Bit Level Encryption Standard (BLES): Version-I

Neeraj Khanna
Department of Computer Science & Engineering
National Institute of Technology Calicut
Kerala, India

Dripto Chatterjee
Department of Informatics
Kings College
United Kingdom

Asoke Nath
Department of Computer Science
St. Xavier's College (Autonomous), Kolkata
Kolkata, India

Joyshree Nath
Jogesh Chandra Chaudhuri College
Kolkata, India

ABSTRACT

In the present paper the authors have introduced a new symmetric key cryptographic method called Bit Level Encryption Standard(BLES) which is based on bit exchanging or bit reshuffling method. The authors have introduced a completely new bit level encryption method. Nath et. al has already developed bit manipulation method called NJJSAA where the authors mainly used bit level right shift, bit level XOR operation. In the present paper the authors have used bit level exchange using random key generator and also byte level exchange using random key generator. The bit exchange was made using different block sizes such as 16 bits, 64 bits, 256 bits and 1024 bits long. To make the system hard the authors have changed the randomization matrix each time when data is extracted from plain text file and whenever the size of the block is changed. After finishing bit level exchange for the entire file the authors used the byte exchange method. The authors have also introduced a special bit manipulation method so the encryption algorithm will work even for all characters with ASCII Code 0 or all characters with ASCII Code 255. The standard encryption algorithm will fail to encrypt a file where all characters are ASCII '0' or all characters with ASCII '255' but the present method will be able to encrypt a file where all characters are ASCII '0' or all characters are ASCII '255'. The present method will be effective for encrypting short message, password, confidential key etc. The spectral analysis in the result sections shows that the present method is free from known plain text attack, differential attack or any type brute force attack.

General Terms

encryption, bit exchange, byte exchange

Keywords

BLES, NJJSAA, bit exchange, XOR, random key, differential attack

1. INTRODUCTION

With the tremendous development in internet technology in the last few years now it is a real challenge for the sender to send confidential data from one computer to another computer. There is no guarantee that between the sender and the receiver there is no one intercepting those confidential data especially if the data is not encrypted or properly protected. The security or the originality of data has now become a very important issue in data communication network. One cannot send any confidential or important message in raw form from one computer to another computer as any hacker can intercept that confidential message or important message. Sending question papers or sending bank statement is now a common practice over the mail. But this method is not fully secured as anybody can intercept the data from internet and misuse it. Nowadays it is not at all difficult for a hacker to intercept an e-mail and retrieve the confidential data especially if it is not encrypted. There are many sectors such as Banking, E-business, E-commerce, Railway or Air Reservation system where the data should not be tampered or intercepted by an unauthorized person. Any confidential data must be protected from any unwanted intruder to avoid any disaster. The disaster may happen if a sales manager of a company is sending some crucial data related to the sales of the company to his Managing Director over the e-mail and some intruder intercepts that data from the internet and passes it on to some other rival company. This type of disaster may happen when the data is moving from one computer to other computer in an unprotected manner. To overcome this problem one has to send the encrypted text or cipher text from client to server or to another client. To protect any kind of hacking problems nowadays network security and cryptography is an emerging research area where the programmers are trying to develop some strong encryption algorithm so that no intruder can intercept the encrypted message. This methods are called classical cryptographic algorithm and those methods can be divided into two categories: (i) symmetric key cryptography where one key is used for both encryption and decryption purpose. (ii) Public key cryptography where two different keys are used one for encryption and the other for decryption purpose. The merit of symmetric key cryptography is that the key management is very simple as one key is used for both encryption as well as for decryption purpose. In symmetric key algorithm the key is called secret key and it should be known to sender and receiver both and no one else. In public key cryptography there are two keys used one key is called public key which is used only for encryption purpose and the other is called private key which is used only for decryption purpose. The public key is not secret and it can be shared by anybody but the decryption key should be kept by the receiver only and by no one else. The public key methods have both merits as well as demerits. The problem of Public key cryptosystem is that one has to do massive computation for encrypting any plain text. Moreover in some public key cryptography the size of encrypted message may increase. Due to massive computation the public key crypto system may not be suitable in a case like sensor networks. So the security problem in sensor node is a real problem. In the present work we are proposing a symmetric key method called BLES method which can be applied in sensor network, mobile network, ATM network.

