

Implementation of Automatic Invertible Matrix Mechanism in NTRU Matrix Formulation Algorithm

Mohan Rao Mamdikar, Vinay Kumar & D. Ghosh

National Institute of Technology, Durgapur

E-mail : Mohanrao.mamdikar@gmail.com, vk841232@gmail.com, profdg@yahoo.com

Abstract – we present a paper of an enhancement of NTRU public key cryptosystem based on matrix formulation. In this work, an existing NTRU algorithm for matrix formulation Nayak et al.[2] seeking the modular inverse of NTRU algorithm for matrix is modified. Since the modular inverse plays important role in NTRU key generation. It requires to be check repeatedly taking considerable processing time for existence of modular inverse.

In this work we propose modification of existing NTRU algorithm which considerably reduces the processing time and makes the process much straightforward.

Here we investigate, square trinary matrix of a message is encrypted through matrix based NTRU to generate public/private key for encryption and decryption.

Index Terms— NTRU cryptosystem, matrix operations, modular inverse, public key, private key.

I. INTRODUCTION

NTRU is a public key cryptosystem presented by J.Hoffstein, J. Pipher and J. Silverman [1]. The first version of the NTRU encryption system was presented at the crypto ‘96 conference; [1]. The computational basis of the NTRU lies in polynomial algebra. Two different moduli are used for reduction of polynomials as fundamental tools which significantly speeds up its main competitors RAS and ECC. Polynomial algebra is the basic building block of the NTRU Encryption system. The truncated polynomials in the ring $R=Z[x]/(x^n-1)$ are basic objects and the reduction of polynomials with respect to relatively prime moduli i.e., p and q are the basic tool. Difficulty of finding a ‘Short’ factorization is the main strength of NTRU security system for such polynomial. This latter problem is equivalent to in certain 2N dimension lattice, finding a short vector. This widely studied as NP hard problem. NTRU polynomials $a(x)$ are frequently reduced modulo

p and q , the small and large moduli. The large modulus q is an integer, so reduction of $a(x) = a_0 + a_1x + a_2x^2 + \dots + a_{N-1}x^{N-1} \pmod q$ means just reduction of each a_i modulo q . The small modulus p can also be an integer. It is required that p and q are relatively prime i.e. $\gcd(p; q) = 1$. The main objects in the systems are ‘small’ polynomials; i.e. polynomials with small coefficients. The public key h is defined by an equation $f * h = p * g \pmod q$, where f and g are small polynomials.

The polynomial f should always have inverses modulo p and q , $f * fp \equiv 1 \pmod p$ and $f * fq \equiv 1 \pmod q$. Moreover, the parameters N , p and q are also public, and can be used as common domain parameters for all users. Polynomials f and g are private to the key owner. The polynomial g is needed only in key generation. Firstly Bob chooses two small polynomials f and g in the ring of truncated polynomials and keeps f and g private. He then computes inverse of $f \pmod p$ [fp] and inverse of $f \pmod q$ [fq], where p and q are relatively prime to each other. He then computes $h = p * fq * g \pmod q$, which becomes the public key for Alice the pair of polynomials f and fp forms his private key pair. The message is also represented in the form of a truncated polynomial. Let it be m . The sender Alice encrypts using the public key i.e. h as $e = h * r + m \pmod q$, where r is a random polynomial basically used to obscure the message. This encrypted message may be sent in a public channel. Alice decrypts the encrypted message using his private key pair by performing the following operations:

$$a = f * e \pmod q$$

$$b = a \pmod p$$

$$c = fp * b \pmod p \text{ and } c \text{ is the original message:}$$

$$c = fp * [f * (p * fq * g * r + m) \pmod q] \pmod p = m \text{ using the identities } f * fp = 1 \text{ and } f * fq = 1$$

Remark1. An existing algorithm about the inverse of polynomial or matrix is partially modulo p which is prime. But in NTRU suggested parameters [3], the modulus q is 2^r with $r > 1$ and q must large prime (at least 10^{th} prime from p).

