ISSN: 2321-7782 (Online)
Volume 2, Issue 10, October 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

3d Multi Way Feedback Encryption Standard Version I (3dMWFES-1)

Arijit Ghosh[1]
Department of Computer Science
St. Xavier's College
Kolkata – India

Prabhakar Chakraborty[2]
Department of Computer Science
St. Xavier's College
Kolkata – India

Dr. Asoke Nath[3]
Department of Computer Science
St. Xavier's College
Kolkata – India

Shamindra Parui[4]
Department of Computer Science
St. Xavier's College
Kolkata – India

Abstract: ***In the present paper the authors have introduced a new symmetric key cryptographic method which is based on two way simultaneous feedback method applied along two dimensions on plain text. The process starts by dividing the plain text into a number of square matrices where the size of the matrix and no of layers are balanced. For example if plain text has 20 characters then the size of the matrix will (3x3) and no of layers will be 2 and 2 additional residual characters. The initial feedback is predetermined. The forward and the backward feedback is applied to each layer (i.e. each square matrix) along two dimensions (i.e. row wise and column wise). The ASCII value of plain text, the key value, the forward feedback value and the backward feedback value calculated along two dimensions are used to generate the cipher text. The intermediate cipher text is taken modulo 256 to get the final cipher text. After completion of applying feedback mechanism row wise column wise feedback is then applied. The decryption process is obtained by observing the pattern of cipher text and by evaluating the values of feedback arrays. The present feedback method is totally a new method and this can be used for encryption of message, password, and confidential information's in sensor networks. The results of this method show that the present method is free from common attacks such as known plain text attack, differential attack or any kind of brute force attack.***

Keywords: multiway feedback, forward feedback, backward feedback.

I. INTRODUCTION

Internet is now almost open to all. Therefore it is very difficult to send any confidential message from one computer to another without any encryption. There is a very common attack where an intruder can listen to both the sender and receiver and then if he/she wants can manipulate the actual message. The private data must be encrypted first and then it should be sent over internet. The hackers are always ready to intercept the message of others. So it is always recommended that if the message is very confidential then that must be encrypted first before sending it through e-mail or over internet. The hackers have already developed many sites where they keep soft wares that can be used to crack password of email. The cryptographers try to develop some good encryption algorithm and the hackers try to make some brute force method to crack the encryption method without knowing the actual decryption method. The security or the originality of data has now become a very important issue in data communication network. It must be ensured that in any kind of e-business, air or railway reservation system or in credit card or debit card system the data should not be tampered or intercepted by an unauthorized person. In corporate sector it would be a disaster if the data is sent from one computer to another in an unprotected manner. To get rid of this problem one has to send any important message in encrypted form rather than plain raw text form. To protect data from intruder now network security and cryptography is a very important research area where the programmers are trying to develop some strong

encryption algorithm so that no intruder can intercept the encrypted message. The present encryption method can be applied multiple times to make the system fully secured. Through tests were made on some standard plain text files and it was found that it is quite impossible for any intruder to extract the plain text from encrypted text using brute force method.

II. Method used to Generate Feedback

The forward and backward feedbacks are applied to each row simultaneously and the resultant value is stored in the corresponding feedback array. The forward feedback value is measured by ifxf = ((strarr[k][i][j] + keyarr[k][i][j] + ifxfarr[k][i][j] + ibxfarr[k][i][j])%256); and the backward feedback value is measured by ibxf=((strarr[k][i][c1-j-1] + keyarr[k][i][c1-j-1]+ ifxfarr[k][i][c1-j-1]+ibxfarr[k][i][c1-j-1])%256); the measured value are then stored to the corresponding cell of the feedback arrays. Before encryption the arrays will look like,