The present method uses bit exchange and byte exchange methods with complements and xor operation. The key element is the bit exchange depending on the randomized matrix which is generated every time and each one is unique. With different levels of extractions such as 2 bytes, 8bytes, 32 bytes and 128 bytes, the data finally gets shuffled to such an extent that without knowing the process and key, it would be impossible to decrypt. We have implemented the bit-wise exchange method as follows:

Firstly, we begin with initial transformation where the data is broken down to its corresponding bits and are then xored and complemented. These bits are stored in a reverse manner into a new file and this new file is now worked with.

Secondly, randomization number and encryption number is calculated.

Thirdly, first 2 bytes of data is extracted till the end of the file is read and is worked with, then 8 bytes, then 32 bytes and then 128 bytes. This process is executed till encryption number is reached.

The multiple encryptions make our system very secure.

2. ALGORITHM OF BLES VERSION-I

The present method is fully dependent on the text-key which is any string of maximum length 16 characters long. From the text-key we calculate two important parameters (i) Randomization number and (ii) Encryption number. To calculate this two parameters we use the method developed by Nath et al(1). We are giving below how we calculate the above two parameters and then the algorithm of BLES:

A. To calculate randomization number and encryption number:

The bit-manipulation may be applied multiple times to make the encryption process hard. The number of times the bit-manipulation will be repeated to be calculated from user entered text-key. The algorithm to calculate different parameters will be almost same as the method described in MSA algorithm by Nath et. al.

Suppose the user entered a key=”ABCD”
Therefore the length of the key=4
Step-1: Now calculate a sum from the given text key as follows:

Sum=Σ base^{postion} * ASCII value of character

Here for ASCII code of A=65, B=66, C=67, D=68

Base=5

Therefore s1=5^1 * 65 + 5^2 * 66 + 5^3 * 67 + 5^4 * 68=325+1650+8375+42500

=51365

(i) To calculate randomization number we proceed as follows:

Calculate again sum of the digits in s1 as s2=5+1+3+6+5=20

Find the modulo with s1 to obtain randomization number:

Nrand=s1 % s2=51365 % 20= 5

If nrand=0 or nrand>32 then set nrand=32.

(ii) To calculate encryption number we proceed as follows:

Calculate a sum as follows:

S3=5^1*68+5^2*67+5^3*66+5^4*65=340+1675+8250+40625=50890

Sum of digits in s3 is s4=5+0+8+9+0=22

Therefore encryption number nencrypt=s3 % s4=50890 % 22 =4
If nencrypt=0 or >32 then set nencrypt=32.

Now we begin with the Algorithm of BLES Version-I

B. Algorithm for Encryption method : Bit Exchange Algorithm :

The algorithm will be divided into several modules where each module gives the pseudo code of the function that is implemented.

Module 1
Main Block:

1. Enter file to be encoded i.e. file4
2. Enter file to be saved to i.e. file1
3. Run ini_trans()
4. Run keygen()
5. for t = 1 to times
    a. do
    b. fp1 -> fopen(file1)
    c. l -> ftell(fp1)
    d. for i1 = 1
        i. do
        ii. if (l-sh) >=0 && i1 <=4 then
            1. sh -> sh<<2
            2. i1 -> i1+1
        iii. else
            1. i1 -> i1-1
            2. break
    e. sh -> 2
    f. sh1 -> 2
    g. fclose(fp1)
    h. for m = 1 to i1
        i. do
        ii. open file1 in read mode
        iii. open file2 in write mode
        iv. sh1 -> sh1<<1
        v. siz -> sh1
        vi. l -> ftell(fp1)
        vii. n1 -> l/sh
        viii. n2 -> l%sh;
        ix. n1 -> 0
        x. for i = 0 to sh1-1
            1. do
            2. for j = 0 to sh1-1
                a. do
                b. mat[i][j] -> nl++
                c. j -> j+1
            3. i -> i+1
        xi. for q = 1 to n1
            1. do
            2. nl -> 0
            3. for i = 0 to sh1-1
                a. do
        i. for j = 0 to sh1-1
        ii. do
        iii. mat[i][j] -> nl++
        iv. j -> j+1
                b. i -> i+1
            4. if secure>500 then
                a. secure -> secure -488
                b. tem -> 2
            5. for i = 1 to secure
                a. do
    b. randomization()

