IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 13, Issue 1 (Jul. - Aug. 2013), PP 138-146
www.iosrjournals.org

A Block Cipher Based Cryptosystem through Modified Forward Backward Overlapped Modulo Arithmetic Technique (MFBOMAT)

Debajyoti Guha[1], Rajdeep Chakraborty[2], Abhirup Sinha[3]
[1]*Dept. of IT, Siliguri Institute of Technology,Darjeeling-734009, West Bengal, India.*
[2]*Dept of CSE, Netaji Subhash Engineering College, Garia , Kolkata-700152, West Bengal, India.*
[3]*Tata Consultancy Services, New Town, Rajarhat, Kolkata-700156,West Bengal, India.*

Abstract: *In this paper, a new Cryptosystem based on block cipher has been proposed where the encryption is done through Modified Forward Backward Overlapped Modulo Arithmetic Technique (MFBOMAT). The original message is considered as a stream of bits, which is then divided into a number of blocks, each containing n bits, where n is any one of 2, 4, 8, 16, 32, 64, 128, 256. The first and last blocks are then added where the modulus of addition is 2^n. The result replaces the last block (say N^{th} block), first block remaining unchanged (Forward mode). In the next attempt the second and the N^{th} block (the changed block) are added and the result replaces the second block(Backward mode).Again the second (the changed block) and the $(N-1)^{th}$ block are added and the result replaces the $(N-1)^{th}$ block (Forward mode).The modulo addition has been implemented in a very simple manner where the carry out of the MSB is discarded to get the result. The technique is applied in a cascaded manner by varying the block size from 2 to 256. The whole technique has been implemented by using a modulo subtraction technique for decryption.*

Keywords: *FBOMAT, Symmetric block cipher, Cryptosystem*

I. Introduction

Lack of security may exist when a volume of data is transferred from its source to the destination if no measure is taken for its security. For one reason or the other, most of the data being transmitted must be kept secret from others [2]. A very important reason to encode data or messages is to keep them secret. From e-mail to cellular communication, from secured web access to digital cash, cryptography [3] is an essential part of today's information systems. It can prevent fraud in electronic commerce and assure the validity of financial transactions. It can prove one's identity and protect one's anonymity. These electronic commerce schemes may fall fraud through forgery, misrepresentation, denial of service and cheating if we do not add security to these systems. In fact, computerization makes the risks even greater by allowing attacks that are impossible in non-automated systems. Only strong cryptography can protect against these attacks.

The Section I of this paper deals with the proposed scheme. A concept of key-generation is given in Section II. Results and comparisons are illustrated in Section III. Conclusions are drawn in Section IV, acknowledgements are shown in section V and Section VI lists the references.

1. The Modified Forward Backward Modulo Arithmetic Technique (MFBOMAT)

In the proposed scheme the source file is input as streams of binary bits. For our implementation we have taken the stream size to be 512 bits though the scheme may be implemented for larger stream sizes also.

The input stream, S, is first broken into a number of blocks, each containing n bits ($n=2^k$, k=1,2,3,......,8) so that S = $B_1B_2B_3$.......B_m where m=512/n. Starting from the MSB, the blocks are paired as (B_1,B_m), (B_2,B_m), (B_2,B_{m-1}),(B_3,B_{m-1}) and so on. So there is a common member in any two non-adjacent block-pairs, i.e. the block-pairs are overlapping and hence the name given to the technique The FBOMAT operation is applied to each pair of blocks. The process is repeated, each time increasing the block size till n=256.The proposed scheme has been implemented by using the reverse technique, i.e. modulo subtraction technique, for decryption. Section 1.1 explains the operation in detail.

