within polynomial time is very hard. Time complexity of best known algorithm for solving lattice problems is 2^n [10]. The core hard problems used in building lattice-based encryption schemes and signature schemes [17] are SVP [22] and CVP [23] respectively.

SVP: Given a basis B for a lattice, find a non-zero vector $u \in L(B)$, such that $||u|| = \ \gamma * min||v||$ where $v \in L(B)$ and γ is the approximation factor.

Lenstra-Lenstra-Lovàsz (LLL) lattice basis reduction algorithm [12] produces a *relatively shortest vector* in polynomial time but does not solve SVP.

Figure 2 shows a 2-D lattice generated by basis vectors $V1$ and $V2$ and short vector u of that lattice.

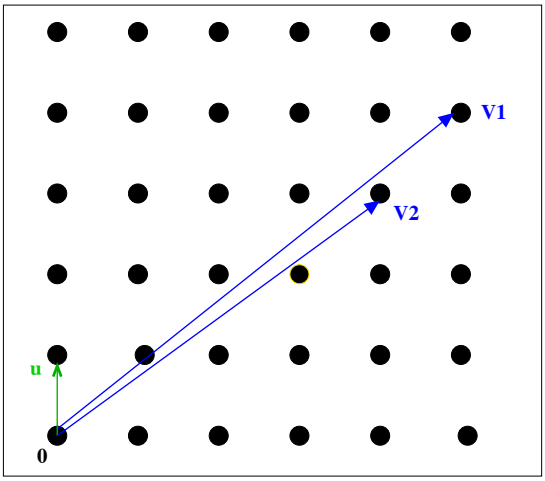

Fig. 2: Shortest Vector Problem

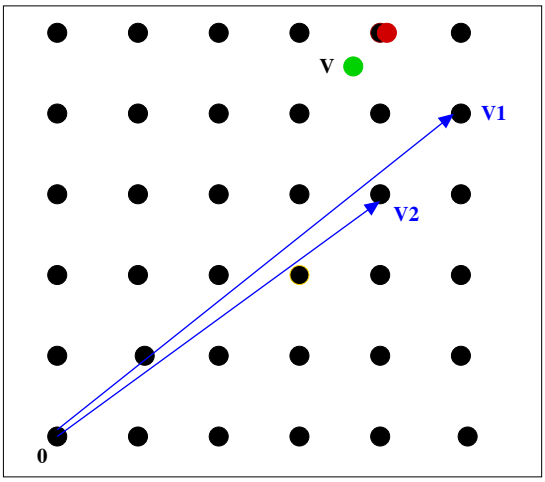

Fig. 3: Closest Vector Problem

CVP: Given a basis B and a point V, find a lattice point that is atmost γ times farther than the closest lattice point.

In Figure 3, given a lattice generated by basis vectors $V1$, $V2$ and a point V outside the lattice, the problem is to find a lattice point that is close to the point V.

B. Use of Lattice Problems in Cryptography

For most cryptographic algorithms, worst-case hardness forms the basis for security proofs [27]. However, many hard problems are only worst-case hard and lack this average-case hardness property. Lattice problems are proven to be average-case hard which has turned the interest towards building cryptographic algorithms whose security can be reduced to solving these problems. The worst-case hardness of average-case lattice problems can be used to create secure cryptographic schemes that are resistant to quantum computers. If one succeeds in solving lattice problems with small probability, any instance of these problems can be solved [34].

IV. NTRU Cryptosystem

NTRU was designed by Hoffstein, Pipher and Silverman [22]. NTRU is a lattice-based cryptosystem which is resistance to quantum computing and lower computational complexity compared to RSA. To encrypt a point in NTRU, we select a random point p in N-dimensional space and add a message vector to it. In decryption, the ciphertext point is mapped back to the lattice point and the message is recovered as the difference between the cipher point and the point p . As this lattice point is used to mask the message, it is called *masking point* .

1) Parameters: Table II describes the parameters used in NTRU algorithm.

