International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-4 Issue-13 December-2013

Modified Multi Way Feedback Encryption Standard (MWFES) Ver-I

Prabal Banerjee[1], Purnendu Mukherjee[2], Asoke Nath[3]

Abstract

Nath et al recently developed a method Multi Way Encryption Standard ver-1(MWFES ver-1[17]). In MWFES ver-1 the authors have introduced a method which is based on introduction of two way simultaneous feedback. In MWFES ver-1 the authors applied feedback from left hand side as well as from right hand side one after the other. The ASCII code of plain text, key and the corresponding forward and backward feedback is added to get some temporary number. The number is taken modulo with 256 to get the corresponding cipher text and also the feedback for the next available column. In MWFES ver-1 the authors did not consider any block size. It was essentially a stream cipher method. In the present modified MWFES ver-I the authors have added some modifications. Firstly the authors have taken a fixed block size for doing forward feedback as well as backward feedback. The initial feedback is calculated from user entered key and also using MSA algorithm. After finishing all blocks if the file contains 2 bytes then also the same MWFES method applied for characters separately and if 1 byte is left then the authors performed simply XOR operations between plain text and the key. After completion of one full round the entire file is reversed and the same feedback method applied using block-size=16. The authors noticed that there is a significant change found in the encrypted text. The authors shows that modified MWFES ver-I is better than MWFES ver-I. The encryption operation was performed n-number of times on the entire file. The decryption process is just the reverse process of encryption. The present feedback method can used be for encryption of SMS, password or any confidential message. The results shows the present method is free from Differential attack, known plain text attack or any kind of brute force attack.

Purnendu Mukherjee, Computer Science, St. Xavier's College (Autonomous), Kolkata, India.
Prabal Banerjee, Computer Science, St. Xavier's College (Autonomous), Kolkata, India.
Asoke Nath, Computer Science, St. Xavier's College (Autonomous), Kolkata, India.

Keywords

[Backward feedback, Forward feedback, Cipher Text, Plain Text, Two Way feedback, Multi way feedback, MWFES, MSA].

1. Introduction

Data confidentiality is now an open challenge to all of us. Sending confidential data from one computer to another computer is not at all safe. The attackers or the hackers are always ready to intercept any confidential data over internet. There are quite a number of common attacks in internet such as known plain text attack, middle man attack where a hacker is listening to both sender and the receiver and then retrieve confidential message and then divert it some one else. This may be sometimes very dangerous as both the sender and the receiver will be in problem after some time. The private/confidential data must be encrypted first and then it should be sent over the internet. E-mail is now one of the most important methods for communicating message from one person to other and hence one should be very careful about their e-mails. The hackers have already established their own websites where they keep all hacking s/w. Using those s/w one can hack password of any e-mail. If password is very weak then it takes hardly any time to hack the password of any e-mail. So it is recommended that the user must change his/her password from time to time and also the password should bit strong and uncommon. So therefore it is now a real challenge how we can send some confidential message through e-mail. The cryptographers always try to develop some good encryption algorithm and on the other hand the hackers try to apply brute force method to get back original message without knowing the actual decryption algorithm. The hackers mostly apply brute force and trail and error method to extract data from any encrypted data. The confidentiality of data is now a very important issue in data communication network. It is now no more safe to send any kind of authentic or confidential data in original from one place to another place. There is no guarantee that in between no one will tamper it. While doing e-business, e-banking, e-ticketing or money transfer through credit card or a debit card it must be ensured that the data should be fully protected and no unauthorized person should be able to tamper those data. 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. To protect data from intruder or hacker 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. Recently Nath et al [1-15] developed several symmetric key cryptography methods. Nath et al also developed a very recent method which is based on forward feedback as well as backward feedback along with some key. The method is called Multi way feedback encryption standard ver-1(MWFES ver-1)[17]. It was first time the authors developed a method where the feedback is taken from left to right and also from right to left simultaneously but one after the other. Initial forward and backward feedback was chosen some random character and it was constructed from user entered key and also MSA algorithm. In the present paper the authors have modified the method using a fixed block-size that may be variable also. from Nath et al[15] developed for the first time bit-wise as well as byte feedback encryption methods. In the present study the authors have developed a completely new concept that is the introduction of feedback both in forward direction (FF) i.e. from left to right and backward (BF) in alternate run. Moreover the authors have improvised the idea of the key. The concept of key is here is dynamic unlike the methods which were earlier proposed by Nath et al[1 -15]. 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 absolutely impossible for any intruder to extract any plain text from encrypted text using any brute force method. The results show that the present method is free from any kind of plain text or differential attack.