c. i -> i+1
6. if sh=2 then
a. read 2 bytes data from file1
b. bit_stream()
c. encrypt_bit()
7. if sh=8 then
a. read 8 bytes data from file1
b. bit_stream()
c. encrypt_bit()
8. if sh=32 then
a. read 32 bytes data from file1
b. bit_stream()
c. encrypt_bit()
9. else
a. read 128 bytes from file1
b. bit_stream()
c. encrypt_bit()
10. secure -> secure + tem
11. tem -> tem +2
12. q -> q+1

xii. if n2 ! 0 then
1. for i = 0 to n2-1
2. do
3. extract first residual byte
4. bit_stream1()
5. for j = 0 to 7
a. do
b. write bit to file file2
c. j -> j+1
6. I -> i+1

xiii. sh -> sh<<2
xiv. close all files
xv. open file2 in read mode(file2 is the bit file)
xvi. open file1 in write mode
xvii. n -> ftell(fp1)
xviii. /* Now we read the file in reverse direction
xix. while n>0
1. do
2. n -> n-8
3. read 8 bytes from file2
4. convert it to corresponding character
5. write it to file1

xx. close all files
xxi. byte_exchange()
xxii. m -> m+1

i. t -> t+1

6. end

Module 2
bit_stream()
The above function is used to break down character to its corresponding bits and store in data[]. The use of this function is that it coverts 2 bytes, 8 bytes, 32 bytes and 128 bytes at once.

Module 3
bit_stream1()
This function breaks the residual characters to corresponding bits. The use of this function is to convert each character at a time.

Module 4
encrypt_bit()
1. Complement every position in the bit array (data[]) which is divisible by the corresponding bytes being extracted.
2. s1 -> 0
3. for i = 0 to sh1-1
a. do
b. for j = 0 to sh1-1
i. do
ii. x -> mat[i][j]
iii. temp -> data[s1]
iv. data[s1] -> data[x]
v. data[x] -> temp;
vi. s1 -> s1+1
vii. x -> 0
viii. j -> j+1
c. i -> i+1
4. store elements in the bit array (data[]) into file2.

Module 5
byte_exchange()
1. open file1 in read mode
2. open file2 in write mode
3. l -> ftell(fp1)
4. n1 -> l/16
5. n2 -> l%16
6. nl -> 0
7. for j = 0 to 3
a. do
b. for k = 0 to 3
i. mat[j][k] -> nl++
ii. k -> k+1
c. j -> j+1
8. if secure > 500 then
a. secure -> secure – 488
b. tem -> 2
9. for i = 1 to secure
a. do
b. randomization()
c. i -> i+1
10. for i = 1 to n1
a. do
b. read 16 bytes data from file1
c. encrypt_byte()
d. i -> i+1
11. if n2 <> 0 then
a. for i = 1 to n2
i. do
ii. read one character from file1
iii. write it to file2
iv. i -> i+1
12. close all files
13. copy file2 to file1

Module 6
encrypt_byte()
This function is similar to encrypt_bit()

Module 7
ini_trans()
Step-1: Open plain_text_file in input mode

Step-2: Open output1 in output mode
Step-3: Read one byte from plain_text_file
Step-4: Convert the byte into 8 bits
Step-5: Reverse all bits
Step-6: Take 1-s complement of all 8-bits
Step-7: Perform XOR operations among the bits as follows:
Step-7.1: Compute Bit-1 xor Bit-8 and substitute the new bit in bit-8 position.
Step-7.2: Compute Bit-2 xor Bit-7 and substitute the new bit in bit-7 position.
Step-7.3: Compute Bit-3 xor Bit-6 and substitute the new bit in bit-6 position.
Step-7.4: Compute Bit-4 xor Bit-5 and substitute the new bit in bit-8 position.