TABLE I

String Array(strarr)		
1	1	1
1	1	1
1	1	1

TABLE III

Key Array(keyarr)		
1	1	1
1	1	1
1	1	1

TABLE IIIII

Forward X-Axis Array(ifxfarr)		
0	0	0
0	0	0
0	0	0

TABLE IVV

Backward X-Axis Array(ibxfarr)		
0	0	0
0	0	0
0	0	0

TABLE V

Forward Y-Axis Array(ifyfarr)		
0	0	0
0	0	0
0	0	0

TABLE VI

Backward Y-Axis Array(ibyfarr)		
0	0	0
0	0	0
0	0	0

The forward feedback and backward feedback arrays will be modified after each step. The array keyarr will remain same. The strarr will be modified after the completion of feedback method along row wise. In the following diagrams we present different intermediate values of the feedback arrays. After completion of first iteration the feedback matrix will look like,

TABLE VII

Forward X-Axis Array(ifxfarr)		
0	2	0
0	0	0
0	0	0

TABLE VIII

Backward X-Axis Array(ibxfarr)		
0	2	0
0	0	0
0	0	0

The forward and backward feedback will applied to each row. And after completion or each row the feedback will be passed to the next row. After completion of feedback in first row the matrices will be look like,

TABLE IX

Forward X-Axis Array(ifxfarr)		
0	2	6
8	0	0
0	0	0

TABLE X

Backward X-Axis Array(ibxfarr)		
6	2	0
0	0	8
0	0	0

After completion of feedback method along row wise the ifxf and the ibxf array will look like,

TABLE XI

Forward X-Axis Array(ifxfarr)		
104	2	6
8	10	22
32	34	70

TABLE XII

Backward X-Axis Array(ibxfarr)		
6	2	104
22	10	8
70	34	32

The strarr array will be modified as:

strarr[k][i][j]=(strarr[k][i][j]+keyarr[k][i][j] + ifxfarr[k][i][j] + ibxfarr[k][i][j]). The matrix will look like after modification:

TABLE XIII

String Array (strfarr)		
6	2	104
22	10	8
70	34	32

Now the column wise feedback will be applied to the modified strarr. Before applying column wise feedback the arrays will be,

TABLE XIV

String Array (strfarr)		
6	2	104
22	10	8
70	34	32

TABLE XV

Key Array (keyarr)		
1	1	1
1	1	1
1	1	1

TABLE XVI

Forward X-Axis Array (ifxfarr)		
104	2	6
8	10	22
32	34	70

TABLE XVII

Backward X-Axis Array (ibxfarr)		
6	2	104
22	10	8
70	34	32

TABLE XVIII

Forward Y-Axis Array (ifyfarr)		
0	0	0
0	0	0
0	0	0

TABLE XIX

Backward Y-Axis Array (ibyfarr)		
0	0	0
0	0	0
0	0	0

The forward and backward feedback will be applied on the cipher text same as previous method but along column wise. After completion of first row the feedback will be applied to the next row. And the forward feedback of the last cell of a column will be transferred to the first cell of next column. The final forward and backward feedback array will look like,

TABLE XX

Forward Y-Axis Array (ifyfarr)		
140	100	232
113	107	89
251	53	131

TABLE XXI

Backward Y-Axis Array (ibyfarr)		
251	53	131
105	179	9
220	108	160

After completion of feedback mechanism along column wise the string array will be modified as

Strarr[k][i][j]=(strarr[k][i][j]+keyarr[k][i][j]+ifyfarr[k][i][j] +ibyfarr[k][i][j]).

The values stored in the string array is the cipher text. According to our algorithm if the plain text is not divided into perfect square matrices then for rest of the characters remained will be put into a new matrix which is of the same size and layer as of the strarr. The remaining cells of the new matrix will be filled up using values from cipher text array and will be re-encrypted. As in our example we have no extra characters that are why the cipher text will be taken into the new array and will be re-encrypted. The final cipher text will look like.