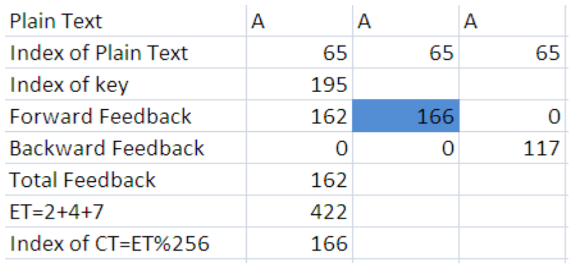
2. The Method

The novelty of the present technique is the application of the two way feedback. The initial forward and backward feedbacks are generated by a function on the partial key which is extracted from a random matrix generated by MSA algorithm.

The encryption process is done as a two way feedback control i.e. the control is passed alternately between the forward process and the backward process. The encryption starts with the forward process at the first character of the plain text and the control is transferred to the backward process after calculating and propagating the cipher text of one character. Similarly the backward process starts at the last character of the plain text and the control is transferred to the forward process after calculation and propagation of one cipher text character. Total number of times encryption done was totally random and it depends on encryption number generated from user entered key using MSA algorithm. In the forward process, the cipher text of the each character is propagated to the next character as the forward feedback. The forward process is resumed from this position once it gets the control back from backward process. Thus the forward process generates the forward feedback.

Similarly, in the backward process the cipher text of the each character is propagated to the previous character as the backward feedback. The backward process is resumed from this position once it gets the control back from forward process. Thus the backward process generates the backward feedback.
The total feedback corresponding to each character position is the sum of the forward and backward feedbacks. For block of length less than or equal to two, MWFES fails as it cannot be decrypted. So simple XOR encryption was used in this special case. The key corresponding to each character of the plain text is dependent on its corresponding total feedback, as the key is taken from the MSA matrix at the position denoted by the total feedback value. The cipher of each character is achieved by applying a mod function on the sum of plain text, total feedback and the key achieved from the MSA matrix using the total feedback.

An illustration of this method is shown below.
Example: plaintext: AAA Key: a



Plain Text	A	A	A
Index of Plain Text	65	65	65
Index of key	195		
Forward Feedback	162	166	0
Backward Feedback	0	0	117
Total Feedback	162		
ET=2+4+7	422		
Index of CT=ET%256	166		

Plain Text	A	A	A
Index of Plain Text	65	65	65
Index of key	195		47
Forward Feedback	162	166	0
Backward Feedback	0	229	117
Total Feedback	162		117
ET=2+4+7	422		229
Index of CT=ET%256	166		229

Plain Text	A	A	A
Index of Plain Text	65	65	65
Index of key	195	70	47
Forward Feedback	162	166	18
Backward Feedback	0	229	117
Total Feedback	162	139	117
ET=2+4+7	422	530	229
Index of CT=ET%256	166	18	229

Plain Text	A	A	A
Index of Plain Text	65	65	65
Index of key	195	70	47
Forward Feedback	162	166	18
Backward Feedback	18	229	117
Total Feedback	162	139	117
ET=2+4+7	422	530	229
Index of CT=ET%256	166	18	229

Plain Text	A	A	A
Index of Plain Text	65	65	65
Index of key	195	70	114
Forward Feedback	58	166	18
Backward Feedback	18	229	117
Total Feedback	162	139	135
ET=2+4+7	422	530	314
Index of CT=ET%256	166	18	58

Plain Text	A	A	A
Index of Plain Text	65	65	65
Index of key	25	70	114
Forward Feedback	58	166	18
Backward Feedback	18	229	166
Total Feedback	76	139	135
ET=2+4+7	166	530	314
Index of CT=ET%256	166	18	58