TABLE II: NTRU Parameters

Parameter	Description	Knowledge
N	No. of coefficients in a polynomial	Public
q	Large Modulus to which coefficients are reduced	Public
p	Small Modulus to which coefficients are reduced	Public
f	Private key polynomial	Private
g	Used for generation of public key	Private
h	Public key polynomial	Public
r	Random blinding polynomial	Private
d_f	No. of coefficients with value 1 in polynomial f	Public

2) NTRU Key Pair Generation: Suppose that the decryptor wants to create a key pair. The decryptor follows the steps below to create an NTRU key pair:

- (i) Generate polynomial f such that the no. of coefficients with value 1 is equal to parameter d_f and f is invertible $modulo\ p$ and $modulo\ q$.
- (ii) Generate a random polynomial g.
- (iii) For the polynomial f, find inverse of $f\ modulo\ q$ and inverse of $f\ modulo\ p$ that have properties $f \circledast f_q \equiv 1(mod\ q)$ and $f \circledast f_p \equiv 1(mod\ p)$ [39].
- (iv) Compute $h = p \circledast f_q \circledast g$ (mod q).
- (v) The decryptor's key pair is: *Public Key:* h *Private key:* f

The NTRU key pair generation is illustrated in Figure 6

3) NTRU Encryption: Suppose the encryptor wants to send message to the decryptor.

The encryptor follows the steps below to encrypt a message:

- (i) Represent the message as a polynomial with its coefficients in the interval $[-p/2, p/2)$.
- (ii) Generate a random blinding polynomial r.
- (iii) The encryptor uses m, random chosen polynomial r and public key h to compute cipher polynomial using the formula: $e = r \circledast h + m\ (mod\ q)$.
  The cipher polynomial e is transmitted to the decryptor.

4) NTRU Decryption: Suppose that the decryptor received a message e from the encryptor and wants to decrypt it using his private key f. To do this The decryptor should have precomputed f_p. The modular reduction in decryption uses center lifting.

The decryptor follows below steps to decrypt the cipher polynomial:

- (i) Compute the polynomial a as follows: a = $f \circledast e$ $(mod\ q)$
- (ii) Calculate polynomial b as follows: b = a $(mod\ p)$
- (iii) Now, the decryptor recovers the plain text as follows: m = $f_p \circledast b$ $(mod\ p)$

5) Why decryption works: Given: ciphertext e and private key f, f_p. To derive: message m from ciphertext e.

e = $r \circledast h$ + m $(mod\ q)$ $\implies$ $r \circledast p \circledast f_q \circledast g$ + m $(mod\ q)$ By multiplying with f we get: $\implies$ $(r \circledast p \circledast f_q \circledast g$ + m $(mod\ q)) \circledast f$ Since, the coefficients of f and m are small, the modulo operation produces $f \circledast m$. $\implies$ $r \circledast p \circledast g$ + $f \circledast m$ By multiplying with f_p and modulo p we get: 0 + m $\implies$ m

V. Implementation description

In this section, we present modules we have used during the implementation of NTRU cryptosystem based on IEEE standard 1363.1 [42]. These modules form the core components of the design and their in-depth understanding is essential for the developers to be able to implement NTRU. Moreover, some of these modules employ algorithms that are more efficient than others for similar computations, thus, influencing the overall efficiency of the cryptosystem.

(i) *Polynomial polyMul(Polynomial f,Polynomial g)*

The *polyMul()* function calculates the convolution product of two polynomials and returns the resulting polynomial. The formula used for multiplication is as follows:

$c_k = \sum_{i=0}^{k} a_i b_{k-i} + \sum_{i=k+1}^{N-1} a_i b_{N+k-i}$

where a and b are the polynomials to be multiplied and c is the resultant polynomial.

(ii) *Polynomial generateF(int N, int df)*

The *generateF()* function generates a random polynomial f of degree N with df number of coefficients with value 1. The polynomial f should be invertible modulo p and q. Here, it is very important to use cryptographically secure random generator to avoid specific attacks related to random numbers. The parameter df is introduced to increase the probability of inverse existence and decrease the probability of decryption failures. The generation of polynomial f is illustrated in figure 4.