TABLE XXII

Final Cipher Array (cipharr)		
113	179	213
31	123	159
149	67	153

III. Block Diagram of Encryption Method

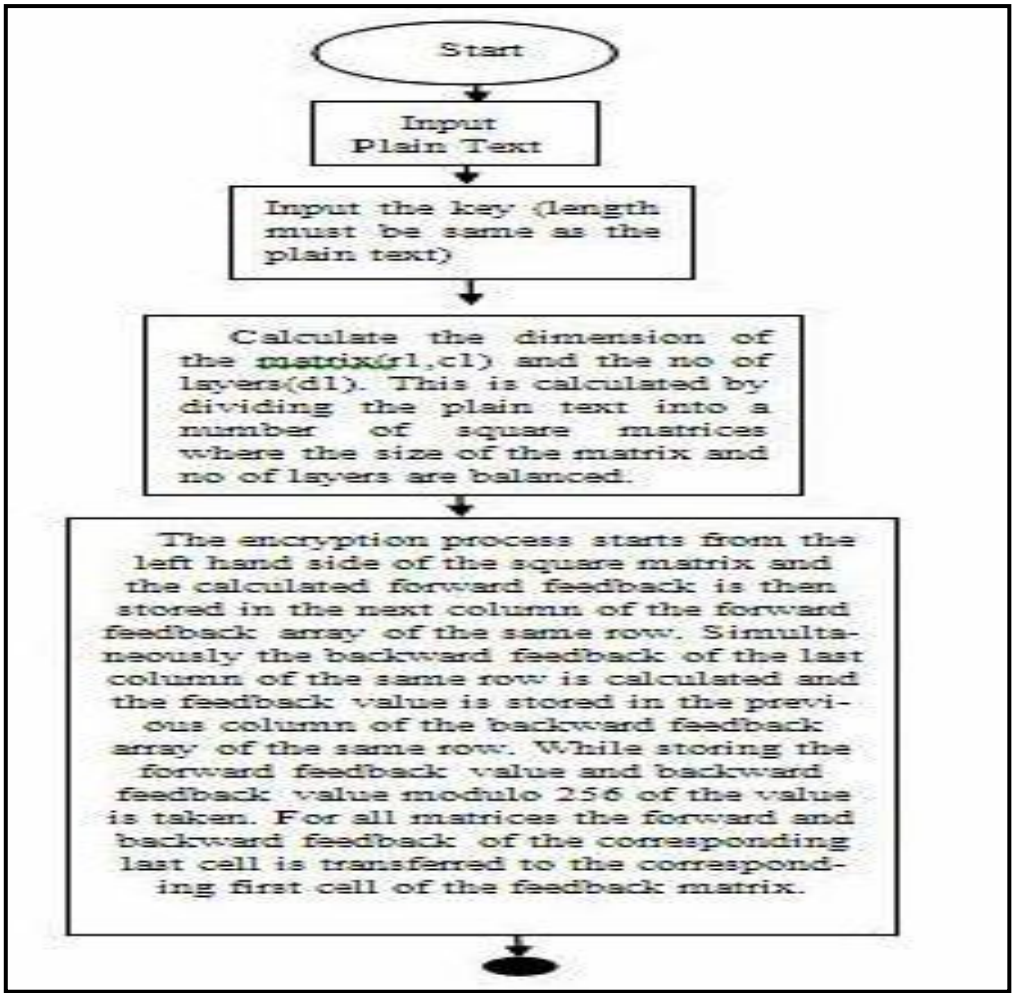

Fig 1. Encryption

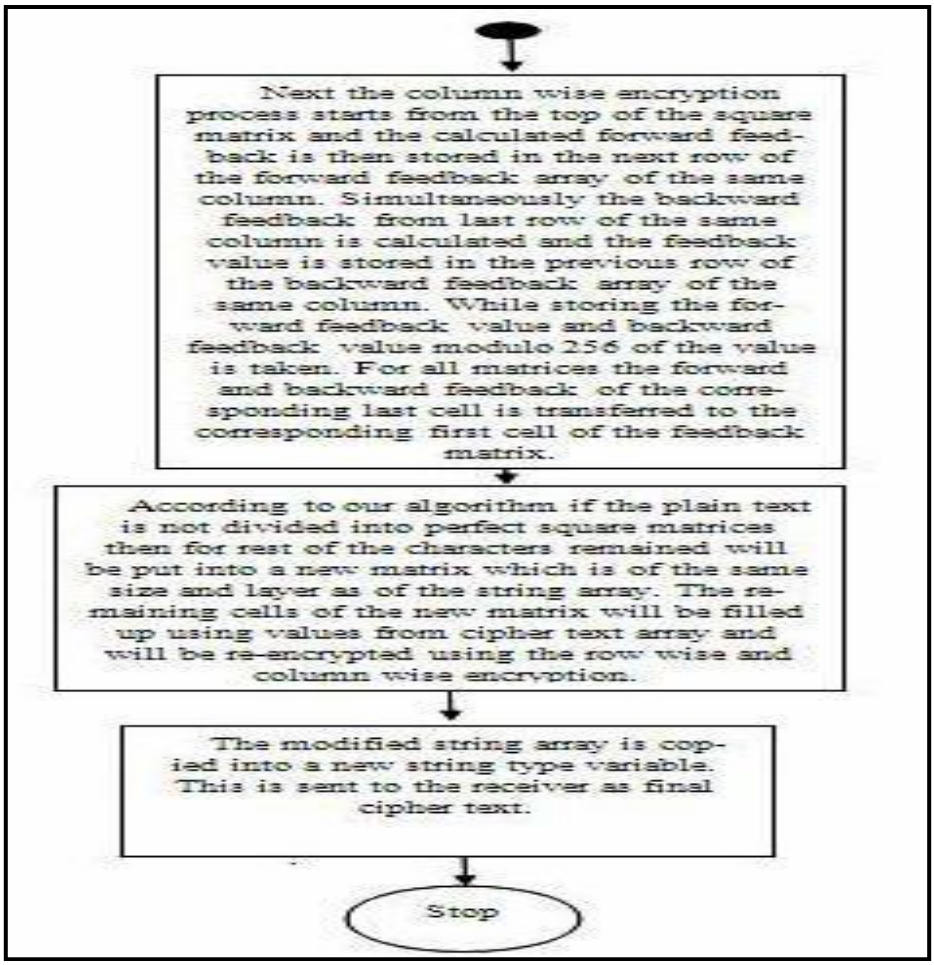