1.1. The Algorithm for MFBOMAT

After breaking the input stream into blocks of 2 bits each and pairing the blocks as explained in Section 1, the following operations are performed starting from the most significant side:

Round 1: In each pair of blocks, the first member of the pair is added to the second member where the modulus of addition is 2^n for block size n. Therefore for 2-bit blocks, the modulus of addition will be 4. This round is repeated for a finite number of times and the number of iterations will form a part of the session key as discussed in Section 3**.**

Round 2: The same operation as in Round 1 is performed with block size 4.
In this fashion several rounds are completed till we reach ***Round 8*** where the block size is 256 and we get the encrypted bit-stream. The operations of the non adjacent block-pairs increases the complexity of the algorithm resulting in the enhancement of security.

During decryption, the reverse operation, i.e. modulo subtraction, is performed instead of modulo addition, starting from the blocks $B_m/2$ and $((B_m)/2)+1$ and then $((B_m)/2)$ and $((B_m)/2)+2$ and then $((B_m)/2)-1$ and $((B_m)/2)+2$ .The process continues until all the remaining blocks are decrypted.
.

1.1. The Modulo Addition

An alternative method for modulo addition is proposed here to make the calculations simple. The need for computation of decimal equivalents of the blocks is avoided here since we will get large decimal integer values for large binary blocks. The method proposed here is just to discard the carry out of the MSB after the addition to get the result. For example, if we add 1101 and 1001 we get 10110. In terms of decimal values, 13+9=22. Since the modulus of addition is 16 (2^4) in this case, the result of addition should be 6 (22-16=6). Discarding the carry from 10110 is equivalent to subtracting 10000 (i.e. 16 in decimal). So the result will be 0110, which is equivalent to 6 in decimal. The same is applicable to any block size.

1.2. Example of the Scheme

Although the proposed scheme is applied to a 512-bit input stream, for the sake of brevity, consider a stream of 32 bits, say S = 1101001100011011 each round is performed only once to make the process simple for understanding.

1.2.1 The Encryption Scheme

Round 1: Block size = 2, number of blocks = 8

Output

11	11	01	01	00	10	01	10
B1	B2	B3	B4	B5	B6	B7	B8

(B4, B6) mod4, Change B4

Input

11	11	01	01	00	10	01	10
B1	B2	B3	B4	B5	B6	B7	B8

Output

11	11	01	01	01	10	01	10
B1	B2	B3	B4	B5	B6	B7	B8

(B4, B5) mod4, Change B5

Round 2: Block size = 4, number of blocks = 4

Input

1111	0101	0110	0110
B1	B2	B3	B4

Output:

1111	0101	0110	0101
B1	B2	B3	B4

(B1, B4) mod16, Change B4

Input

1111	0101	0110	0101
B1	B2	B3	B4

Output

1111	1010	0110	0101
B1	B2	B3	B4

(B2, B4) mod16, Change B2

Input

1111	1010	0110	0101
B1	B2	B3	B4

Output

1111	1010	0000	0101
B1	B2	B3	B4

Input

(B2, B3) mod16, Change B3

Round 3: Block size = 8, number of blocks = 2

Input

11111010	00000101
B1	B2

Output

11111010	11111111
B1	B2

(B1, B2) mod 256, Change B2

Since we have considered only a 16-bit stream we cannot proceed further. The output from Round 3, say S', is the encrypted stream, i.e. S' =.1111101011111111.For decryption the opposite method i.e. modular subtraction is used to get back the original bit stream in S.

1.2.2 The Decryption Scheme

For decryption the opposite method i.e. modular subtraction is used to get back the original bit stream in S.
Round 1:Block size=8, number of blocks =2

