

NTRU ENCRYPTION USING HUFFMAN COMPRESSION

¹M.N.M. PRASAD , ²DR. MOHAMMED ALI HUSSAIN, ³DR. C.V. SASTRY

¹Research Scholar, Department of Computer Science and Engineering, KLEF University, Vaddeswaram, Guntur, Andhra Pradesh, India.

²Professor, Department of Electronics and Computer Engineering, KLEF University, Vaddeswaram, Guntur, Andhra Pradesh, India.

³Professor Department of Computer Science and Engineering, Regency Institute of Technology, Yanam, U.T. of Pondicherry, India.

E-mail: ¹prasadmushini@gmail.com, ²dralihussain@kluniversity.in, ³cvsastry40@yahoo.co.in

ABSTRACT

NTRU Labs have proposed a scheme for secure transmission using ring of truncated polynomials in $Z_q(x) / (x^n - 1)$. We have proved in this paper that a pre-processing the data to be transmitted greatly decreases the time of transmission. We have used a matrix version of NTRU Cryptosystems.

Keywords: *Truncated, Transmission, Polynomials, Ring, Secure, Number Theory Research Unit (NTRU)*

1. INTRODUCTION

Secure communication has been the subject of serious research since the discovery of RSA[1] encryption scheme using a public key for encryption and the corresponding private key for decryption. NTRU Labs [2, 3, 4] has come up with a scheme using a ring of truncated polynomials. In this scheme of things, two small polynomials f and g are taken from the ring of polynomials $Z_q(x) / (x^n - 1)$ which are invertible modulo q and modulo p , where q and p as two relatively prime numbers. The security of NTRU was shown to be equivalent to the hardness of some Lattice problems. Although there are many attacks [6, 7, 8, 9] no significant weakness in the NTRU encryption has ever been reported.

The public key is constructed as $h = p * fq * g \bmod q$ and f, fp are kept as private. Let m be the message to be transmitted, constructed as a polynomial belonging to $Z_q(x) / (x^n - 1)$. The encrypted text is obtained as $e = h * r + m$ where r is a random polynomial chosen to obscure the message. The message can be recovered by decrypting the encrypted message as follows using the private keys f and fp :

$$\begin{aligned} a &= f * e \bmod q \\ &= f * (h * r + m) \bmod q \\ &= f * (p * fq * g \bmod q + m) \bmod q ; \end{aligned}$$

a is the intermediate text generated and the original message is obtained as:

$$\begin{aligned} b &= a \bmod p \\ &= f * [(p * fq * g \bmod q + m) \bmod q] \bmod p = m \end{aligned}$$

This has been recently extended by Nayak[5] et al., where square matrices are used to generate public and private keys, and the message is also taken in the form of a matrix. The matrix formulation has the advantage that matrix inversions are easier to perform than polynomial inversions in modulo arithmetic.

In this paper we compress the message to be transmitted using Huffman code [11, 12, 13] and surprisingly find that the time taken to zip the message, encrypt the code, decrypt the code and unzip the message is less than the time taken for directly encrypting the message and decrypting the message, when the message size is comparatively larger. We use the matrix formulation of the NTRU cryptosystems.

2. PROPOSED ALGORITHM

We use $N \times N$ ordered set of matrices and generate multiple public/private key pairs [10]. The public keys are made available to everyone. Each of these public keys has a unique corresponding private key which will be known only to the receiver of the message. The sender sends the encrypted message along with public key and the receiver chooses the corresponding

private key. First the sender chooses the two $N \times N$ matrices F and G from $\{-1, 0, 1\}$ and F should be invertible. The inverse of F modulo p [F_p] and inverse of F modulo q [F_q] are then calculated. The matrices F_p and F_q satisfy $F * F_q = I \pmod{q}$ and $F * F_p = I \pmod{p}$. Then the public key is constructed as $P_u = p * F_q * G \pmod{q}$. The sender also generates the other set of public and private key pairs based on message length.

2.1 Key Generation

- Randomly generate $N \times N$ matrices F and G
- Compute F_p and F_q
- Generate Public key $P_u = p * F_q * G \pmod{q}$

2.2 Encryption

- The text message is then compressed using Huffman algorithm.
 $M_c = H(m)$ where M_c is compressed message
- Encrypt the message as $E_c = P_u * R + M_c \pmod{q}$ where R is any $N \times N$ random matrix.