Table-1(i) Plain Text=ASCII 1=00000001_2

Original	0	0	0	0	0	0	0	1
Reverse	1	0	0	0	0	0	0	0
1-s Comp	0	1	1	1	1	1	1	1
Xor=Final pattern	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**1**

Table-1(ii) Plain Text=ASCII 2=00000010_2

Original	0	0	0	0	0	0	1	0
Reverse	0	1	0	0	0	0	0	0
1-s Comp	1	0	1	1	1	1	1	1
Xor=Final pattern	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**0**

Table-1(iii) Plain Text=ASCII 3=00000011_2

Original	0	0	0	0	0	0	1	1
Reverse	1	1	0	0	0	0	0	0
1-s Comp	0	0	1	1	1	1	1	1
Xor=Final pattern	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**

Step-8 : Convert 8 bits to corresponding byte and write to output1 file.
Step-9: Read next byte from the same input file and repeat step-4 to step-8.
Step-10 : Repeat Step-9 till the end of file.
Step-11 : Close output1 file

The entire process above can be repeated n number of times depending on the value generated by the keygen function.

Decryption Algorithm
The functions used in Decryption Module are :
1.keygen();
2. rev();
3. char_convert();
4. char_convert2();
5. bit_wise_xor(int []);
6. final_trans();
7. randomization();
8. leftshift();
9. cycling();
10. upshift();
11. rightshift();
12. downshift();
13. decrypt_bit();
14. decrypt_byte();
15. byte_exchange();
16. bit_stream(char s[]);
17. bit_stream1(char s,int i1);
18. bit_stream2(char s);

The above functions have been described in out encryption module. The only difference is that decryption methods will work in reverse order of the encryption process. It means the last operation in encryption process should be the first operation in decryption process.
Functions 7 to 12 are described in MSA algorithm.

RANDOMIXZATION OF MATRIX USING MEHEBOOB, SAIMA & ASOKE(MSA) RANDOMIZATION METHOD

We first create a square matrix of size n x n where n can be 4, 8, 16 and 32. First we store numbers 0 to (n*n-1). We apply the following randomization techniques to create a random key matrix. The detail description of randomization methods is given by Nath et.al[1].

The following Randomization methods were applied on initial key matrix to obtain a randomized key matrix:

Step-1: call Function cycling()
Step-2: call Function upshift()
Step-3: call Function downshift()
Step-4:call Function leftshift()
Step-5:call Function rightshift()

3. RESULTS AND DISCUSSION

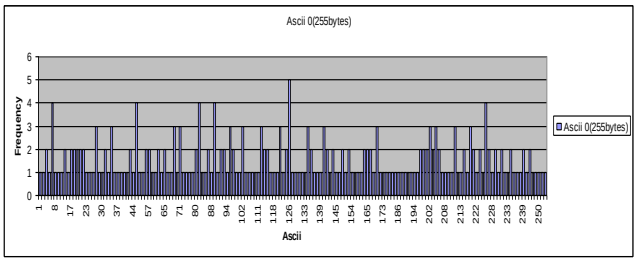

Fig-1: Frequency Spectral analyses of Plain Text file containing 255 ASCII '0'.

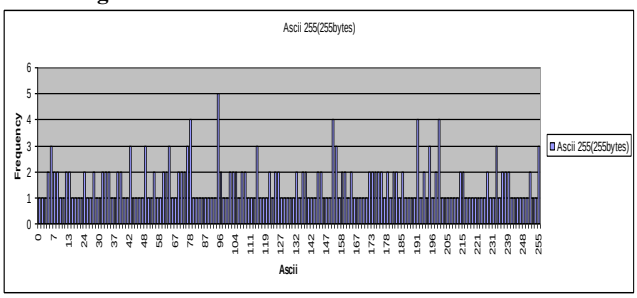

Fig-2: Frequency Spectral analyses of Plain Text file containing 255 ASCII '255

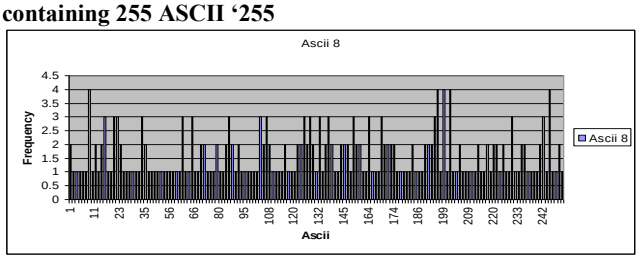

Fig-4: Frequency Spectral analyses of Plain Text file containing 255 ASCII '8'.