Plain Text	A	A	A
Index of Plain Text	65	65	65
Index of key	25	70	114
Forward Feedback	58	166	18
Backward Feedback	18	229	166
Total Feedback	76	139	184
ET=2+4+7	166	530	368
Index of CT=ET%256	166	18	112

The following major modifications made over MWFES Version-I:
The MWFES algorithm was implemented block wise instead of considering the entire plaintext as a stream to be encrypted.
The whole encryption process was done random number of times on the basis of the MSA matrix.
A Generalized Inverse Modular function was implemented replacing the previous case specific approach.
After completion of encryption, the output was reversed and the algorithm again applied upon.

3. Encryption Algorithm

The present method is dependent both on the text-key and the plaintext. From the text-key a randomization matrix is generated using the method developed by Nath et al [1]. The encryption algorithm of MWFES is given as follows:

Step 1: Input plaintext filename or plaintext itself and save it in PT. Let its size be ‘size’.
Step 2: Input text-key string. Let it be ‘input’. Call MSA_generate(MSA_matrix, input) where MSA_matrix is an uninitialized 16x16 matrix. The function generates MSA matrix according to input and initializes MSA_matrix
Step 3: Initialise enc_times with the random number of times encryption to be done, decided by the MSA algorithm
Step 4: The KEY[] is taken as the ‘size’ places from last position of the MSA_matrix to the jth position of the MSA_matrix for every iteration, where j =(enc_times-1)*size.
Step 5 : For all the blocks that can be extracted from the plaintext of fixed length block_size, apply block_encrypt(block_size,matrix,pt,ct,key)
Step 6: If the remainder plaintext length is of size less than or equal to 2, apply false_encrypt(key,ct,pt,matrix,size). Else, apply

block_encrypt(remaining_size,matrix,pt,ct,key) on the last block.
Step 7: Convert all derived ciphertext bytes to plaintext bytes if further encryption is to be done.
Step 8: enc_times=enc_times-1
Step 9: if enc_times>0, goto step 4
Step 10: Reverse the plaintext, make it the ciphertext and repeat steps 3 to 9.
Step 11: Output the derived ciphertext to a desired file to get the encrypted file.

Function block_encrypt(size,matrix,pt,ct,key)

Step 1: Initial forward feedback fwdfb[0] is taken as the sum of even positions of 'size' places from j-th position of MSA_matrix. This sum is divided by 2 and is added to the last byte value of the plaintext which is stored as modulo 256. Here, j =((No. of times encryption to be done)-1)*size.
Step 2: Initial backward feedback bckfb[size-1] as the sum of odd positions of 'size' places from j-th position of MSA_matrix. It is then added to the last plaintext byte and saved as modulo 256. Here, j =((No. of times encryption to be done)-1)*size.
Step 3: set i=0
Step 4: Total feedback for i-th bit totfb[i] is taken as the sum of bckfb[i] and fwdfb[i]
Step 5: ET[i] = totfb[i] + key_matrix (MSA_matrix, totfb[i]) + pt[i]
Step 6: ct[i]=ET[i] mod 256
Step 7: key[i]=key_matrix(MSA_matrix,totfb[i]);
Step 8: fwdfb[(i+1) mod size]=ET[i]%256
Step 9: totfb[size-1-i]=fwdfb[size-1-i]+bckfb[size-1-i]
Step 10: ET[size-1-i] = totfb[size-1-i] + key_matrix (MSA_matrix, totfb[size-1-i]) + pt[size-1-i]
Step 11: ct[size-1-i]=ET[size-1-i] mod 256
Step 12: key[size-1-i] = key_matrix(MSA_matrix,totfb [size-1-i])
Step 13: if size-1-(i+1))>=0 then temp = (size-1-(i+1))
Otherwise temp = size-1
Step 14: bckfb [temp] = ET[size-1-i] mod 256
Step 15 : i=i+1
Step 16: if i<size then goto step 4
Step 17: totfb[size-1]=fwdfb[size-1]+bckfb[size-1]
Step 18: ET[size-1] = totfb[size-1] + key_matrix (MSA_matrix,totfb[size-1]) + pt[size-1]
Step 19: ct[size-1]=ET[size-1] mod 256