Fig 2. Encryption (contd.)

IV. Block Diagram of Decryption Method

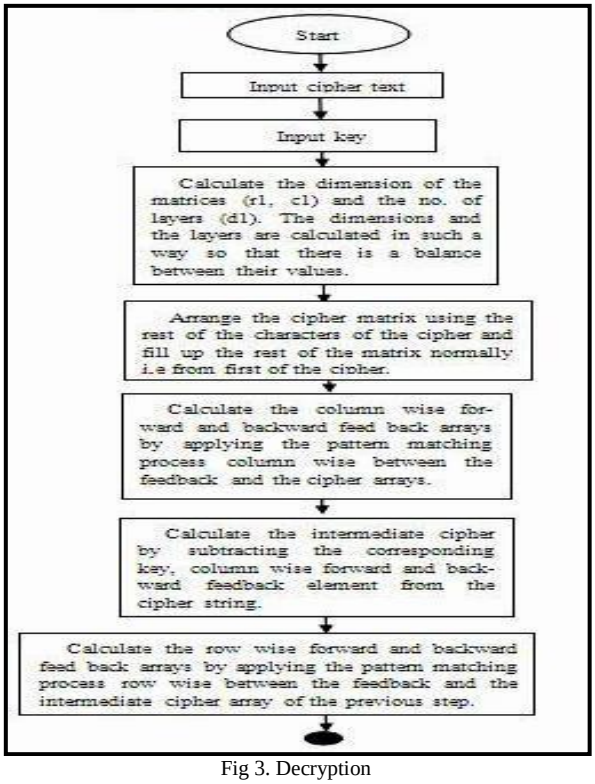

Fig 3. Decryption

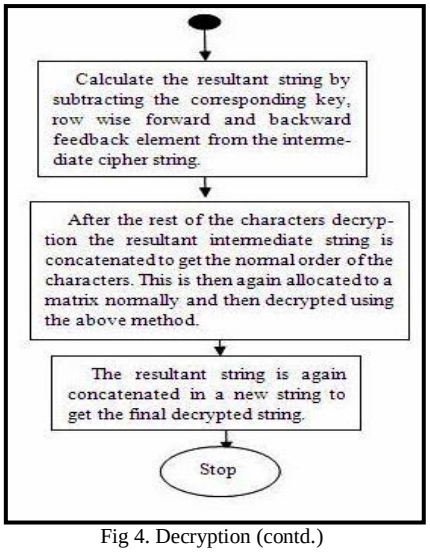

Fig 4. Decryption (contd.)

