International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-3 Issue-11 September-2013

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

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

Abstract

In the present paper the authors have introduced a new symmetric key cryptographic method which is based on introduction of two way simultaneous feedback methods. Nath et al [14] already introduced both bit level and byte level feedback method and they have shown that the results were quite satisfactory. In the present study the authors tried to apply both forward feedback and backward feedback. The initial feedback is random and that will be determined by user entered key. The process starts from Left hand side of the plain text. The forward feedback (FF) and the key value is generated from user entered key string. The ASCII value of plain text is added with key and forward feedback and backward feedback (BF) to obtain intermediate cipher text. The intermediate cipher text is taken modulo operation with 256 to get cipher text. This cipher text is taken as feedback for the next column. In the second round we calculate the cipher text from the RHS. Here we use backward feedback (BF) from user entered text string. The ASCII code of plain text (last byte), key, BF and FF (initially 0) is added to obtain intermediate cipher text. It is then taken modulo with 256 to get cipher text. In the 3rd round we again do the same process with FF from 2nd byte from the beginning. The entire operation is continued till all feedback (both FF and BF) becomes non zero or modified. The encryption operation we perform n-number of times on the entire file. The decryption process is just the reverse process of encryption. The present feedback method is totally a new method and this can used for encryption of SMS, password in sensor networks. 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.
AsokeNath, 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.

1. Introduction

In the present information and communication technology (ICT) it is very difficult to send any confidential message from one computer to another without any encryption. Internet is now almost open access to anybody. There is a common attack called middleman attack where an intruder can listen to both the sender and the receiver and then divert the message to a different person and can manipulate the actual message. So it is almost mandatory to encrypt the message of the sender and then to send to the receiver. 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 tools communicating message from one person to other and hence one should be very careful about their e-mails. 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 to the receiver. The hackers have already developed different sites where they keep s/w which can be used to crack any password of e-mail. It is possible to crack any password of any e-mail using those s/w. So therefore it is now a real challenge how we can send some confidential message through e-mail. 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 is now a common practice in any academic institution to send students' marks, exam question papers, minutes of some confidential meeting, bank statement over e-mail. But in the present situation this method is not fully secured as anybody can intercept the data from internet and misuse it. 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. The disaster may happen in any corporate sector, business house when the data is sent from one computer to other computer 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 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. Nath et al [1-15] developed several symmetric key cryptography methods. 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.
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.

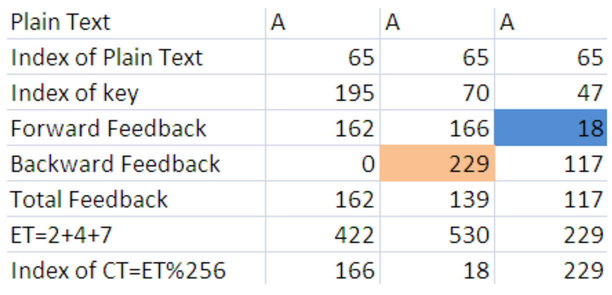
Similarly, in the backward process the cipher text of the each character is propagated to the previous character as the backward feedback. The backward process is resumed from this position once it gets the control back from forward process. Thus the backward process generates the backward feedback.
The total feedback corresponding to each character position is the sum of the forward and backward feedbacks.

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.

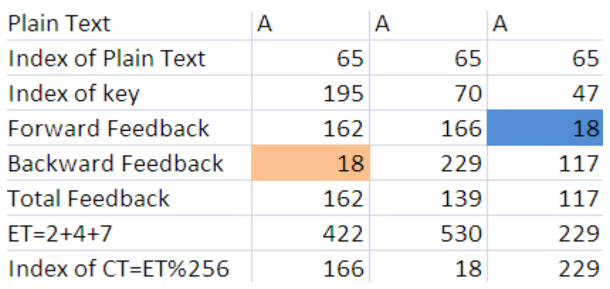
An illustration of this method is shown below.
Table-1 :Encryption of Plain Text with Random key
Example: plaintext: AAA
Key: a

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		

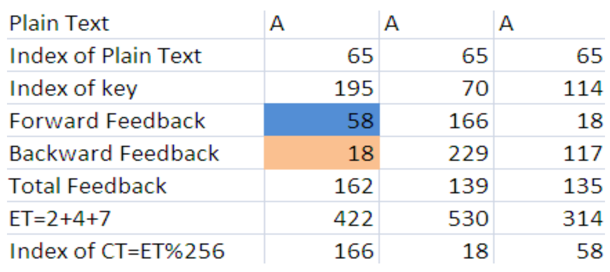
Plain Text	A	A	A
Index of Plain Text	65	65	65
Index of key	195		47
Forward Feedback	162	166	0
Backward Feedback	0	229	117
Total Feedback	162		117
ET=2+4+7	422		229
Index of CT=ET%256	166		229