Input

11111010	11111111
B1	B2

Output

11111010	00000101
B1	B2

(B1, B2) mod 256, Change B2

Round 2:Block size=4, number of blocks=4
Input

1111	1010	0000	0101
B1	B2	B3	B4

Output

1111	1010	0110	0101
B1	B2	B3	B4

(B2, B3) mod16, Change B3

Input

1111	1010	0110	0101
B1	B2	B3	B4

Output

1111	0101	0110	0101
B1	B2	B3	B4

(B2, B4) mod16, Change B2

Input

1111	0101	0110	0101
B1	B2	B3	B4

Output

1111	0101	0110	0110
B1	B2	B3	B4

(B1, B4) mod4, Change B4

Round 3:Block size=2, number of blocks =8

Input

11	11	01	01	01	10	01	10
B1	B2	B3	B4	B5	B6	B7	B8

Output

11	11	01	01	00	10	01	10
B1	B2	B3	B4	B5	B6	B7	B8

(B4, B5) mod4, Change B5

Input

11	11	01	01	00	10	01	10
B1	B2	B3	B4	B5	B6	B7	B8

Output

11	11	01	11	00	10	01	10
B1	B2	B3	B4	B5	B6	B7	B8

(B4, B6) mod4, Change B4

Input

11	11	01	11	00	10	01	10
B1	B2	B3	B4	B5	B6	B7	B8

11	11	01	11	00	01	01	10
B1	B2	B3	B4	B5	B6	B7	B8

(B3, B6) mod4, Change B6

Input

11	11	01	11	00	01	01	10
B1	B2	B3	B4	B5	B6	B7	B8

Output

11	11	00	11	00	01	01	10
B1	B2	B3	B4	B5	B6	B7	B8

(B3, B7) mod4, Change B3

Input

11	11	00	11	00	01	01	10
B1	B2	B3	B4	B5	B6	B7	B8

Output

11	11	00	11	00	01	10	10
B1	B2	B3	B4	B5	B6	B7	B8

(B2, B7) mod4, Change B7

Input

11	11	00	11	00	01	10	10
B1	B2	B3	B4	B5	B6	B7	B8

Output

11	01	00	11	00	01	10	10
B1	B2	B3	B4	B5	B6	B7	B8

(B2, B8) mod4, Change B2

Input

11	01	00	11	00	01	10	10
B1	B2	B3	B4	B5	B6	B7	B8

Output

11	01	00	11	00	01	10	11
B1	B2	B3	B4	B5	B6	B7	B8

(B1, B8) mod4, Change B8

The decrypted bit stream :S”=1101001100011011.So S=S”.

II. Key Generation

In the proposed scheme, eight rounds have been considered, each for 2, 4, 8, 16, 32, 64, 128, and 256 block size. As mentioned in Section 2.1, each round is repeated for a finite number of times and the number of iterations will form a part of the encryption-key. Although the key may be formed in many ways, for the sake of brevity it is proposed to represent the number of iterations in each round by a 16-bit binary string. The binary strings are then concatenated to form a 128-bit key for a particular key. Example in Section 3.1 illustrates the key generation process.

2.1. Example of Key Generation

Consider a particular session where the source file is encrypted using iterations for block sizes 2, 4, 8, 16, 32, 64, 128, and 256 bits, respectively. Table 1 shows the corresponding binary value for the number of iterations in

Table 1 : Representation of no. of iterations in each round by bits.

Round No.	**Block Size**	**No. of Iterations**	
		Decimal	**Binary**
1	2	74	0000000001001010
2	4	680	0000001010101000
3	8	4278	0001000010110110
4	16	44428	1010110110001100
5	32	44443	1010110110011011
6	64	48878	1011111011101110
7	128	49870	1100001011001110
8	256	50020	1100001101100100

Each round. The binary strings are concatenated together to form the 128-bit binary string : 00000000010010100000001010101000000010000101101101010110110001100101011011001101110110111111011 10111011000010110011101100001101100100.. This 128-bit binary string will be the encryption-key for this particular session. During decryption, the same key is taken to iterate each round of modulo-subtraction for the specified number of times.