Function false_encrypt(key,ct,pt,matrix,size)

Step 1: i=0
Step 2: k=key_matrix(matrix,key[i])
Step 3: ct[i]=pt[i]^k
Step 4 : i=i+1
Step 5: if i<size, goto step 2

Function key_matrix (matrix, position)

Step 1: position = position mod 256
Step 2: Return the (position mod 16)-th value of the floor(position/16)-th row of the matrix.
Step 3: Return control to calling function

Function MSA_generate(matrix , string)

Step 1: Fill the matrix with values from 0 to 255 in order in a row-major way.
Step 2: Randomize the matrix using MSA algorithm[1] based on key string.

4. Decryption Algorithm

Step 1: Input ciphertext filename and save the file contents in CT. Let its size be 'size'.
Step 2: Input key string. Let it be 'input'. Call MSA_generate(MSA_matrix, input) where MSA_matrix is an uninitialized 16x16 matrix. The function generates MSA matrix according to input and initializes MSA_matrix(same process as during encryption).
Step 3: Initialise dec_times with the random number of times decryption to be done, decided by the MSA algorithm
Step 4: The KEY[] is taken as the 'size' places from 0^{th} position of the MSA_matrix to the jth position of the MSA_matrix for every iteration, where j =(dec_times-1)*size.
Step 5 : For all the blocks that can be extracted from the plaintext of fixed length block_size, apply block_encrypt(block_size,matrix,pt,ct,key)
Step 6: If the remainder plaintext length is of size less than or equal to 2, apply false_encrypt(key,ct,pt,matrix,size). Else, apply block_encrypt(remaining_size,matrix,pt,ct,key) on the last block.
Step 7: Convert all derived ciphertext bytes to plaintext bytes if further encryption is to be done.
Step 8: enc_times=enc_times-1
Step 9: if enc_times>0, goto step 4
Step 10: Reverse the plaintext, make it the ciphertext and repeat steps 3 to 9.
Step 11: Output the derived plaintext to a desired file to get the decrypted file.

Function block_decrypt(size,matrix,pt,ct,key)

Step 1: The total feedback F for the (size-1)th position is calculated as F=(CT[size-2]+CT[0])%256
Step 2: K=key_matrix(MSA_matrix,F);
Step 3: The PT[size-1] is achieved by invoking the invMod function as:
PT[size-1]=invMod(CT[size-1],F,K);

Step 4: The backward feedback ‘bf’ is the sum of odd positions of the KEY[]. It is then added to the last plaintext byte and saved as modulo 256.
Step 5: The previous value of F at (size-1) position is calculated as F= (CT[size-2]+bf)%256
Step 6: The previous value of CT[size-1] is calculated
CT[size-1]=(pt[size-1]+F+key_matrix(MSA_matrix,F))%256
Step 7: The total feedback F for the 0th position is calculated as F=(CT[size-1]+CT[1])%256
Step 8: K=key_matrix(MSA_matrix,F)
Step 9: The PT[0] is achieved by invoking the invMod function as: PT[0]=invMod(CT[0],F,K)
Step: 10: Forward feedback ‘ff’ is taken as the sum of even positions of the key. This sum is divided by 2 and is added to the last byte value of the plaintext which is stored as modulo 256.
Step 11: The previous value of CT[0] is calculated
CT[0]=(PT[0]+ff+key_matrix(MSA_matrix,ff))%256
Step 12: The initial value of CT[size-1] is calculated as
ct[size-1]=(pt[size-1] +bf+key_matrix(MSA_matrix,bf))%256
Step 13: j=1
Step 14: if(j%2==0) goto Step 21.
Step 15: k=j/2 +1 (k assumes postion values in the box starting from the 2nd position till the middle)
Step 16: F=(CT[k-1]+CT[k+1])%256
Step 17: K=key_matrix(MSA_matrix,F)
Step 18: PT[k]=invMod(ct[k],F,K)
Step 19: Calculate the previous value of ct[k] as:
ct[k]=(pt[k]+ct[k-1]+key_matrix(MSA_matrix,ct[k-1]))%256
Step 20: goto Step 26.
Step 21: k=size-(j/2 +1) (k assumes postion values in the box starting from the 2nd last position till the middle).
Step 22: F= (ct[k-1]+ct[k+1])%256
Step 23: K=key_matrix(MSA_matrix,F)
Step 24: PT[k]=invMod(CT[k],F,K)
Step 25: Calculate the previous value of ct[k] as:
ct[k]=(pt[k]+ct[k+1]+key_matrix(MSA_matrix,ct[k+1]))%256
Step 26: j=j+1; if(j<=size-2) goto Step 14.