To ensure the inverses of polynomial modulo p and q exist at the same time, we need to the improvement on the exiting algorithm given by Nayak et al. [2].

II. PROPOSED ALGORITHM

Bob creates a public/private key pair. He first randomly chooses two matrices f and g , where matrix f must have the following properties:

- 1) f is invertible mod p .
- 2) f is invertible mod q .
- 3) f is small.

In the previous section, we guaranteed that f was invertible mod p and q because, during the key generation process, we threw f away if the inverse didn't exist. In commercial applications, we use an alternative way of choosing f . We take

$$f = I + pF,$$

Where F is a small Matrix and I is identity matrix. This choice means that f is equal to $I \pmod{p}$, which has the following advantages:

- 1) f is always invertible mod p (in fact, $f^{-1} = I \pmod{p}$). This speeds up key generation, because we don't have to explicitly calculate the inverse mod p .
- 2) Because $f^{-1} = I \pmod{p}$, we no longer have to carry out the second matrix multiplication when decrypting. This speeds up decryption considerably, as it now only requires one multiplication, not two. It also means that we don't have to store $fp = f^{-1} \pmod{p}$ as part of the private key.

A. Key Generation

Bob creates a public/private key pair. He first randomly chooses two matrices F and g , consisting of $\{-1, 0, 1\}$. Where matrix F should be an invertible matrix (modulus p) that is determinant of F should be 1 and the number of 1's of matrix F should be one more than number of -1's. Similarly the number of 1's in g should be equal to the number of -1's. Bob keeps the matrices f and g private, since anyone who knows either one of them will be able to decrypt messages sent to Bob. Bob's next step is to compute the inverse of f modulo q and the inverse of f modulo p . Thus he computes matrix fq and fp which satisfies $f*fq = I \pmod{q}$ and $f*fp =$

$I \pmod{p}$. Where I is identity matrix. (Bob does not bother about the existence of inverse of matrix f by checking f is non-singular and f is invertible mod p because $f = I + pF$ is always invertible). Now Bob computes the product $H = p*fq*f*g \pmod{q}$. Bobs private key is the pair of matrices f and fp and his public key is the matrix H .

Key generation Example:

$F = 6 \times 6$ matrix, $g = 6 \times 6$ matrix, $p = 3$ and $q = 128$, $N=36$

$$F = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & -1 & -1 & -1 \\ 0 & 1 & 1 & -1 & -1 & -1 \\ 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 & 1 & -1 \\ -1 & 1 & 0 & 1 & -1 & 1 \end{bmatrix}$$

Then $f = I + pF =$

$$\begin{bmatrix} -2 & 0 & 0 & 3 & 0 & 3 \\ 3 & 1 & 3 & -3 & -3 & -3 \\ 0 & 3 & 4 & -3 & -3 & -3 \\ 0 & 3 & 0 & 4 & -3 & 0 \\ 0 & 0 & 3 & -3 & 4 & -3 \\ -3 & 3 & 0 & 3 & -3 & 4 \end{bmatrix}$$

And also generates

$$g = \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & -1 \\ -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & -1 & 0 & 1 \\ 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & -1 & 1 \end{bmatrix}$$

Next Bob find inverse of f (modulo q) [fq], which turns out to be

$$fq = \begin{bmatrix} 256 & 0 & 255 & 0 & 0 & 0 \\ 255 & 0 & 256 & 0 & 0 & 0 \\ 0 & 0 & 0 & 256 & 0 & 256 \\ 0 & 256 & 0 & 0 & 256 & 256 \\ 256 & 0 & 256 & 0 & 0 & 0 \\ 256 & 0 & 256 & 256 & 0 & 0 \end{bmatrix}$$

Now Bob generates a public key H as $H = p * f_q * g \pmod{q}$.

$$H = \begin{bmatrix} 765 & 3 & 0 & 765 & -768 & -768 \\ 768 & -3 & 0 & 768 & -765 & -765 \\ -768 & -768 & 1536 & -768 & -768 & 1536 \\ 0 & -768 & 2304 & -768 & -1536 & 768 \\ 768 & 0 & 0 & 768 & -768 & -768 \\ 0 & 0 & 768 & 0 & -768 & 0 \end{bmatrix}$$

Bob makes H publically available as his public key.