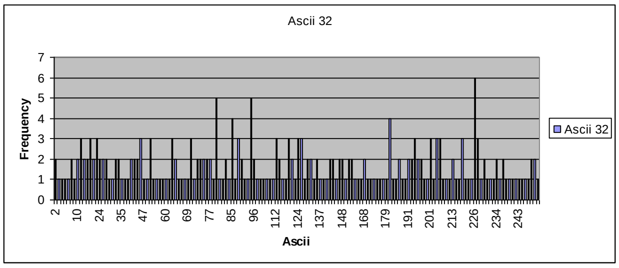

Fig-6: Frequency Spectral analyses of Plain Text file containing 255 ASCII '32'

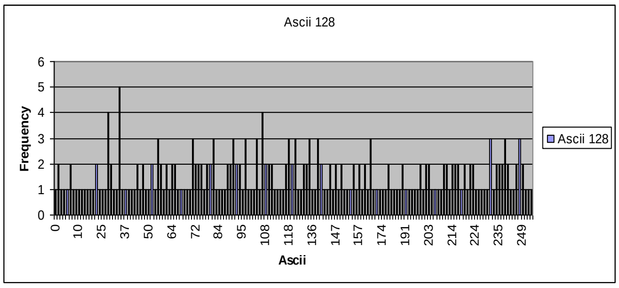

Fig-8: Frequency Spectral analyses of Plain Text file containing 255 ASCII '128'.

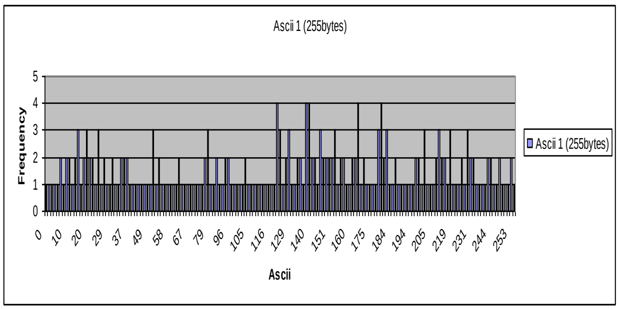

Fig-9: Frequency Spectral analyses of Plain Text file containing 255 ASCII '1'.

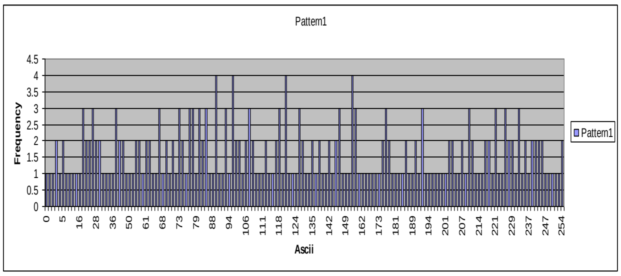

Fig-10: Frequency Spectral analyses of Plain Text file containing 254 ASCII '1' and 1 ASCII '2'.

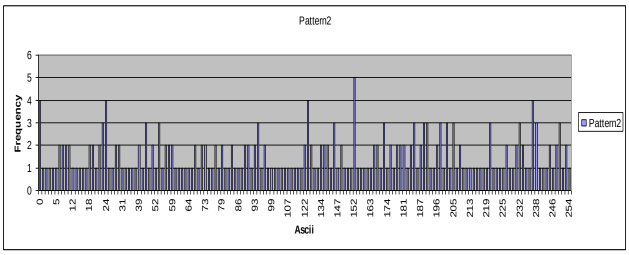

Fig-11: Frequency Spectral analyses of Plain Text file containing 1 ASCII '2' and 254 ASCII '1'.