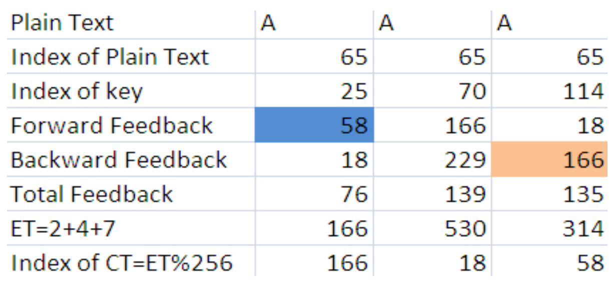
Plain Text	A	A	A
Index of Plain Text	65	65	65
Index of key	195	70	47
Forward Feedback	162	166	18
Backward Feedback	0	229	117
Total Feedback	162	139	117
ET=2+4+7	422	530	229
Index of CT=ET%256	166	18	229



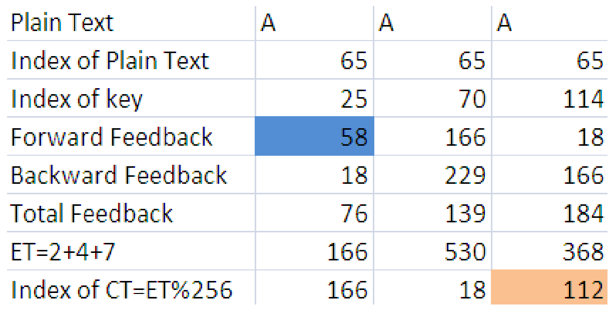
Plain Text	A	A	A
Index of Plain Text	65	65	65
Index of key	195	70	47
Forward Feedback	162	166	18
Backward Feedback	18	229	117
Total Feedback	162	139	117
ET=2+4+7	422	530	229
Index of CT=ET%256	166	18	229



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

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: 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 decryption to be done)-1)*size.
Step 4: 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 decryption to be done)-1)*size.
Step 5: set i=0
Step 6: Total feedback for i-th bit totfb[i] is taken as the sum of bckfb[i] and fwdfb[i]
Step 7: ET[i] = totfb[i] + key_matrix (MSA_matrix, totfb[i]) + pt[i]
Step 8: ct[i]=ET[i] mod 256
Step 9: key[i]=key_matrix(MSA_matrix,totfb[i]);
Step 10: fwdfb[(i+1) mod size]=ET[i]%256
Step 11: totfb[size-1-i]=fwdfb[size-1-i]+bckfb[size-1-i]
Step 12: ET[size-1-i] = totfb[size-1-i] + key_matrix (MSA_matrix, totfb[size-1-i]) + pt[size-1-i]
Step 13: ct[size-1-i]=ET[size-1-i] mod 256
Step14:key[size-1-i] = key_matrix(MSA_matrix,totfb [size-1-i])
Step 15: if size-1-(i+1))>=0 then temp = (size-1-(i+1))
Otherwise temp = size-1
Step 16: bckfb[temp] = ET[size-1-i] mod 256
Step 17 : i=i+1
Step 18: if i<size then goto step 6
Step 19: totfb[size-1]=fwdfb[size-1]+bckfb[size-1]
Step 20: ET[size-1] = totfb[size-1] + key_matrix (MSA_matrix,totfb[size-1]) + pt[size-1]
Step 21: ct[size-1]=ET[size-1] mod 256
Step 22: Convert all derived ciphertext bytes to plaintext bytes if further encryption is to be done.
Step 23: Output the derived ciphertext to a desired file to get the encrypted file.