(iii) *Polynomial generatePolynomial(int N)*

The *generatePolynomial()* function generates a random polynomial of degree N. This function is used in generation of polynomials r and g which are used in public key generation. The random generator should be cryptographically secure. The polynomial generation is illustrated in figure 5.

(iv) *int fqinverse(Polynomial f, Polynomial fq, int q)*

The *fqinverse()* function finds the inverse modulo q of polynomial f if exists and stores it in the polynomial fq and returns 1 as success. Otherwise, it returns 0 as failure indication. The algorithm used in this function is Almost inverse algorithm [39] described in algorithm 1.

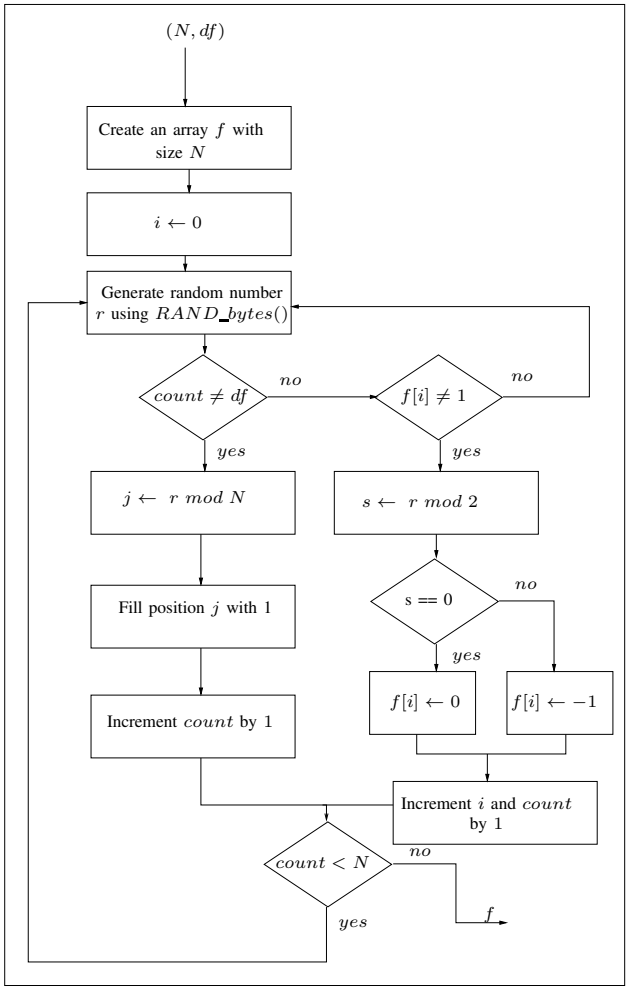

Fig. 4: Generation of f polynomial

(v) *int fpinverse(Polynomial f, Polynomial fp, int p)*

The *fpinverse()* function finds the inverse modulo p of polynomial f if exists and stores it in the polynomial fp and returns 1 as success. Otherwise, it returns 0 as failure. The algorithm used in this function is Almost inverse algorithm [39] described in algorithm 2.

(vi) *Polynomial publicKey(Polynomial g, Polynomial fq, int p, int q)*

The *publicKey()* function takes inverse fq, polynomial g, moduli. It calculates the public key and returns the public key polynomial.

(vii) *Polynomial encrypt(Polynomial m, Polynomial r, Polynomial h, int q)*

The *encrypt()* function takes message, random polynomial r, public key h and large modulus. It calculates the ciphertext and returns the ciphertext polynomial. The random polynomial is introduced in-order to make the encryption probabilistic in nature.