Function false_decrypt(key,ct,pt,matrix,size)
Step 1: i=0
Step 2: k=key_matrix(matrix,key[i])
Step 3: pt[i]=ct[i]^k
Step 4 : i=i+1
Step 5: if i<size, goto step 2

Function invMod(c, f, k)
Step 1: ktf=(k+f)%256;
Step 2 ctinv=256-c;
Step 3: ptinv=(ktf+ctinv)%256;
Step 4: pt=256-ptinv;
Step 5 return pt;

5. Randomization of matrix using Meheboob, Saima and 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()

6. Results and Discussion

The present method applied on various text files such as all characters ASCII ‘0’, all characters ASCII ‘1’ and the frequency distribution of the encrypted text is shown the below spectral graph. Beside each graph, a corresponding spectral graph of a similar file when encrypted using the unmodified MWFES Version-I is shown for comparison and clarity.

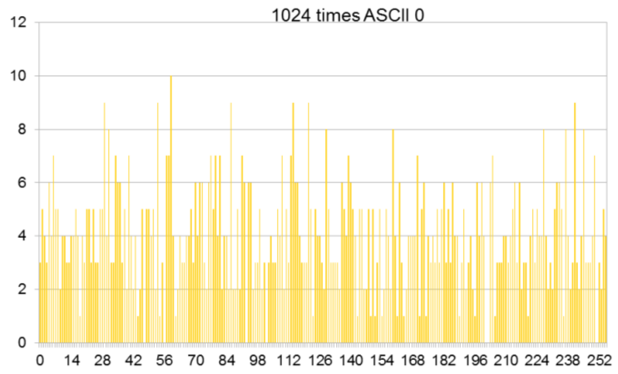


Fig-1: Frequency Spectral analyses of Plain Text file containing 1024 ASCII ‘0’ and Key=’a’ using original MWFES Ver-I.

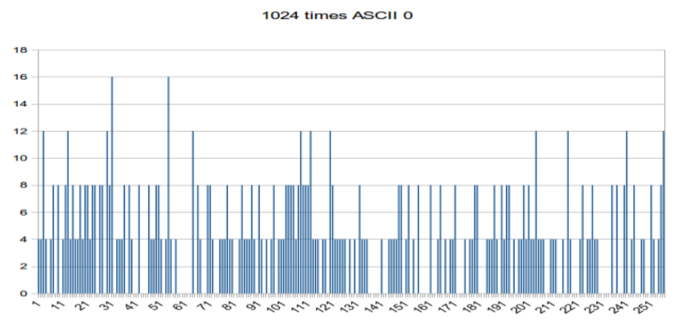


Fig-2: Frequency Spectral analyses of Plain Text file containing 1024 ASCII '0' and Key='a' using Modified MWFES Ver-I.

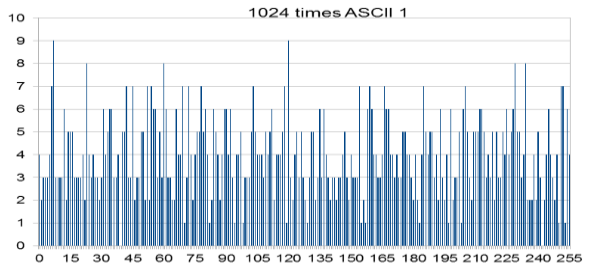


Fig-3: Frequency Spectral analyses of Plain Text file containing 1024 ASCII '1' and Key='a'using original MWFES Ver-I.