B. Encryption

Now Alice wants to send a message M to Bob using public key H. Alice put message in the form of binary matrix and its size is same as private key f and g. To create a encrypted message she chooses a Random matrix R of size f and g, this matrix is based on “blinding value”.

Alice computes encrypted message using R and Bob’s public key as follows. $E = R * H + M \pmod{q}$. The matrix E is the encrypted message which Alice sends to Bob.

Encryption Example:

Now, suppose Alice want to send the message M to bob by using bob’s public key.

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

She first choose a random matrix R

$$R = \begin{bmatrix} -1 & -1 & -1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & -1 & 0 \\ 1 & -1 & -1 & 1 & 1 & 0 \\ -1 & 1 & 1 & 1 & 0 & 1 \\ -1 & -1 & -1 & -1 & 1 & -1 \end{bmatrix}$$

respectively. Encrypted message $E = (R * H + M) \pmod{q}$ is computed as

$$E = \begin{bmatrix} 4 & 0 & 1 & 4 & 254 & 254 \\ 1 & 253 & 0 & 0 & 4 & 3 \\ 3 & 253 & 0 & 3 & 1 & 1 \\ 253 & 7 & 0 & 254 & 253 & 253 \\ 4 & 251 & 1 & 3 & 4 & 4 \\ 3 & 1 & 0 & 4 & 253 & 253 \end{bmatrix}$$

Alice sends this encrypted message E to Bob.

C. Decryption

Now Bob receives Alice’s encrypted message E and decrypt it. He begins by using his private matrix f to compute the matrix in following steps.

$A = f * E \pmod{q}$. Bob next computes the matrix $B = A \pmod{p}$.

Obtained B is the decrypted cipher-text which should be equal to M (Previously, Bob would also have had multiply B by the inverse of f mod p [fp], but we have chosen f so that its inverse mod p is equal to 1. This final step therefore become trivial multiplication by 1.) That is, he reduces each of the coefficients of A (modulo p).

Decryption Example:

Bob has received the encrypted message from Alice. He uses his private key f to compute $A = f * E \pmod{q}$

$$A = \begin{bmatrix} -8 & 24 & -2 & -2 & -14 & -14 \\ 10 & -21 & 0 & 6 & 7 & 6 \\ 3 & -30 & -3 & -3 & 22 & 19 \\ -21 & 34 & -3 & -17 & -12 & -15 \\ 25 & -53 & 4 & 15 & 37 & 37 \\ -18 & 31 & -6 & -11 & -15 & -18 \end{bmatrix}$$

Since A is computing A modulo q, he chooses the coefficients of A to lie between $-q/2$ and $q/2$. When A reduces the coefficients of $f * E \pmod{128}$, he chooses values lying between -63 and 64 and not between 0 and 127. Next B reduces the coefficients of A modulo p and choosing the interval in [0, 1], as follows $B = A \pmod{p}$:

$$B = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

The matrix will be Alice’s original message M, So Bob has successfully decrypted Alice message.

III. PROOF OF ALGORITHM

Alice’s encrypted message E looks like $E = R * H + M \pmod{q}$, but of course Bob doesn’t know the values of R and M. Bob’s first step is to compute $f * E$ and reduce the resultant modulo q. Remember that Bob’s public key H was actually formed by multiplying $p * f * q * g$ and reducing its coefficients modulo q. So although Bob doesn’t know R and M, when he computes $A = f * E \pmod{q}$, he is actually performing the following computation:

$$\begin{aligned} A &= f * E \pmod{q} \\ &= f * (R * H + M) \pmod{q} \\ &\quad [\text{Since } E = R * H + M \pmod{q}] \\ &= f * (R * p * f * q * g + M) \pmod{q} \\ &\quad [\text{Since } H = p * f * q * g \pmod{q}] \\ &= p * R * g + f * M \pmod{q} \\ &\quad [\text{Since } f * f * q = I \pmod{q} \text{ where } I \text{ is the identity matrix}] \end{aligned}$$