V. ALGORITHM USED IN 3DMWFES-1

strarr[][][] and cipharr[][][] is an array to store the sender's string as well as string created in the intermediate **Step**s of encryption.

strarr1[][][] is used store the required string for encryption of rest of the characters.

keyarr[][][] is an array to store the key provided by the sender.

ifxfarr[][][] is used to store the x-axis forward feedback which is initialized to 0.

ibxfarr[][][] is used to store the x-axis backward feedback which is initialized to 0.

ifyfarr[][][] is used to store the y-axis forward feedback which is initialized to 0.

ibyfarr[][][] is used to store the y-axis backward feedback which is initialized to 0.

str stores the string provided by user.

ciphstr stores the intermediate and final cipher text.

ifxf, ibxf, ifyf, ibyf stores temporary values for **ifxfarr[][][], ibxfarr[][][], ifyfarr[][][], ibyfarr[][][]** respectively.

n is the length of the string provided for encryption.

Algorithm to create dimension for the 3 dimension arrays

Step 1: if(n<=3) then

Step 2: Repeat the **Step**s 3 to 11 for i=1 to n

Step 3: Repeat the **Step**s 4 to 11 for j=1 to n

Step 4: temp=(n-(i*i*j));

Step 5: if(temp<0)

Step 6: break;

end if;

Step 7: if(temp<rem) then

Step 8: rem=temp;

Step 9: d1=j;

Step 10: r1=i;

Step 11: c1=i;

end if;

end if;

Step 12: else

Step 13: Repeat the **Step**s 14 to 24 for i=2 to n/2

Step 14: Repeat the **Step**s 15 to 24 for j=1 to i

Step 15: temp=(n-(i*i*j));

Step 16: if(temp<0) then

Step 17: break;

end if;

Step 18: if(temp<(n-temp)) then

Step 19: if(j>temp1 || j==temp1) then

Step 20: rem=temp;

Step 21: d1=j;

Step 22: r1=i;

Step 23: c1=i;

Step 24: temp1=j;

end if;

end if;

end else;

Step 25: if(r1==2 && d1==2)

Step 26: r1=3;

Step 27: c1=3;

Step 28: d1=1;

end if;

Step 29: if(n==8) then

Step 30: str=str+" ";

Step 31: key=key+" ";

Step 32: n=9;

end if;

Note: Special Care has been taken to avoid 2 X 2 X 2 matrix dimensions as it is impossible to create feedback arrays from cipher text during decryption. The required matrices are initialized using the dimensions calculated.

Algortihm for Encryption

Step 1: Take the plain text as input from user and load it into strarr array.

Step 2: Take the key as input from user and load it into the keyarr array.

Step 3: Continue steps 4 to 6 until all the values in the cells of the ifxfarr and ibxfarr are modified. Alternately update ifxfarr and ibxfarr after calculation of each cell of each matrix.

Step 4: Calculate the sum of the values in cell of strarr, keyarr, ifxfarr and ibxfarr and store the value in the next cell (columnwise) of the forward feedback array(ifxfarr).The process starts from the first cell of the array.

Step 5: After each cell calculation of ifxfarr calculate the value of ibxfarr in the following way. Calculate the sum of strarr, keyarr, ifxfarr and ibxfarr and store it to the previous cell of ibxfarr starting from the last column of the first row.

Step 6: After each row calculation, the calculation of next row begins. The last backward and forward feedback values of each row are passed to the last and first column of next row. The same is true for all the next layers.

Step 7: The last forward and backward value is passed to starting cell of ifxfarr and ibxfarr i.e. the first cell of ifxfarr and the last column of first row of ibxfarr.

Step 8: calculate and modify all the values of string array in the following way: strarr[k][i][j]= trarr[k][i][j]+ keyarr[k][i][j]+ifxfarr[k][i][j]+ ibxfarr[k][i][j];

Step 9: Calculate the values of all cells of ifyfarr and ibyfarr by repeating the procedure from 3 to 7 columnwise. For example, the process starts from the first row of the column in case of ifyfarr and last row of first column for ibyfarr.