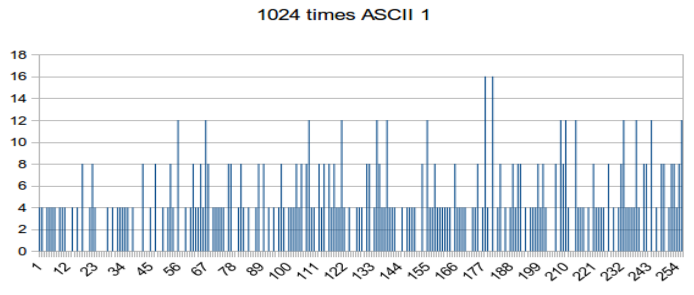


**Fig-4: Frequency Spectral analyses of Plain Text file containing 1024 ASCII '1' and Key='a'using modified MWFES Ver-
I.**

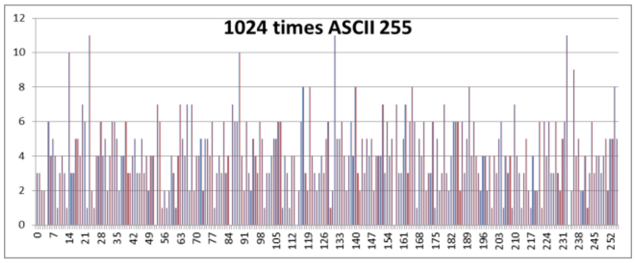


Fig-5: Frequency Spectral analyses of Plain Text file containing 1024 ASCII '255' and Key='a' using original MWFES Ver-I.

Table 1: Sample plaintexts and their corresponding ciphertexts.

PLAINTEXT	ORIGINAL	MODIFIED
AAA	àzk	‰e®
AAB	õ¸’	Òïÿ
ABA	ku	/¯à
BAA	H":	¼
BBB	ú~	q—‘
AAAAAAAAAA AAAAAA	ÿã\Ëòïôè§ñ± È?º	0ÍÕ_¥º{ ÀL0`º±
AAAAAAAAAA AAAAAB	…û8?^ è[ZÇî	Z’?v /Ëa)Åôñˆ Ñ#

Table 2: A Plain Text file containing a paragraph and its encrypted file after encryption.

The Society of Jesus, a Christian Religious Order founded by Saint Ignatius of Loyola in 1540, has been active in the field of education throughout the world since its origin. In the world, the Jesuits are responsible for 3,897 Educational Institutions in 90 countries. These Jesuit Educational Institutions engage the efforts of approximately 1,34,303 teachers, educating approximately 29,28,806 students.	XÿaœL&Â· • xÅí"wñXî½ä¤ ——————————— b*¸#EvÝSA'òšRÀÇò'oàX\Fä×Ø ,ääh6□ eaRˆcÚ}(¼‘ÿ(Ä[°ã)Bé(çeöã>ò- þd^é,¾ÉÈo›É‘€›Owý©a‡×Ë²Â Î=Ü’×--4’û¼]·×h9—]FõK³„ïÀIÃ?qñ4¨Ù½ü«mByNÊ ¡ZÁsŒºÁ=o‘ò•xh±v# ÷t_·¨pV³- F*ÐM • ÐÈFÃÔÕé!2à¯P§)òI8¡ kÍ]˜Å.˜s;]=áž • û22t□±LM´}^¸j®#qÕ˜•,09Œœi tˆ’ÇsýŠ‹à£š'ísÑˆN Ñ¬WnT@5/(6ðîÕTŸ10(ŒSŠÃ BÔ»Ùºk1“ÛÇø7ôÿÎ_ØS)3ŠÌï& ±û¥'2 ©$9ùêrÈÐ+„æµÌC+B ñãƒëˆÌ¶ø¢ÉP`]nxð^åLÝÍ¼xÁú ‰#“7[&T×î□ëI

Figures 1-4 show that encrypting similar files of same size and using the same key yields considerably varied results. Even if the input is multiple repetitions of the same ASCII code, the ciphertext is a mixture of almost all possible ASCII codes. This makes our system very hard to crack. A single change in the plaintext file can yield completely different results as shown in Table 1. Table 2 is a sample plaintext and its corresponding ciphertext.