Table 1: A Plain Text file contain a paragraph and its encrypted file.

<table>
<tr><td>St.Xavier's College is owned and managed by the Jesuits
of the Calcutta Province of the Society of Jesus. With the
registrarion of Catholic Mission of West Bengal (also known
as Calcutta Province of the Society of Jesus) under the Societies
Registration Act 1961, the ownership of St. Xavier's College was
vested with this said society from its registration in 1972, and it
was administered by a Governing Body constituted as per statutes of Calcutta University.</td><td>┤;qⁿcs}╬╚º┤Uk┌╞#→RR┤║å◘RÇ╔☻≈
Ö•Θ╬÷X╧▀±≤→╕á>]ny╔┬╧uß▬\Ü
╒║├y▄U╘╘╚è]═nΩì┘@╤7╒┤║Nó▄■
↕
Ü±₧)L█αBM\╚≤=U@²┬½É╡╕Mëy^
φd♪ò╜'♦ä♫╒°í╗~↑c·R╒ò
♀¬╓↓¢H≤v╛rWTJ%·C▼╕º↔fr_á
►♠≥ê╖;╥→Æ♫Du▬i☻ⁿ|£╧α☼§c→
δ
↕╕à▼à█\N⌡ilin8ô╙,τ@`s╥ªOoε
ε σ!~Σ∩ïW╙ÿ◄|╝↔Ω╘╚◘
┼ºƒ⌂ê#KH(☼═↑O Θ;┬☻
ÿ°m┘ò÷{b┬üFI⌠☺ß└¿h
lO@■π═└Jî═qZ9╘τ\ÿ=£8%☺╓Ω╡
‼^á∩g¼→èoQ┘╘☺S|ⁿZ├HC┤Ñτt
áαº_L←VX£▼ó≥o▲░Üa╥▀ΣIV9♣╡
ßδ▓ƒφc;ÿù<Φ2÷¶b#u*÷%╖IDT♪aôZ╞
Dælfú/lX[ⁿÑ■═ªë·Φ<f¥╟•ε±↨ε╦@⌡╙
Ü;XX↓←ë┬kú╤m☻IΩfh⌐üΓ$ε-
▓q╟¿]2sσ?╔gx╧
(xòAÜ¶¿C→_ªΩM X┘≥╛q+A╟W
∟~!∟→t*Ω÷↑═ò╬╡H♣i╜mN</td></tr>
</table>

4. CONCLUSION AND FUTURE SCOPE

The maximum randomization number we have used is 500 and encryption number 32. The encrypted text cannot be decrypted without knowing the exact initial random matrix. The size of random matrices is 4x4, 8x8, 16x16, 32x32. The numbers may be arranged in 16! Ways in 4x4 matrix. The numbers in 8x8 matrix may be arranged in 64! Ways. The numbers in 16x16 may be arranged in 256! Ways. The numbers in 32x32 may be arranged in 1024! ways. To complete the whole process we choose any of the random matrix to perform bit exchange method and there is no similarity between any two matrices and even if there is then it is very hard to find out the similar ones. The spectral analysis shows that our present method is free from standard cryptography attacks namely brute force attack, known plain text attack and differential attack. The present method will be most effective to encrypt short message such as SMS in mobile phone, password encryption and any type of confidential message. If the file size is large then the present method will take more time to encrypt. So therefore BLES may be used in defence systems, Banking systems, Sensor networks, Mobile computing etc. BLES may be further upgraded by introducing generalized modified vernam cipher method with feedback introduced by Nath et al[7,8].

5. ACKNOWLEDGMENTS

We are very much grateful to the Department of Computer Science to give us this opportunity to work on symmetric key Cryptography. One of the authors (AN) sincerely expresses his gratitude to Fr. Dr. Felix Raj, Principal of St. Xavier's College(Autonomous) for giving constant encouragement in doing research in cryptography. NK is grateful to Mr. Biswanath Chakraborty for his support and inspiration.