Step 10: Calculate values of all cells of modified strarr as:

strarr[k][i][j]=strarr[k][i][j]+ keyarr[k][i][j]+ ifyfarr[k][i][j]+ ibyfarr[k][i][j];

Step 11: If these are any residual characters in the input string while allocating it to the strarr then add the residual characters to the beginning cells of the strarr and fill the rest of the matrix with cipher array characters. If there are no residual characters re-encrypt the modified strarr again.

Step 12: Encrypt the characters using the steps 3 to 10.

Step 13: After encryption rearrange the characters in correct order to get the final cipher.

Algorithm for Decryption

Step 1: Take the cipher and key as input and load it into the arrays strarr and keyarr.

Step 2: After calculating the dimensions of the array calculate the residual characters and load the residual characters (if any) at the beginning of the strarr array and fill the rest of the array normally using characters from the cipher string. Otherwise carry out the normal operation.

Step 3: Find the values of ifyfarr and ibyfarr in the following ways (Steps 5 to 8).Repeat the steps for all values of the above matrices:

Step 4: mid= (int) r1/2.

Step 5: if j>=mid and j<(c1-1)

ifyfarr[k][j+1][i]=starr[k][j][i].

and ibyfarr[k][j][i]=strarr[k][j+1][i] -strarr[k][j][i].

Step 6: if j<=mid and j>0 then ifyfarr[k][j][i]=strarr[k][j-1][i]- strarr[k][j][i] and ibyfarr[k][j-1][i]= strarr[k][j][i].

Step 7: if j==(r1-1) then if the element is the last element i.e cell [d1-1][r1-1] [c1-1] then it is copied to the first cell of the ifyfarr array. If the element is the last element of a layer other than the last layer then it is transferred to the first element of the next layer of ifyfarr. And if it is the last element of a column it is transferred to the first cell of the next column of ifyfarr.

Step 8: if j==0 then if the element is the first row of the last column element i.e. cell [d1-1][0][c1-1] then it is copied to the last row of first column of the ibyfarr array. If the element is the first row of the last column of a layer other than the last layer then it is transferred to last row of first column of the next layer of ibyfarr. And if it is the first element of a column it is transferred to thelast cell of the next column of ibyfarr.

Step 9: Update the strarr array as

strarr[k][i][j]=strarr[k][i][j]-((keyarr[k][i][j]+ifyfarr[k][i][j]+ibyfarr[k][i][j])%256).

Except for the cells ((k==0 && j==0 && i==1) or (k==0 && j==0 && i==(c1-2) or| (k==0 && j==1 && i==0)or (k==0 && j==1 && i==(c1-1)) or (c1==1 && d1>1 && k==1) or (c1==1 && d1==1)) and if(c1==2 && k==0 && j==0 && i==0) where k is the layer no,

j is the row no. and i is the column no.

Step 10: Some of the cells are to be manually calculated as follows [Steps have been provided as per the code as it was difficult to provide a writing for it]:

Step 11: if c1>3 then

ifyfarr[0][1][0]= (strarr[0][0][0] +keyarr[0][0][0])%256.

strarr[0][1][0]=strarr[0][1][0]-((keyarr[0][1][0]+ifyfarr[0][1][0]+ ibyfarr[0][1][0])%256).

ibyfarr[0][c1-2][0]=(strarr[0][c1-1][0] +keyarr[0][c1-1][0])%256.

strarr[0][c1-2][0]=strarr[0][c1-2][0] -((keyarr[0][c1-2][0]+ ifyfarr[0][c1-2][0] +ibyfarr[0][c1-2][0])%256);

Step 12:else if(c1==3) then

ifyfarr[0][1][0]=(strarr[0][0][0] +keyarr[0][0][0])%256;

ibyfarr[0][c1-2][0]=(strarr[0][c1-1][0]+keyarr[0][c1-1][0])%256;

strarr[0][1][0]=strarr[0][1][0]-((keyarr[0][1][0]+ifyfarr[0][1][0]+ibyfarr[0][1][0])%256);