Algorithm 1 Algorithm for finding inverse of f modulo q

Input: Polynomial f and modulus q
Output: Inverse $b(x)$
$k = 0$
$b(x) = 1, c(x) = 0, f(x) = a(x), g(x) = x^N - 1$
loop:
while $f_0 == 0$ **do**
    $f(x) := f(x)/x$
    $c(x) := c(x) * x$
    $k := k + 1$
if $f(x) == 0$ **then**
    Cyclically shift $b(x)$ by k places
    return $b(x)$
if $deg(f) < deg(g)$ **then**
    exchange f and g and exchange b and c
if $f_0 == g_0$ **then**
    $f(x) := f(x) - g(x)(mod\ 3)$
    $b(x) := b(x) - c(x)(mod\ 3)$
else
    $f(x) := f(x) + g(x)(mod\ 3)$
    $b(x) := b(x) + c(x)(mod\ 3)$
goto *loop*
$q = p$
while $q < p^r$ **do**
    $q = q^2$
    $b(x) := b(x)(2 - a(x)b(x))(mod\ q)$

Algorithm 2 Algorithm for finding inverse of f modulo p

Input: Polynomial f and modulus p
Output: Inverse $b(x)$
$k = 0$
$b(x) = 1, c(x) = 0, f(x) = a(x), g(x) = x^N - 1$
loop:
while $f_0 == 0$ **do**
    $f(x) := f(x)/x$
    $c(x) := c(x) * x$
    $k := k + 1$
if $f(x) == 0$ **then**
    Cyclically shift $b(x)$ by k places
    return $b(x)$
if $deg(f) < deg(g)$ **then**
    exchange f and g and exchange b and c
if $f_0 == g_0$ **then**
    $f(x) := f(x) - g(x)(mod\ 3)$
    $b(x) := b(x) - c(x)(mod\ 3)$
else
    $f(x) := f(x) + g(x)(mod\ 3)$
    $b(x) := b(x) + c(x)(mod\ 3)$
goto *loop*

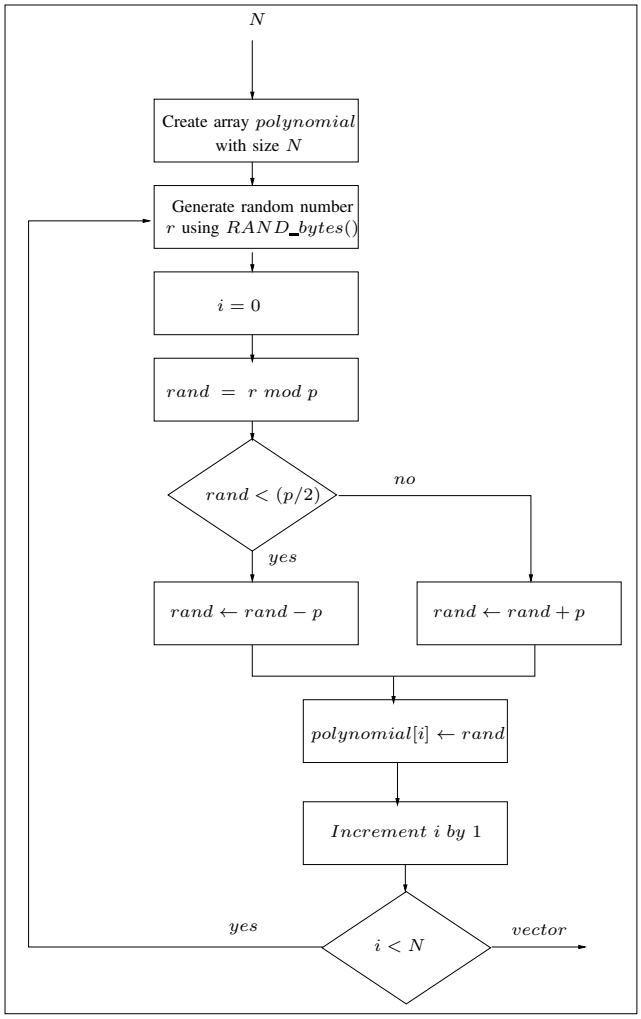

Fig. 5: Generation of polynomial

(viii) *Polynomial decrypt(Polynomial cipher, Polynomial f, Polynomial fp, int p, int q)*

The *decrypt()* function takes cipher polynomial, private key f, f_p and moduli. It decrypts the cipher and returns the message polynomial.