III. Results and Comparisons

The variation of frequencies of all the 256 ASCII characters between the source file and the encrypted file are given in this section. The evenly distribution of character frequency over the 0-255 region of the encrypted file

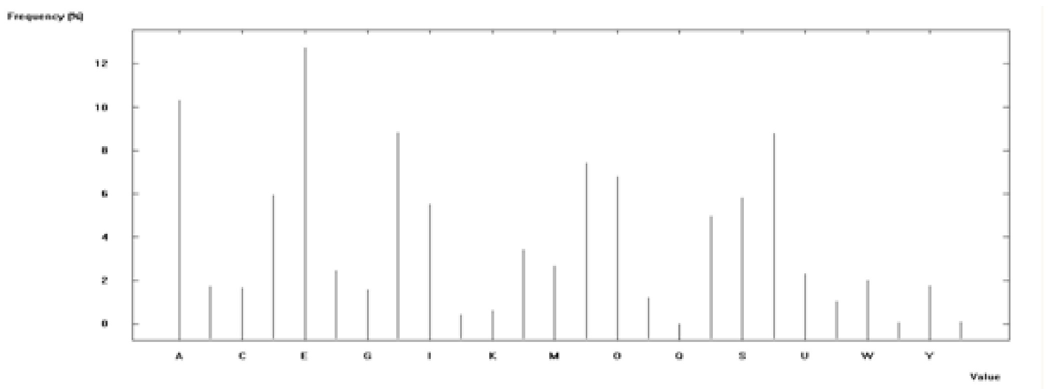


Fig. 1 : Frequency Distribution of ASCII characters in the source files.

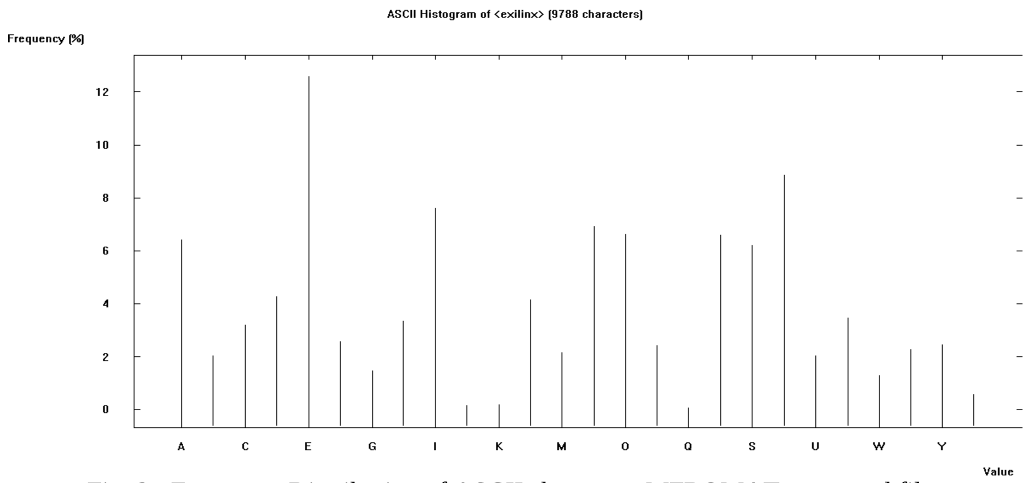


Fig. 2 : Frequency Distribution of ASCII characters MFBOMAT encrypted files

Against the source file ensures better security provided by the proposed algorithm and it also shows the heterogeneity between the two files. The frequency distribution graph is drawn according to the percentage of occurrence of a particular character, not the total number of occurrence.