Step 13:else if(c1==2) then

Step 14: if(d1==1) then

ibyfarr[0][0][0]=cipharr[d1-1][r1-1][0]-cipharr[0][0][c1-1];

end if[step 14].

strarr[0][0][0]=strarr[0][0][0]-((keyarr[0][0][0]+ifyfarr[0][0][0]+ibyfarr[0][0][0])%256);

ifyfarr[0][0][1]=ibyfarr[0][0][0];

strarr[0][0][1]=strarr[0][0][1]-((keyarr[0][0][1]+ifyfarr[0][0][1]+ibyfarr[0][0][1])%256);

ifyfarr[0][1][0]=(strarr[0][0][0] +keyarr[0][0][0])%256;

strarr[0][1][0]=strarr[0][1][0]-((keyarr[0][1][0]+ifyfarr[0][1][0]+ibyfarr[0][1][0])%256);

end else if[step 13].

Step 15:else if(c1==1)

Step 16:if(d1==1)

strarr[0][0][0]=strarr[0][0][0]-(4*keyarr[0][0][0])%256;

if(strarr[0][0][0]<0) then

strarr[0][0][0]=strarr[0][0][0]+512;

strarr[0][0][0]=strarr[0][0][0]/4;

ifyfarr[0][0][0]=(strarr[0][0][0] +keyarr[0][0][0])%256.

ibyfarr[0][0][0]=(strarr[0][0][0]+ keyarr[0][0][0]+ifyfarr[0][0][0])%256

end if(step 16).

Step 17:else

ifyfarr[1][0][0]=strarr[0][0][0] +keyarr[0][0][0];

ibyfarr[1][0][0]=strarr[0][0][0] +keyarr[0][0][0];

strarr[1][0][0]=strarr[1][0][0]-((keyarr[1][0][0]+ifyfarr[1][0][0]+ibyfarr[1][0][0])%256);

end else[step 17]

end else if[step 15].

Step 18: if(c1>1)

Step 19: if(c1!=2)

ifyfarr[0][0][1]=(strarr[0][c1-1][0]+keyarr[0][c1-1][0] +ifyfarr[0][c1-1][0])%256;

strarr[0][0][1]=strarr[0][0][1]-((keyarr[0][0][1]+ifyfarr[0][0][1]+ibyfarr[0][0][1])%256);

end if[step 19].

ibyfarr[0][c1-1][1]=(strarr[0][0][0] +keyarr[0][0][0]+ibyfarr[0][0][0])%256;

strarr[0][c1-1][1]=strarr[0][c1-1][1]-((keyarr[0][c1-1][1]+ifyfarr[0][c1-1][1]+ibyfarr[0][c1-1][1])%256);

end if[step 18].

Step 20: Check if any cell of strarr has a negative value. If so, add 256 to it.

Step 21: Find the values of ifxfarr and ibxfarr by applying the steps row wise instead of column wise i.e. cells are denoted in the form **[k][i][j]** instead of [k][j][i].

Step 22: Rearrange the characters in their original order (if needed) and input them again in the into strarr and redecrypt it using steps 3 to 21.

Step 23: Rearrange and concatenate the string in its correct order to get the plaintext needed.

VI. Results and Discussion

In the following table we take some plain text and the corresponding cipher text is shown. Here the plain text and the key are taken same and for next time for same plain text we take different key.

TABLE XXIII

PLAIN TEXT	CIPHER TEXT
aa	aY
bbbbbb	àX<,À†
Hello how are you?	IZ`ÅÃu'?{[bÂ»'^-
aa	M¥
bbbbbb	U A<
Hello how are you?	□븬瀨□A稗□談

It is seen that for same plain text different cipher text is generated so without knowing the key it is quite impossible for any intruder to generate the plain text from the cipher text without knowing the decryption process. Our algorithm can even encrypt ASCII 0 value and also a single character. The following table shows it. Here the key used is the character of ASCII value 1 according to the plain text size.