VI. Technical Issues

In this section, we discuss technical issues that we came across during the implementation of NTRU. Considering these issues while implementing NTRU will make the encryption less vulnerable to attacks and more efficient. Each issue addresses specific problem related to the overall security or efficiency of the system.

(i) *Secure Random Generator:*

The security of NTRU depends on the secret key vector f that is known only to authorized systems or personnel. The design of random function that generates f should be cryptographically secure to withstand specific attacks like direct

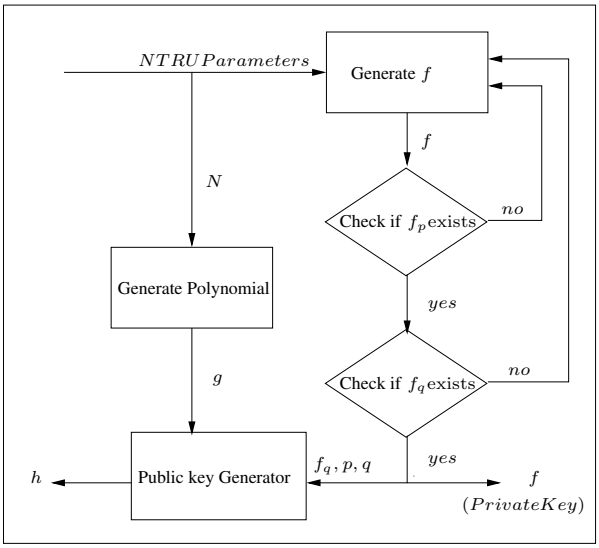

Fig. 6: Key Pair Generation for NTRU.

cryptanalytic attack, input-based attack and state compromise extension attack [26]. We have used $RAND\_bytes()$ function from OpenSSL 1.0.1e. The random generator of OpenSSL [5] is implemented according to National Institute of Standards and Technology(NIST) standard SP 800-90A [13]. Although OpenSSL has been updated since the release of 1.0.1e (the current version being OpenSSL 1.0.1h) we remark that these latest versions have not changed $RAND\_bytes()$ function from the one we have used. Developers should be careful in choosing the version of OpenSSL for $RAND\_bytes()$ functionality and it is recommended to use the latest version. In particular, $RAND\_bytes()$ function from OpenSSL 0.9.8c-1 up to versions before 0.9.8g-9 on Debian-based operating systems [8] and OpenSSL FIPS Object Module 1.1.1 [7] should be avoided as a security flaw was exposed in its implementation. It is recommended to use $RAND\_bytes()$ with secure hash algorithm-256 (SHA-256) [16] to generate random values.

(ii) *Fast Key Generation:*

The most computationally expensive part of NTRU cryptosystem is key generation. When we use matrix based approach for inverses, the key generation is extremely slow as we have to compute determinant and inverse of $N \times N$ matrix. We could find inverse for a polynomial of maximum degree N=11 using this method. Following this method, we used GNU Scientific library [1] for obtaining inverse and determinant for polynomials upto degree N=40 which is not sufficient to guarentee minimum security level. According to IEEE P1363.1 [42], minimum degree to guarentee the security of NTRU is N=401. When we used the Almost inverse algorithm [39], we could find the inverse of polynomial with any degree.

(iii) *Nature of q:*

If the value of q is taken as power of 2 and greater than $N^2/9$ we can certainly avoid decryption failures. Selecting lower values of q will have some risk of decryption failures.

(iv) *Relation between p and q:*

The selection of p and q should be in such a way that they are co-primes. If p divides q, retrieving m will be obvious as $p \circledast r \circledast h + m \ (mod\ q)$ will result in m. If p and q are not co-primes, then the reduction modulo of common factor will produce m.