7. Conclusion and Future Scope

The present method is tested on various types of files such as and the results found quite satisfactory. The encryption and decryption methods work smoothly. The result shows that MWFES ver-1[17] and modified MWFES ver-1 are quite different. The brute force method or known plain text attack is not possible in modified MWFES ver-1 method. In the present method the encrypted text cannot be decrypted without knowing the exact initial random matrix. The size of random matrix taken is 16x16. The numbers in 16x16 may be arranged in 256! Ways. To complete the whole process the authors have chosen any of the random matrix depending on the user entered text-key. The results shows that if there is one change in the plain text then decrypted texts become totally different. The present modified MWFES ver-I method may be applied to encrypt any short message, password, confidential key. Due to introduction of block size for using One can apply this method to encrypt data in sensor networks. The work is already been in progress to make the key further complex by using some non-linear operation and modulo operation. Nath et al already completed MWFES ver-II[18] and ver-III[19] the feedback entered in random order and also a skip is entered. Work is also in progress to use multiple keys and block encryption method instead of stream cipher method.

Acknowledgment

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 (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.

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 July 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 Las Vegas 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 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, Advanced 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] Ultra Encryption Standard(UES) Version-IV: New Symmetric Key Cryptosystem with bit-level columnar Transposition and Reshuffling of Bits, Satyaki Roy, Navajit Maitra, Joyshree Nath, Shalabh Agarwal and Asoke Nath, International Journal of Computer Applications(IJCA)(0975-8887) USA Volume 51-No.1.,Aug, Page. 28-35(2012).

[13] Bit Level Encryption Standard(BLES): Version-I, Neeraj Khanna, Dripto Chatterjee, Joyshree Nath and Asoke Nath, International Journal of Computer Applications(IJCA)(0975-8887) USA Volume 52-No.2.,Aug, Page.41-46(2012).

[14] Bit Level Generalized Modified Vernam Cipher Method with Feedback : Prabal Banerjee, Asoke Nath, Proceedings of International Conference on Emerging Trends and Technologies held at Indore, Dec 15-16,2012.

[15] Advanced Symmetric Key cryptosystem using Bit and Byte Level encryption methods with Feedback: Prabal Banerjee, Asoke Nath, Proceedings of International conference Worldcomp 2013 held at Las Vegas, July 2013.

[16] Cryptography and Network Security, William Stallings, Prentice Hall of India.

[17] Prabal Banerjee,Purnendu Mukherjee,Asoke Nath," Modified Multi Way Feedback Encryption Standard (MWFES) Ver-I", International Journal of Advanced Computer Research (IJACR),Volume-3, Number-3, Issue-11, September-2013.

[18] Asoke Nath, Debdeep Basu, Surajit Bhowmik, Ankita Bose4 Saptarshi Chatterjee5," Multi Way Feedback Encryption Standard Ver-2(MWFES-2)", International Journal of Advanced Computer Research (IJACR), Volume-3, Number-4, Issue- 13, December-2013.

[19] Multi Way Feedback Encryption Standard Ver-3, Asoke Nath, Debdeep Basu, Ankita Basu, Saptarshi Chatterjee, Surojit Bhowmik, published in WICT 2013 IEEE proceedings held in Dec 15-18, 2013 at Hanoi, Vietnam.

Asoke Nath is the Associate Professor in Department of Computer Science. Apart from his teaching assignment he is involved with various research work in Cryptography, Steganography, Green Computing, E-learning. He has presented papers and invited tutorials in different International and National conferences in India and in abroad.

Prabal Banerjee Pursuing Bachelor of Science (Computer Science Honours) at St. Xavier's College (Autonomous), Kolkata. Presently involved in research work in Bit level encryption methods.

Purnendu Mukherjee is a pursuing his Masters in Computer Science at St. Xavier's College (Autonomous), Kolkata. He is actively doing research in Cryptography and Machine Learning.