TABLE XXIII

PLAIN TEXT	CIPHER TEXT
Character of ASCII value 0 repeated 9 times	Œ¢"ŽÄ vœV
a	$

VII. Conclusion and Future Scope

The present method is tested on various types of files such as .doc, .jpg, .bmp, .exe, .com, .dbf, .xls, .wav, .avi and the results were quite satisfactory. The encryption and decryption methods work smoothly. The algorithm has the complexity of $O(n^3)$. For encryption and decryption of large files it takes a large time. We have also the aim to implement the encryption along z-axis. This can be also expanded to bit level encryption.

Acknowledgement

The authors are very much grateful to the Department of Computer Science for giving the opportunity to work on symmetric key Cryptography. One of the authors (Dr. Asoke Nath) 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.

References

1. Purnendu Mukherjee, Prabal Banerjee, Asoke Nath, Multi Way Feedback Encryption Standard Ver-I(MWFES-I) , International Journal of Advanced Computer Research(IJACR), Volume-3, Number-3, Issue-11, September 2013, Pages:176-182.
2. Asoke Nath, Debdeep Basu, Surajit Bhowmik, Ankita Bose and Saptarshi Chatterjee, Multi Way Feedback Encryption Standard Ver-2(MWFES-2),. Paper submitted in International Conference: IEEE WICT 2013 to be held in December 15-18, 2013 at Hanoi, Vietnam.
3. Asoke Nath, Payel Pal, Modern Encryption Standard Ver-IV(MES-IV), International Journal of Advanced Computer Research(IJACR), Volume-3, Number-3, Issue-11, September 2013, Page:216-223.
4. Asoke Nath, Bidhusundar Samanta, Modern Encryption Standard Ver-V(MES-V), International Journal of Advanced Computer Research(IJACR), Volume-3, Number-3, Issue-11, September 2013, Pages:257-264.
5. DriptoChatterjee, JoyshreeNath, SoumitraMondal, SuvadeepDasgupta and AsokeNath, Advanced Symmetric key Cryptography using extended MSA method: DJSSA symmetric key algorithm: Journal of Computing, Vol 3, Issue-2, Page 66-71,Feb(2011).
6. Dripto Chatterjee, Joyshree Nath, Suvadeep Dasgupta and AsokeNath, A new Symmetric key Cryptography Algorithm using extended MSA method: DJSA symmetric key algorithm,: Proceedings of IEEE International Conference on Communication Systems and Network Technologies, held at SMVDU(Jammu) 03-06 June,2011, Page-89-94(2011).
7. Neeraj Khanna, Joel James,JoyshreeNath, Sayantan Chakraborty, Amlan Chakrabarti and AsokeNath, New Symmetric key Cryptographic algorithm using combined bit manipulation and MSA encryption algorithm: NJJSAA symmetric key algorithm: Proceedings of IEEE CSNT-2011 held at SMVDU(Jammu) 03-06 June 2011, Page 125-130(2011).
8. Dripto Chatterjee, Joyshree Nath, Sankar Das, Shalabh Agarwal and AsokeNath, Symmetric key Cryptography using modified DJSSA symmetric key algorithm, Proceedings of International conference WorldComp 2011 held at Las Vegas 18-21 July 2011, Page-306-311, Vol-1(2011).

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

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

AUTHOR(S) PROFILE

Arijit Ghosh is presently pursuing M.Sc in Computer Science. He has completed his graduation from Jogesh Chandra Chaudhuri College. He is presently involved in research work in chaotic neural networks.

Prabhakar Chakraborty is pursuing Masters of Science (computer science) at St Xaviers College. He has completed his graduation from Bhairab Ganguly College. Presently involved in research work in parallel co-ordinate system.

Dr. Asoke Nath is presently working as Associate Professor in the Department of Computer Science, St Xavier's College (Autonomous), Kolkata [under University of Calcutta], India. His research areas are Cryptography and Network security, Steganography, Green Computing, E-learning methodologies and MOOCs. Dr. Nath has already published 112 papers in international and national journals and proceedings of conferences. Dr. Nath delivered lectures on Cryptography and Network Security in several International conferences in the US, Vietnam and in India.

Shamindra Parui is currently pursuing MSc in Computer Science (final year) from St. Xavier's College (kolkata).

I have completed my graduation (2013) in Computer Science from Prabhu Jagatbandhu College (under University of Calcutta).