The encrypted message can be decrypted through the following steps:

- Compute $A = F * E_c \pmod{q}$ and choose the coefficients in the range $-q/2$ to $q/2$
- Decrypt the message $D = F_p * A \pmod{p}$.
- Then the decrypted message D is uncompressed using Huffman code to generate the original message.

The sender encrypts the block of text, M_c as $E_c = P_u * R + M_c \pmod{q}$, where P_u is public key, R is a random matrix. The receiver first computes $F * E_c \pmod{q}$ using his private keys F and F_p corresponding to the public key P_u :

$$\begin{aligned} A &= F * E_c \pmod{q} \\ &= F * (P_u * R + M_c) \pmod{q} \\ &= F * (p * F_q * G * R + M_c) \pmod{q} \\ &= p * R * G + F * M_c \pmod{q} \quad [\text{since } F * F_q = I \pmod{q} \text{ and } I \text{ is identity matrix}] \end{aligned}$$

$$\begin{aligned} B &= F_p * A \pmod{p} \\ &= F_p * [p * R * G + F * M_c \pmod{q}] \pmod{p} \end{aligned}$$

$$\begin{aligned} &= M_c \pmod{p} \quad [\text{since } F_p * F \pmod{p} = I] \\ &= M_c \end{aligned}$$

3. ILLUSTRATION OF THE PROPOSED ALGORITHM

3.1. Key Generation

The sender generates the public key/private key pairs using the following parameters: $p=3$, $q=128$ and $N=15 \times 15$. Two randomly generated 15×15 matrices F and G consisting of elements from $\{-1, 0, 1\}$ are chosen as:

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

and $G =$

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

The sender computes the inverse of F (modulo p) [Fp] and F(modulo q)[Fq].

Fp =

```

0 1 2 0 2 2 1 1 1 2 0 1 1 1 0
2 2 1 2 1 1 0 0 2 0 2 2 1 0 2
1 1 2 0 2 2 1 0 2 0 0 2 1 0 1
2 0 0 1 1 2 1 2 0 2 1 1 2 0 2
2 2 0 0 1 1 0 2 1 1 0 2 2 1 1
0 0 2 2 1 2 1 2 1 1 2 0 2 2 1
0 0 0 1 1 2 1 1 2 0 2 2 1 0 2
1 1 2 1 1 2 2 0 0 1 0 2 0 2 2
1 2 2 2 2 0 1 1 2 2 2 2 2 1 2
0 1 0 0 0 1 2 2 2 1 1 2 2 1 1
0 2 0 2 0 0 2 1 0 1 2 1 1 2 1
0 2 0 2 0 1 1 0 0 2 0 2 0 2 2
2 2 2 2 0 2 0 1 0 0 1 0 0 2 0
0 1 1 1 1 2 1 0 2 0 1 2 1 1 2
2 2 1 0 0 1 0 2 0 1 0 2 1 1 2

```

Fq =

```

12 78 18 10 11 43 49 75 13 19 36 78 10 9
14 4 5 6 5
11 80 10 90 92 67 21 95 35 88 99 74 5 11 1
4 3 2 0
14 24 56 36 89 81 10 11 11 12 50 75 66 10 1
3 4 7 3 1
8 12 11 12 65 19 11 81 60 71 96 77 84 29 5
4 4 9 7
11 12 98 51 90 11 73 90 76 30 10 12 20 16 7
2 5 6 8 5 5
37 12 10 39 23 10 78 5 25 95 9 47 54 11 2
1 5 3 0 3
89 11 50 12 97 35 54 96 12 77 13 12 49 10 5
7 7 7 2 3 9
3 64 37 93 30 78 4 12 55 45 63 48 12 2 2
1 3
50 64 45 10 81 94 27 11 11 2 12 87 43 72 1
6 2 1 5
6 11 11 0 12 10 12 12 14 13 12 12 11 12 7
4 6 5 1 1 3 7 2 0
11 66 11 17 12 63 80 85 12 71 45 45 37 20 9
2 1 7 6
63 12 29 10 52 44 22 15 44 11 26 11 79 10 4
7 6 8 0 1 3
10 32 20 40 89 19 10 79 6 50 11 63 10 44 2
8 8 7 8 1
39 77 11 78 25 54 26 10 11 25 0 12 39 78 8
1 2 3 6
66 88 10 61 78 34 55 11 42 66 94 93 36 12 2
7 9 4 0