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.
Step2: 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: The KEY[] is taken as the ‘size’ places from 0th position of the MSA_matrix to the jth position of the MSA_matrix for every iteration, where j =((No. of times decryption to be done)-1)*size.
Step 4: The total feedback F for the (size-1)th position is calculated as F=( CT[size-2]+CT[0] )%256
Step 5: K=key_matrix(MSA_matrix,F);
Step 6: The PT[size-1] is achieved by invoking the invMod function as:
PT[size-1]=invMod(CT[size-1],F,K);
Step 7: 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 8: The previous value of F at (size-1) position is calculated as F= ( CT[size-2]+bf )%256
Step 9: The previous value of CT[size-1] is calculated
CT[size-1]=(pt[size-1]+F+key_matrix(MSA_matrix,F))%256
Step 10: The total feedback F for the 0th position is calculated as F=( CT[size-1]+CT[1])%256
Step 11: K=key_matrix(MSA_matrix,F)
Step 12: The PT[0] is achieved by invoking the invMod function as: PT[0]=invMod(CT[0],F,K)
Step: 13: 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 14: The previous value of CT[0] is calculated
CT[0]=( PT[0]+ff+key_matrix(MSA_matrix,ff) )%256
Step 15: The initial value of CT[size-1] is calculated as
ct[size-1]=(pt[size-1]+bf+key_matrix(MSA_matrix,bf))%256
Step 16: j=1
Step 17: if(j%2==0) goto Step 24.
Step 18: k=j/2 +1 (k assumes postion values in the box starting from the 2nd position till the middle)
Step 19:F=( CT[k-1]+CT[k+1] )%256
Step 20: K=key_matrix(MSA_matrix,F)
Step 21: PT[k]=invMod(ct[k],F,K)
Step 22: Calculate the previous value of ct[k] as:
ct[k]=(pt[k]+ct[k-1]+key_matrix(MSA_matrix,ct[k-1]))%256
Step 23: goto Step 29.
Step 24: k=size-(j/2 +1) (k assumes postion values in the box starting from the 2nd last position till the middle).
Step 25: F= ( ct[k-1]+ct[k+1] )%256
Step 26: K=key_matrix(MSA_matrix,F)
Step 27: PT[k]=invMod(CT[k],F,K)
Step 28: Calculate the previous value of ct[k] as:
ct[k]=(pt[k]+ct[k+1]+key_matrix(MSA_matrix,ct[k+1]))%256
Step 29: j=j+1; if(j<=size-2) goto Step 17.
Step 30: If another round of decryption is to be done then CT=PT and then goto Step 3.
Step 31: Output the derived plaintext to a desired file to get the decrypted file.
```

Function invMod(c, f, k)