Next he computes

$$\begin{aligned} B &= A \pmod{p} \\ B &= (p * R * g + f * M) \pmod{p} \\ &= f * M \pmod{p} \\ &\quad [\text{Since } p * R * g \pmod{p} = 0] \\ &= (I + pF) * M \pmod{p} \\ &= M \pmod{p} \quad [\text{since } pF * M \pmod{p} = 0] \\ &= M \end{aligned}$$

IV. RESULTS AND OBSERVATION

In our paper we have proposed a method for choosing the different form of $f (= I + pF)$ for encryption and decryption using matrices in spite of taking f as proposed in the original A matrix formulation for NTRU cryptosystem . This method is more efficient compare to the Nayak et al.[2] as follows:

- 1) There is always truncated matrix in NTRU Cryptosystem is in invertible which speeds up the key generation.
- 2) During decryption we no longer have to carry out the second matrix multiplication, this speeds up decryption considerably.

In our paper we have used the following parameter for encryption and decryption procedure.

Table 1.parameter values for encryption and decryption

Degree of Polynomial (N)	q	p
36	512	3

We have successfully performed the encryption and decryption operations for message. We have compiled and run out code for the algorithm given by Nayak et al.[2] and our proposed algorithm on Matlab7.8.0(R2009a)and estimated time for different size of degree of polynomials for p=3,q=512 in table 2.

Table 2.Estimated execution time

Degree of Polynomial (N)	Existing Algorithm	Proposed Algorithm
16	4.0239	2.6239
25	4.1192	2.8678
36	4.6736	2.9737
49	4.7747	3.0167
64	4.9884	3.1610

And corresponding graph is shown in fig. 1.

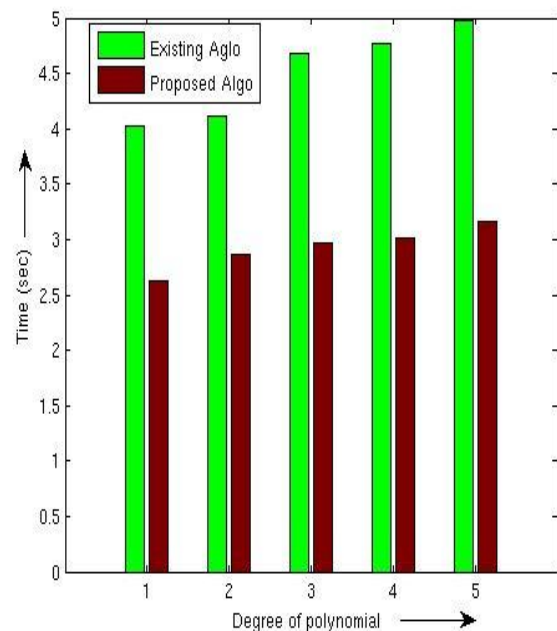


Fig. 1: Average estimated execution time

V. REFERENCES

- [1] J. Hoffstein, J. Pipher and J. H. Silverman, NTRU: A Ring-Based Public Key Cryptosystem. Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, J.P. Buhler (ed.), LNCS 1423, Springer- Verlag, Berlin, 267-288, 1998.
- [2] Rakesh Nayak, C.V. Sastry and Jayaram Pradhan “A Matrix Formulation for NTRU cryptosystem” Proc. 16th IEEE International conference on Networks(ICON-2008), New Delhi, India 12-14 December, 2008.
- [3] NTRU Cryptosystems Technical Report #014, Available from <http://www.ntru.com>.
- [4] Joffery Hoffstein, J. H. Silverman “Optimization for NTRU” Proceedings of conference on public key Cryptography and computational number theory, Warsaw, De Gruyter, 2000 (Sep 11-15),77-78.
- [5] C. Narasimham, Jayaram Pradhan Performance Analysis of Public key Cryptographic Systems RSA and NTRU, IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.8, August 2007,87- 96.
- [6] Rodney D’Souza The NTRU Cryptosystem: Implementation and Comparative Analysis, 2001.