```

Then the public key $Pu = p * Fq * G$ (modulo q) and made available to everyone:

Pu =

```

73 10 36 4 11 3 55 12 10 73 4 15 16 12 57
5 5 5 3 3
12 8 38 85 2 80 63 5 10 13 10 80 12 33 3
5 4 3 1
64 64 11 10 71 11 47 37 10 32 12 12 10 96 10
5 1 3 8 2 2 8
93 10 10 6 51 10 31 90 4 15 32 61 0 11 97
0 3 5
88 14 69 11 30 53 85 89 96 10 10 10 73 11 76
8 6 3 8 2
78 7 58 62 21 84 75 85 80 22 35 27 16 68 10
4
83 56 92 45 81 12 7 75 63 12 67 41 47 73 83
2
95 76 56 27 5 39 55 3 97 10 67 40 9 11 8
6
93 1 73 10 84 61 10 69 10 91 91 24 93 11 8
8 1 6 6
53 12 53 12 66 12 24 53 11 14 74 52 44 29 6
3 9
82 84 24 10 14 86 12 11 62 33 57 91 10 5 75
3 6 0
10 8 10 73 20 53 12 23 80 93 48 85 77 42 15
8 3 1
11 38 98 45 16 57 12 31 12 59 12 77 51 63 93
0 1 2
11 12 27 29 12 10 59 7 82 63 32 12 67 12 22
5 0 1 8 4 0
63 96 9 13 74 99 12 42 44 91 77 11 10 28 19
5 6 8

```

3.2. Encryption

Let us choose the following text message consisting of 55 characters.

“ multiparty interactive network applications such as new “

which when express using ASCII code contains 440 bits. The text is then compressed using generated Hoffman Code for the text message and the length will be reduced to 223 bits:

```

0010010110000111110000111100000100111000
100010100010101110010010010000100011000
0111111010010101010100110001100010110011
1110101000011101110001110001100000001100
0010010101010111101011101100100001101101
00000111101010101000011000

```

The sender arranges this compressed message as 15x15 matrix

Mc =

```

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

```

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

Now choose the randomly generated R matrix as

0	0	1	0	0	-1	1	0	1	-1	1	1	0	1	-1
0	0	0	1	0	-1	-1	0	0	0	0	0	-1	1	-1
0	-1	-1	-1	-1	0	1	-1	0	0	0	0	1	0	0
1	1	1	1	-1	-1	0	-1	1	-1	-1	-1	0	0	-1
1	1	-1	1	0	1	1	0	-1	1	1	-1	0	0	0
-1	-1	1	0	-1	0	0	1	1	0	0	0	0	-1	1
1	0	1	0	0	0	-1	0	0	-1	0	-1	0	1	1
0	0	1	0	1	-1	0	0	0	0	1	-1	1	-1	-1
-1	0	0	0	0	-1	1	-1	0	0	0	0	1	-1	0
1	0	-1	0	-1	0	1	1	0	0	-1	1	-1	0	1
1	-1	0	0	0	-1	1	0	1	1	0	-1	-1	1	-1
1	0	0	-1	0	0	0	0	0	0	-1	0	0	-1	1
-1	-1	0	-1	1	1	-1	0	0	1	0	0	1	1	-1
-1	1	-1	1	0	-1	-1	-1	0	0	1	0	0	0	0
0	1	0	1	1	1	1	1	-1	-1	-1	0	0	1	-1

When we encrypt this message matrix M_c , we get the following cipher text E_c :

21	11	42	81	82	36	34	12	41	74	31	99	98	65	19
2							4							
9	38	17	20	41	47	35	93		5	79	11	8	16	26
										3				
94	35	48	10	15	56	74	89	11	12	34	84	12	17	83
			1					1	1			6		
10	35	42	11	91	55	10	11	86	11	43	27	9	5	11
2			8			8	9		3					
88	38	11	10	20	72	28	96	0	95	40	25	10	93	48
		7	0								4			
12	62	87	33	10	11	10	71	7	10	38	78	48	39	77
3				7	1	5								
41	64	11	30	63	5	25	72	44	10	21	59	33	24	11
		1						5						0
71	11	33	12	16	53	83	17	88	25	6	76	13	4	11
	4		7										6	
12	12	68	0	93	11	10	13	5	86	4	11	31	12	71
4	7				4	7				6		7		
22	42	10	87	29	38	21	93	62	10	6	11	42	10	12
		3						6		2		5	1	
57	58	67	66	45	65	19	0	11	27	43	70	0	55	11
								1						8

60	12	12	22	49	11	91	12	12	85	56	44	7	8	1
6	0				5		0	0						
20	18	28	30	45	11	12	11	12	48	10	12	83	22	50
					3	0		6		3	0			
52	59	11	67	12	89	96	63	13	12	12	59	97	94	91
		5		5				4	6					
89	98	12	12	85	95	20	11	31	39	97	80	67	59	13
		6					5							

This matrix is sent to the receiver along with the public key and the Huffman code is sent as a symbol table.