```
Step 1:if c<f, goto Step 5
Step 2: i=c-f;
Step 3: if i<k then i=i+256;
Step 4: goto Step 12
Step 5: if (f>256 ) then goto Step 9
Step 6: i=c-f+256;
Step 7: if i<k then i=i+256;
Step 8:goto Step 12
Step 9: i=c-f+512
Step 10: if(i>=k+256) then i=i-256, goto Step 12
Step 11: if i<k then i=i+256
Step 12: return i
```

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 Nathet.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.

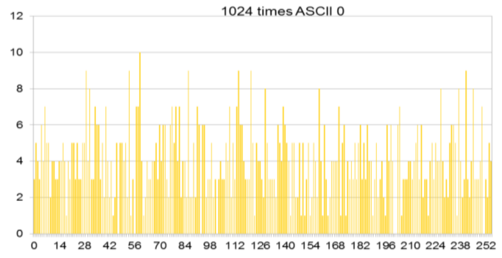


Fig-1: Frequency Spectral analyses of Plain Text file containing 1024 ASCII '0' and Key='a'.

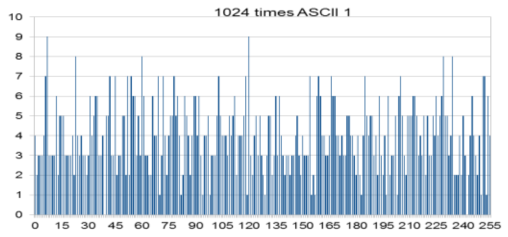


Fig-2: Frequency Spectral analyses of Plain Text file containing 1024 ASCII '1' and Key='a'.

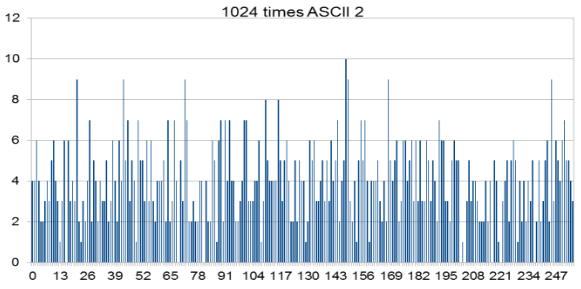


Fig-3: Frequency Spectral analyses of Plain Text file containing 1024 ASCII '2' and Key='a'.

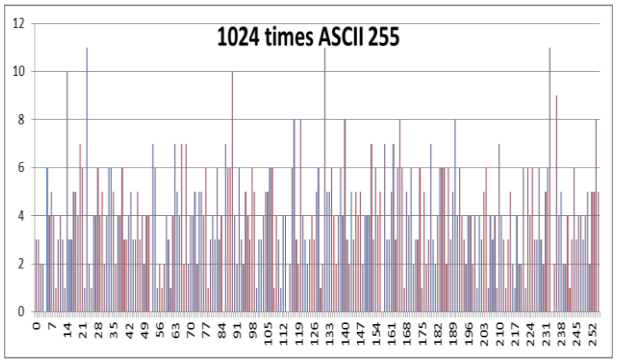


Fig-4: Frequency Spectral analyses of Plain Text file containing 1024 ASCII '255' and Key='a'.

Table 1 : Sample plaintexts and their corresponding ciphertexts

Plaintext	Cipher text
AA	”;
AB	¢ •
AAA	àzk
AAB	õ¸’
ABA	ku
BAA	H":
BBB	ú~
AAAAAAAAAAAAAAAA	ÿã\Ëòïôè§ñ± È?º
AAAAAAAAAAAAAAAB	…û8?^ • è[ZÇî

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

It was Fr. O'Neill (Rector: 1904-1913) who gave the College its crest and the motto: Nihil Ultra, a motto, significant and suggestive of a noble ideal, of an unconquerable hope, that urges the Xaverian to a consistent quest for the higher. The motto of the young Xaverian is the old maxim : " Perfectumnihilest, aliqiriddumrestatagentiim" (Nothing is Perfect as long as anything remains to be done).	□7XX— Å« ”©^(ÖÊHtþ2%Ô §€ ¸`XðYß¾>À"òÌúío‰oû#„| • ² ?7Qú}“a€ë¡w□ ˜â³´T4™˜^^ ‰N.&³7QÅ|2~I¶®%(9 × ~®ÆjaÇBŽ¢='Qˆã B¾ }°WÖ;Ž³±ˆ•ÿ©Øœ«v- m1ÿkãÙú— †R%nµXôš‹ÇUOÔ • ×ç!]Ø÷ osã5¦ ((ÂYÎ0£w±Ê¬š›2äGÆ/YÜÊ ¸<Ù ¥ó»J_Sy□´[;tî6Ò. -

	þ™IDteGÓ®ÈÝÖP† Ögte • Ä • HUú- éXàrŽSYs«*§¿Y2¢WRzí2kù /¡_}¦É□ §zŸ)ii¶ì#rû|K.HoiÍUl Aìâ’2pkØ_Wñö“-ÁÉ0Ä- 17}ÖÕ8V^Û¥bÑOd'ÛÓD¼ O0ˆ¹ ÍÅ- 0ÇÆÛ¥¿ö.ÚŸðP»Œ,•\în™Ô ƒ4□ rç”[Ë¥ý„Æ›qf p¤Cý_y□³:UšO1

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 cipher text.

7. 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 works smoothly. 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 the set of strings where there is only difference in one character there also the encrypted texts are coming totally different. The present method is free from any kind of brute force attack or known plain text attack. The present MWFES ver-I method may be applied to encrypt any short message, password, confidential key. 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. The work has already completed i.e. version-II where the author(s) have introduced the feedback in random order instead of serial way to make the system further complicated. 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: AsokeNath, SaimaGhosh, MeheboobAlamMallik:“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: DriptoChatterjee, JoyshreeNath, SoumitraMondal, SuvadeepDasgupta and AsokeNath,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, DriptoChatterjee, JoyshreeNath, SuvadeepDasgupta and AsokeNath : 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: NeerajKhanna, JoelJames,JoyshreeNath, SayantanChakraborty, AmlanChakrabarti and AsokeNath : 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, DriptoChatterjee, JoyshreeNath, Sankar Das, ShalabhAgarwal and AsokeNath, 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, JoyshreeNath, Megholova Mukherjee, NehaChaudhury and AsokeNath: 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, JoyshreeNath, ShayanDey and AsokeNath, 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, NavajitMaitra, JoyshreeNath,ShalabhAgarwal and AsokeNath, 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, AdbvancedCaeser Cipher Algorithm, Bit Rotation and reversal Method: SJA Algorithm., International Journal of Modern Education and Computer Science, SomdipDey, JoyshreeNath, AsokeNath,(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, Somdipdey, JoyshreeNath, AsokeNath, 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, NavajitMaitra, JoyshreeNath, ShalabhAgarwal and AsokeNath, 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, NeerajKhanna, DriptoChatterjee, JoyshreeNath and AsokeNath, International Journal of Computer Applications(IJCA)(0975-8887) USA Volume 52-No.2.,Aug, Page.41-46(2012).

[14] Bit LevelGeneralized Modified Vernam Cipher Method with Feedback : Prabal Banerjee, AsokeNath, 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, AsokeNath, Proceedings of International conference Worldcomp 2013 held at Las Vegas, July 2013.

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

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 achievements, etc.

Purnendu Mukherjee is a pursuing his Masters in Computer Science at St. Xavier’s College (Autonomous), Kolkata. He is actively doing research in Cryptography as well as Artificial Intelligence.