(v) *Choose appropriate and secure padding scheme:*

The strength of NTRU lies in padding scheme. Failing to select an appropriate padding scheme may result in various attacks. Though careful selection of parameters help protecting the NTRU system against brute force and man-in-the-middle attacks, there are some other attacks which can compromise a system. Jaulmes and Joux [25] had demonstrated a chosen ciphertext attack (CCA) on NTRU which can retrieve the secret key of the system. The attack uses polynomials built specially from the public key which are then sent as input to the decryption algorithm which helps in retrieving the private key. The attack is based on the fact that NTRU system is not plaintext aware, that is, the attacker can build the ciphertexts without knowing the corresponding plaintexts. A way to avoid such attacks is to induce randomness to the message by using schemes such as Optimal Asymmetric Encryption Padding (OAEP) [14] as was done for RSA cryptosystem. However, OAEP does not fit the requirement as it can produce invalid messages. There are some other padding schemes proposed like Rapid Enhanced-security Asymmetric Cryptosystem Transform (REACT) [32], REACT2, PAD3 [33], and these padding schemes were shown to be insecure [33]. A padding scheme called NAEP [24] makes NTRU more resilient to such attacks and NTRU-NAEP is also provably secure.

(vi) *Choose appropriate and secure degree N:*

It is very important to choose degree N according to IEEE standard 1363.1 as lattice based attacks are possible on smaller degrees. The LLL lattice reduction algorithm can be used to retrieve the private key vector. It is necessary to choose N as prime as the cryptosystem with N as composite number was broken [19].

(vii) *Choosing appropriate security parameters:*

The parameters N, p, q and df define the security parameters of NTRU. Choosing them appropriately is crucial to avoid decryption failures and lattice based attacks. The IEEE P1363.1 [42] document defines different parameter sets for different levels of security. It is advisable to follow these parameter sets while developing NTRU.

VII. Why NTRU is faster than RSA?

In this section, we discuss the some significant reasons that makes NTRU cryptosystem faster than RSA.

- (i) The convolution product of polynomials in NTRU involves computations with smaller coefficients i.e. the product of two coefficients are added to get the resultant coefficient using the formula described in section V whereas RSA requires exponentiation operation that involves series of multiplications.
- (ii) All coefficients in NTRU polynomial are atmost 11-bit integers as they are reduced to $mod\ q$ (q is taken

as 2048). Therefore, there is no need of any multi-precision libraries for the computations. Thus, the computational cost is reduced.

(iii) The random blinding polynomial r used in NTRU has coefficients -1 , 0 and $+1$. As the coefficients are small, the convolution product becomes simple unlike RSA which has larger numbers that are to be exponentiated inorder to get the result.

(iv) The modulus reduction involves division operation that is computationally expensive. As the parameter q is a power of 2, the modulus can be calculated with logical AND operation i.e. a mod 2^i=a & (2^i -1) [2]. This avoids the use of division for modular reduction and makes the computations less expensive. For example, to calculate $46\ mod\ 8$ two addition operations and two multiplication operations are required using the long division method but this can also be caculated using single logical AND operation which reduces the number of operations used during modulus calculation.

VIII. Conclusion

In this paper we have addressed key implementation issues and challenges that we came across during the implementation of NTRU encryption scheme. As part of this we have pointed out some efficient algorithms from the literature such as Almost Inverse Algorithm [39] to find inverses during the NTRU key generation and a simple algorithm for calculating convolution product of polynomials. We believe that these resources provided in the paper can be utilized by the developers who are novice to NTRU design and implementation aspects.

It has been shown that fully homomorphic encryption (FHE) [20] schemes can be designed from NTRU [35], [40], [41]. The design, analysis and implementation of FHE is an active research area at the moment. Especially, considering its importance for cloud computing security, R&D companies like IBM and Microsoft are also investing for research in this area. In this context, our paper provides several fundamental aspects regarding the implementation of NTRU for the developers who targets implementing FHE schemes based on NTRU. As part of our future work, we focus on developing an optimal NTRU cryptosystem and use it as a core technology for future work on FHE.

IX. Acknowledgment

We would like to thank Dr. William Whyte for answering our questions on NTRU design and implementation aspects during the course of this project. We thank anonymous reviewers for their comments and suggestions which has helped in improving the paper.

References

[1] GNU Scientific Library Reference Manual. This manual is available at http://www.gnu.org/software/gsl/manual/html_node/ (Accessed On 2/6/2014).