3.3. Decryption

$$A = F * E_c \text{ (modulo } p)$$

$$A =$$

12	12	10	12	10	12	12	0	3	1	11	11	4	12	11
5	5		6		7	6				2	9		6	6
1	5	12	12	1	3	12	11	12	2	10	12	13	12	14
		5	4			0	7	5			2		0	
7	12	8	12	6	12	12	3	6	1	6	1	12	13	12
4		2			2	3						7		4
11	12	12	12	0	12	9	8	12	12	12	4	11	13	12
	1	6	7		6			6	3	7		3		2
12	12	0	12	0	6	8	7	12	11	12	8	12	6	8
3	7		3					3	9	0		5		
12	2	8	1	12	11	11	12	3	12	12	5	12	11	17
2				4	9	9	5		5	3		6	0	
12	12	11	4	12	11	8	12	2	12	11	3	12	11	2
6	6	7		1	4		7		7	6		3	7	
11	12	8	3	12	11	12	12	5	12	3	1	5	12	12
1	3			6	7	3	6		1			1	1	
3	17	0	18	12	4	12	6	12	12	12	11	12	12	7
				2		3		4	1	6	8	6	7	
16	2	6	12	1	12	12	12	12	12	12	12	12	12	8
			6		2	4	6	7	4	4	4	2	6	
12	12	17	12	12	12	11	12	10	12	12	12	4	11	10
6	6		2	1	6	7	7		0	0	6		8	
0	4	5	4	10	12	3	12	2	1	1	12	3	11	11
				2		5				5			8	
0	12	12	6	3	12	4	9	3	2	3	6	11	14	11
	4	7			1							7		9
12	5	12	14	12	12	5	12	12	12	4	12	12	12	12
2		0		3	4		4	1	6		0	5	4	5
18	12	8	12	12	11	6	11	9	12	2	11	0	6	12
	4		1	2	9		7		2		7			2

The receiver chooses the elements of A between $-q/2$ and $q/2$ (-63 and 64).

-3	-3	10	-2	10	-1	-2	0	3	1	-16	-9	4	-2	-12
1	5	-3	-4	1	3	-8	-11	-3	2	10	-6	13	-8	14
7	-4	8	-6	6	-6	-5	3	6	1	6	1	-1	13	-4
11	-7	-2	-1	0	-2	9	8	-2	-5	-1	4	-15	13	-6
-5	-1	0	-5	0	6	8	7	-5	-9	-8	8	-3	6	8
-6	2	8	1	-4	-9	-9	-3	3	-3	-5	5	-2	-18	17
-2	-2	-11	4	-7	-14	8	-1	2	-1	-12	3	-5	-11	2
-17	-5	8	3	-2	-11	-5	-2	5	-7	3	1	5	-7	-7
3	17	0	18	-6	4	-5	6	-4	-7	-2	-10	-2	-1	7

16	2	6	-2	1	-6	-4	-2	-1	-4	-4	-4	-6	-2	8
-2	-2	17	-6	-7	-2	-11	-1	10	-8	-8	-2	4	-10	10
0	4	5	4	10	-6	3	-3	2	1	1	-3	3	11	-10
0	-4	-1	6	3	-7	4	9	3	2	3	6	-11	14	-9
-6	5	-8	14	-5	-4	5	-4	-7	-2	4	-8	-3	-4	-3
18	-4	8	-7	-6	-9	6	-11	9	-6	2	-11	0	6	-6