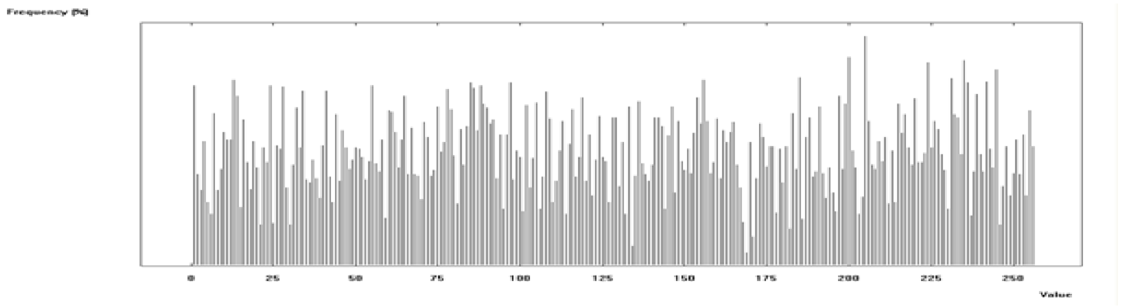


Fig. 3 : Frequency Distribution of ASCII characters in the RSA encrypted file.

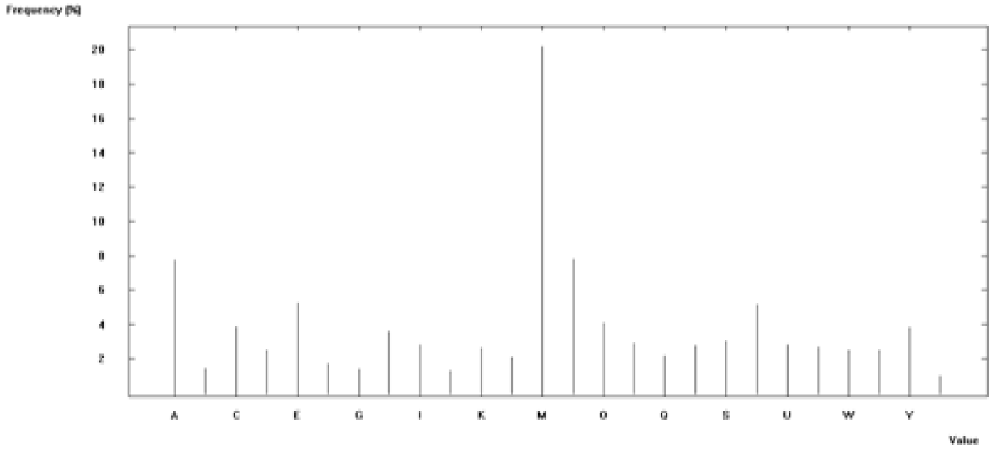


Fig. 4 : Frequency Distribution of ASCII characters in the FBOMAT encrypted file.

Although ten different text files were encrypted and decrypted using both RSA and MFBOMAT, only one such file is considered here for analyzing the results. Figs. 1, 2,3and 4 illustrate the frequencies of occurrence of all the 256 ASCII characters in the source file, encrypted file with MFBOMAT, and encrypted file with RSA and FBOMAT. A close observation will reveal that the characters in the encrypted file using MFBOMAT are fairly well distributed throughout the character space. Hence the MFBOMAT scheme may be comparable with RSA and FBOMAT. Another way to analyze the scheme is to test the homogeneity of the source and the encrypted file. The Chi-Square test has been performed for this purpose. Table 2 and fig 5 shows the source file name, size and the corresponding Chi-Square values (using MFBOMAT, RSA, FBOMAT) for ten different files. Barring some exceptions we see that the Chi-Square value increases with the increase in file size. Further, the high values prove that Chi-Square is highly significant at 1% level of significance.

Table 2 : Test for homogeneity using Chi-Square method.

File Size	FILE NAME	CHIVALUE MFBOMAT	CHIVALUE FBOMAT	CHIVALUE RSA
17	test.exe	41142	36237	65293
21	xilinx.txt	515344	521449	280724
28	Hills.jpg	18657	11273	14495
101	unified.txt	3563896	2951266	1459936
104	Winter.jpg	7698	8472	13140
107	LPAN.doc	75407	71047	91297
125	RPPT07.exe	107983	104619	258734
202	copy.jpg	237854	194465	186974
203	genesys.txt	5922387	6051580	3343845
215	walrithm.exe	1412765	1109781	2297243