[2] Modulus without Division, a tutorial. This manual is available at http://homepage.cs.uiowa.edu/~jones/bcd/mod.shtml(Accessed On 1/8/2014).

[3] Open Source NTRU Public Key Cryptography and Reference Code. This is available at https://github.com/NTRUOpenSourceProject/ntru-crypto (Accessed on 30/5/2014).

[4] OpenPGP open source project RFC 4880. This RFC is available at http://www.ietf.org/rfc/rfc4880.txt (Accessed On 30/5/2014).

[5] OpenSSL open source project User Manual . This manual is available at http://www.openssl.org/docs/fips/UserGuide-2.0.pdf (Accessed On 30/5/2014).

[6] PuTTY open source project User Manual. This manual is available at http://the.earth.li/~sgtatham/putty/0.53b/htmldoc/ (Accessed On 30/5/2014).

[7] Vulnerability in OpenSSL FIPS Object Module 1.1.1 . This manual is available at https://www.openssl.org/news/secadv_20071129.txt (Accessed On 31/7/2014).

[8] Vulnerability Summary . This manual is available at http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-0166 (Accessed On 31/7/2014).

[9] Miklós Ajtai. The shortest vector problem in L_2 is NP -hard for randomized reductions (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, 1998. This can be accessed at http://doi.acm.org/10.1145/276698.276705(Accessed on 30/5/2014).

[10] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, 2001. This is available at http://doi.acm.org/10.1145/380752.380857(Accessed On 30/5/2014).

[11] American National Standards Institute. *ANSI X9.98-2010: Lattice-Based Polynomial Public Key Establishment Algorithm for the Financial Services Industry*, 2011. This is available at http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI+X9.98-2010(Accessed 0n 30/5/2014).

[12] Hendrik Willem Lenstra Arjen Klaas Lenstra and Laszlo Lovasz. Factoring polynomials with rational coefficients. 1982.

[13] Elaine Barker and John Kelsey. *Recommendation for random number generation using deterministic random bit generators (revised)*, 2007. This is available at http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf (Accessed on 13/8/2014).

[14] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption — how to encrypt with RSA. In *Advances in Cryptology – EUROCRYPT ' 94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, Berlin Germany, 1995.

[15] Byron Connolly. Quantum computers will be commercially available in 20 years: scientist, Dec 2012. This article is accessible at http://www.cio.com.au/article/444610 (Accessed on 29/05/2014).

[16] Quynh Dang. Changes in federal information processing standard (FIPS) 180-4, secure hash standard. *Cryptologia*, 2013. This is available at http://dx.doi.org/10.1080/01611194.2012.687431(Accessed On 30/5/2014).

[17] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[18] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology: Proceedings of CRYPTO 84*. Springer-Verlag, 1985, 1984.

[19] Craig Gentry. Key recovery and message attacks on NTRU-composite. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, Lecture Notes in Computer Science, 2001. This can be accessed at http://dx.doi.org/10.1007/3-540-44987-6_12(Accessed on 30/5/2014).

[20] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, 2009. This can be accessed at http://doi.acm.org/10.1145/1536414.1536440 (Accessed on 30/5/2014).

[21] Jens Hermans, Frederik Vercauteren, and Bart Preneel. Speed records for NTRU. In *Topics in Cryptology-CT-RSA 2010*, pages 73–88. Springer, 2010.

[22] Jeffrey Hoffstein, Jill Pipher, and Joseph Hillel Silverman. NTRU: A ring based public-key cryptosystem. In *Third International Algorithmic Number Theory Symposium (ANTS-III)*, Lecture Notes in Computer Science. Springer-Verlag, Berlin Germany, 1998.

[23] Jeffrey Hoffstein, Jill Pipher, and Joseph Hillel Silverman. NSS: An NTRU lattice-based signature scheme. *Lecture Notes in Computer*

Science, 2001. This is available at http://link.springer-ny.com/link/service/series/0558/tocs/t2045.htm(Accessed on 30/5/2014).