6. REFERENCES

[1] Symmetric Key Cryptography using Random Key generator : Asoke Nath, Saima Ghosh, Meheboob Alam Mallik: "Proceedings of International conference on

security and management(SAM'10" held at Las Vegas, USA Jull 12-15, 2010), Vol-2, Page: 239-244(2010).

[2] Advanced Symmetric key Cryptography using extended MSA method: DJSSA symmetric key algorithm: Dripto Chatterjee, Joyshree Nath, Soumitra Mondal, Suvadeep Dasgupta and Asoke Nath, Jounal of Computing, Vol 3, issue-2, Page 66-71,Feb(2011).

[3] A new Symmetric key Cryptography Algorithm using extended MSA method :DJSA symmetric key algorithm, Dripto Chatterjee, Joyshree Nath, Suvadeep Dasgupta and Asoke Nath : Proceedings of IEEE International Conference on Communication Systems and Network Technologies, held at SMVDU(Jammu) 03-06 June,2011, Page-89-94(2011).

[4] New Symmetric key Cryptographic algorithm using combined bit manipulation and MSA encryption algorithm: NJJSAA symmetric key algorithm :Neeraj Khanna, Joel James,Joyshree Nath, Sayantan Chakraborty, Amlan Chakrabarti and Asoke Nath : Proceedings of IEEE CSNT-2011 held at SMVDU(Jammu) 03-06 June 2011, Page 125-130(2011).

[5] Symmetric key Cryptography using modified DJSSA symmetric key algorithm, Dripto Chatterjee, Joyshree Nath, Sankar Das, Shalabh Agarwal and Asoke Nath, Proceedings of International conference Worldcomp 2011 held at LasVegas 18-21 July 2011, Page-306-311, Vol-1(2011).

[6] An Integrated symmetric key cryptography algorithm using generalized vernam cipher method and DJSA method: DJMNA symmetric key algorithm : Debanjan Das, Joyshree Nath, Megholova Mukherjee, Neha Chaudhury and Asoke Nath: Proceedings of IEEE International conference : World Congress WICT-2011 to be held at Mumbai University 11-14 Dec, 2011, Page No.1203-1208(2011).

[7] Symmetric key cryptosystem using combined cryptographic algorithms- generalized modified vernam cipher method, MSA method and NJJSAA method: TTJSA algorithm – Trisha Chatterjee, Tamodeep Das, Joyshree Nath, Shayan Dey and Asoke Nath, Proceedings of IEEE International conference : World Congress WICT-2011 t held at Mumbai University 11-14 Dec, 2011, Page No. 1179-1184(2011).

[8] Symmetric key Cryptography using two-way updated – Generalized Vernam Cipher method: TTSJA algorithm, International Journal of Computer Applications(IJCA, USA), Vol 42, No.1, March, Pg: 34 -39(2012).

[9] Ultra Encryption Standard(UES) Version-I: Symmetric Key Cryptosystem using generalized modified Vernam Cipher method, Permutation method and Columnar Transposition method, Satyaki Roy, Navajit Maitra, Joyshree Nath,Shalabh Agarwal and Asoke Nath, Proceedings of IEEE sponsored National Conference on Recent Advances in Communication, Control and Computing Technology-RACCCT 2012, 29-30 March held at Surat, Page 81-88(2012).

[10] An Integrated Symmetric Key Cryptographic Method – Amalgamation of TTJSA Algorithm, Adbvanced Caeser Cipher Algorithm, Bit Rotation and reversal Method : SJA Algorithm., International Journal of Modern Education and Computer Science, Somdip Dey, Joyshree Nath, Asoke Nath,(IJMECS), ISSN: 2075-0161 (Print), ISSN: 2075-017X (Online), Vol-4, No-5, Page 1-9,2012.

[11] An Advanced Combined Symmetric Key Cryptographic Method using Bit manipulation, Bit Reversal, Modified Caeser Cipher(SD-REE), DJSA method, TTJSA method: SJA-I Algorithm, Somdip dey, Joyshree Nath, Asoke Nath, International Journal of Computer Applications(IJCA 0975-8887, USA), Vol. 46, No.20, Page- 46-53,May, 2012.

[12] Cryptography and Network Security, William Stallings, Prectice Hall of India.