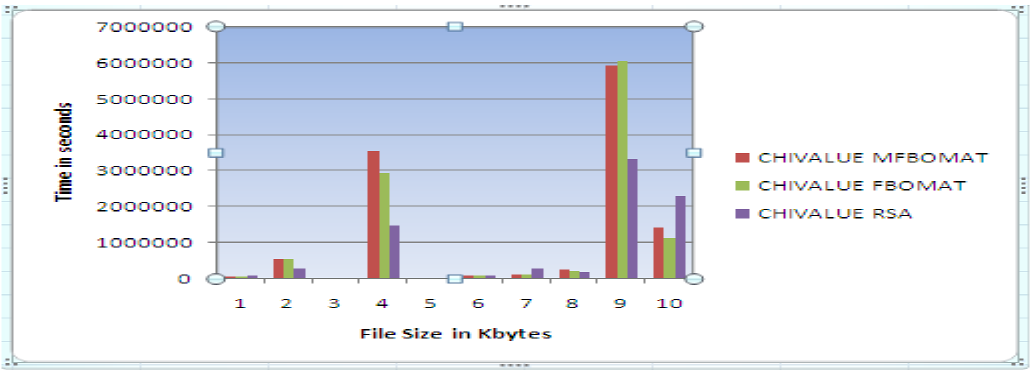


Fig. 5 : Graph showing Chi-Square values for MFBOMAT, FBOMAT and RSA

Hence the source and the corresponding encrypted files are considered to be heterogeneous. Another way to analyze the scheme is to analysis the encryption and decryption time.

Table 3: indicates encryption time for MFBOMAT, FBOMAT and RSA

File Size	MFBOMAT Encryption	FBOMAT Encryption	RSA Encryption
17	0.04	0.05	0.04
21	0.04	0.05	0.06
28	0.06	0.06	0.07
101	0.26	0.28	0.28
104	0.27	0.27	0.26
107	0.17	0.17	0.25
125	0.31	0.33	0.31
202	0.46	0.49	0.53
203	0.44	0.44	0.54
215	0.42	0.44	0.57

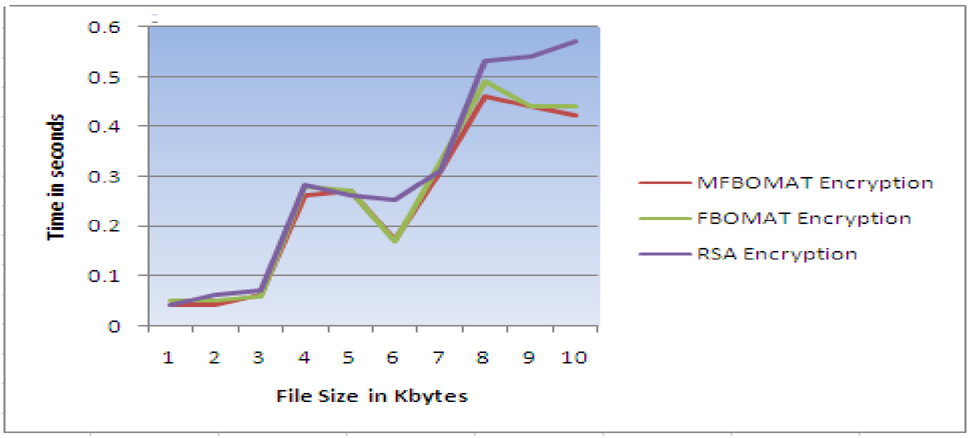
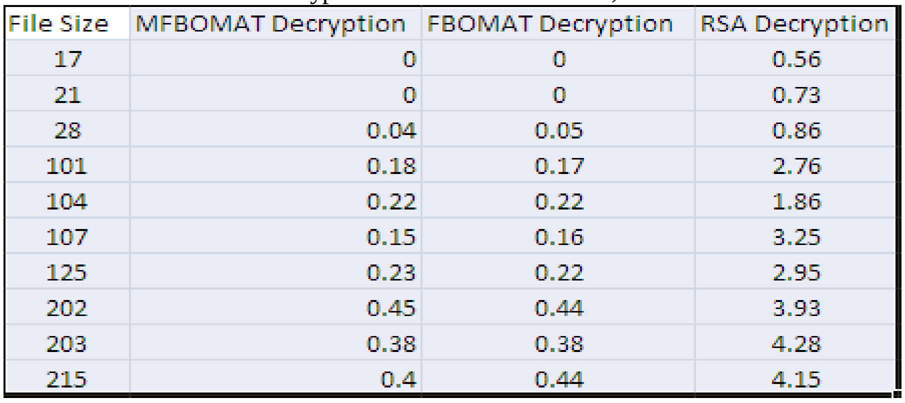