[24] Nick Howgrave-Graham, Joseph Hillel Silverman, Ari Singer, and William Whyte. NAEP: Provable security in the presence of decryption failures. *IACR Cryptology ePrint Archive*, 2003. This is available at http://eprint.iacr.org/2003/172(Accessed On 30/5/2014).

[25] Éliane Jaulmes and Antoine Joux. A chosen-ciphertext attack against NTRU. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2000.

[26] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. In Serge Vaudenay, editor, *Fast Software Encryption: 5th International Workshop*, volume 1372 of *Lecture Notes in Computer Science*, pages 168–188. Springer-Verlag, 1998.

[27] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *IACR Cryptology ePrint Archive*, 2012. This paper is available at http://eprint.iacr.org/2012/090 (Accessed on 30/5/2014).

[28] Ralph C Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[29] Daniele Micciancio. Closest vector problem. In *Encyclopedia of Cryptography and Security (2nd Ed.)*. 2011. This is available at http://dx.doi.org/10.1007/978-1-4419-5906-5(Accessed On 30/5/2014).

[30] Mariano Monteverde. NTRU software implementation for constrained devices, 2009. This is available at https://www.cosic.esat.kuleuven.be/publications/thesis-161.pdf (Accessed on 13/8/2014).

[31] National Institute of Standards and Technology (NIST). The Digital Signature Standard (DSS). Federal Information Processing Standards Publication (FIPS PUB) 186-4, 2013.

[32] Tatsuaki Okamoto and David Pointcheval. REACT: Rapid enhanced-security asymmetric cryptosystem transform. In *CT-RSA*, Lecture Notes in Computer Science, 2001. This is available at http://dx.doi.org/10.1007/3-540-45353-9_13(Accessed On 30/5/2014).

[33] John August Proos. Imperfect decryption and an attack on the NTRU encryption scheme. In *IACR Cryptology ePrint Archive*, 2003. This is available at http://citeseer.ist.psu.edu/557281.html(Accessed On 30/5/2014).

[34] Oded Regev. Lattice-based cryptography. In *Advances in Cryptology-CRYPTO 2006*, pages 131–141. Springer, 2006.

[35] Kurt Rohloff and David Bruce Cousins. A Scalable Implementation of Fully Homomorphic Encryption Built on NTRU. 2014. This is available at www.kaynar-rohloff.com/krohloff/papers/2014/Rohloff_Cousins_WAHC_2014_final.pdf; (Accessed On 30/7/2014).

[36] Adi Shamir Ronald Linn Rivest and Leonard Max Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Communications of the ACM*, volume 21, pages 120–126. ACM, 1978.

[37] RSA Laboratories. *PKCS #1: RSA Encryption Standard*, 1993.

[38] Peter Williston Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In *SIAM journal on computing*, volume 26, pages 1484–1509. SIAM, 1997.

[39] Joseph Hillel Silverman. NTRU cryptosystems technical report 014, version 1 title: Almost inverses and fast NTRU key creation, 1999. This can be accessed at https://www.securityinnovation.com/products/encryption-libraries/ntru-crypto/ntru-resources.html(Accessed on 30/5/2014).

[40] Ron Steinfeld. NTRU Cryptosystem: Recent Developments and Emerging Mathematical Problems in Finite Polynomial Rings. *Radon Series on Computational and Applied Mathematics*, 16. This is available at http://users.monash.edu.au/~rste/NTRU_survey.pdf; (Accessed On 30/7/2014).

[41] Ron Steinfield. NTRU Cryptosystem: Recent Developments. RICAM Special Semester on Applications of Algebra and Number Theory. These slides are available athttp://www.ricam.oeaw.ac.at/specsem/specsem2013/workshop4/ (Accessed On 30/5/2014).

[42] William Whyte, Nick Howgrave-Graham, Jeffrey Hoffstein, Jill Pipher, Joseph Hillel Silverman, and Philip Hirschhorn. IEEE P1363.1 draft 10: Draft standard for public key cryptographic techniques based on hard problems over lattices. *IACR Cryptology ePrint Archive*, 2008. This can be accessed at http://eprint.iacr.org/2008/361(Accessed on 30/5/2014).