Now $M_c = F_p * A$ which is given below

The matrix A becomes

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

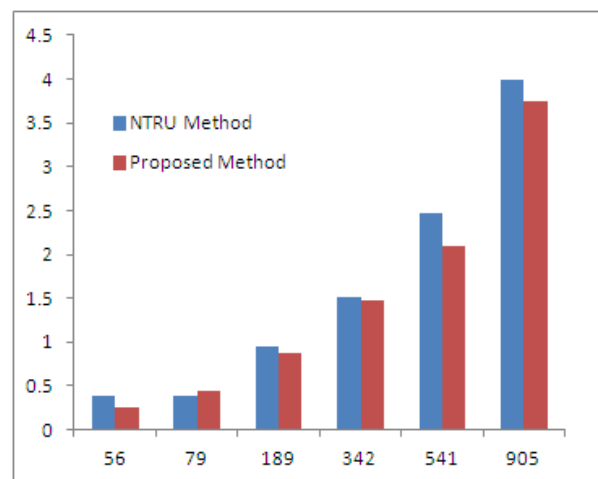
When we decompress this matrix then we get the original text message.

“ multiparty interactive network applications such as new “

4. EXPERIMENTAL RESULTS

The following table gives the time taken for a direct encryption and decryption and encryption and decryption after a Huffman compression:

Message Length	NTRU Method	Proposed Method
56	0.3	0.2105
79	0.3	0.37518
189	0.75	0.71089
342	1.2	1.19887
541	1.95	1.70452
905	3.15	3.0349



5. CONCLUSIONS

We used Huffman algorithm to compress the message and then used NTRU to encrypt the message and transmit it. It was shown that the compression of the data reduces the time for encryption and decryption. However we observe that when message contain 79 character, proposed scheme takes more time since there was no reduction in the number of blocks to be transmitted. We are presently working on further reduction in time by dynamically compressing the data. We used MATLAB Version 7.6.0(R2008a) for the above calculations.

ACKNOWLEDGEMENTS

We sincerely thank Mr. M S R S Prasad for fruitful discussions.

REFERENCES

- [1].R.L.Rivest, A.Shamir, and L.Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, vol.21, pp.120-126, February 1978
- [2].NTRU Cryptosystem, Technical Reports available at <http://www.ntru.com>
- [3].Jofferey Hoffstein, Jill Piper, Joshep H Silverman "NTRU A Ring based public key Cryptosystem" Lecture notes in Computer Science, Springer Verlag, Berlin 1433(1998),267-288.

- [4].A Course in Computational Algebraic Number Theory, H. Cohen, GTM 138, Springer Verlag, Berlin, 1993.
- [5].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.
- [6].D. Coppersmith, A. Shamir: Lattice attacks on NTRU, in Proc of EuroCrypt'97 (Lecture Notes in Computer Science),W. Fumy, Ed. Berlin, Germany: Springer, Vol. 1233 pp. 52-61, 1997.
- [7].Jintai Ding, Solving LWE problem with bounded errors in polynomial time, Cryptology ePrint Archive, Report 2010/558, 2010.
- [8].Jintai Ding, Fast Algorithm to solve a family of SIS problem with l1 norm, Cryptology ePrint Archive, Report 2010/581, 2010.
- [9].N. Howgrave-Graham, J.H. Silverman, W. Whyte: A Meet-In-The-Middle Attack on an NTRU Private Key. Technical Report #004, available at <http://www.ntru.com/cryptolab/technotes.shtml>
- [10]. Vinay Kumar, Mohan Rao Mamdikar, Dr. D. Gosh "Matrix Formulation of NTRU Algorithm using multiple Public keys from Matrix Data Bank for High Degree polynomials" Proc. of the Second Intl. Conf. on Advances in Computer, Electronics and Electrical Engineering -- CEEE 2013, ISBN: 978-981-07-6260-5 doi:10.3850/978-981-07-6260-5_40.
- [11]. Lecture-15: Huffman Coding (CLRS-16.3) available at www.cse.ust.hk/~dekai/271/notes/L15/L15.pdf
- [12]. Salomon D. Huffman Coding available at <http://www.springer.com/978-1-84800-071-1>, ISBN:978-1-84800-071-1,Softcover,2008
- [13]. Huffman Coding overview available at [http://csnotes.upm.edu.my/kelasmaya/pgkm20910.nsf/de8cc2082fc4d31b4825730e002bd111/e7f85333f149d3414825776f00207cbb/\\$FILE/Huffman_Coding1.pdf](http://csnotes.upm.edu.my/kelasmaya/pgkm20910.nsf/de8cc2082fc4d31b4825730e002bd111/e7f85333f149d3414825776f00207cbb/$FILE/Huffman_Coding1.pdf)