Fig 6: shows graphically Time Complexity Analysis among MFBOMAT, FBOMAT & RSA for Encryption.

Table 4: indicates decryption time for MFBOMAT, FBOMAT and RSA



File Size	MFBOMAT Decryption	FBOMAT Decryption	RSA Decryption
17	0	0	0.56
21	0	0	0.73
28	0.04	0.05	0.86
101	0.18	0.17	2.76
104	0.22	0.22	1.86
107	0.15	0.16	3.25
125	0.23	0.22	2.95
202	0.45	0.44	3.93
203	0.38	0.38	4.28
215	0.4	0.44	4.15

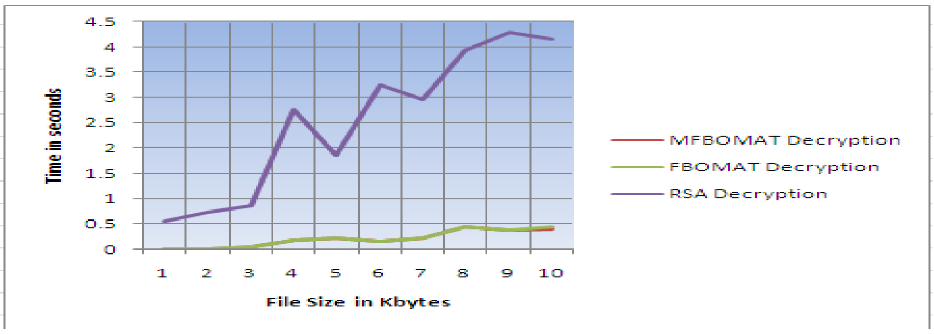


Fig 7: shows graphically Time Complexity Analysis among MFBOMAT, FBOMAT and RSA for Decryption

It can be seen that the time taken to encrypt a file using MFBOMAT is very little compared to that using RSA and FB OMAT.

IV. Conclusion

The technique proposed takes little time to encode and decode though the block length is high. The encoded string will not generate any overhead bits. The block length may further increased beyond 256 bits, which may enhance the security. Selecting the block pairs in random order, rather than taking those in consecutive order may enhance security. The proposed scheme may be applicable to embedded systems.

V. Acknowledgement

The authors express their deep sense of gratitude to the Department of Information Technology, Siliguri Institute of Technology,West Bengal University of Technology. The authors also express their deep sense of gratitude to the Department of Computer Science & Engineering, Netaji Subhash Engineering college,West Bengal University of Technology.

VI . Reference

Journal Paper:

[1] Rajdeep Chakraborty, Debajyoti Guha and J. K. Mandal, "A Block Cipher Based Cryptosystem Through Forward Backward Overlapped Modulo Arithmetic Technique (FBOMAT)", published *in International Journal of Engineering & Science Research Journal (IJESR)*, ISSN 2277 2685, accepted & published in *Volume 2 – Issue 5 (May 2012)* ,Article number 7,pp-349 – 360. Emai rajdeep_chak@indiatimes.com, guha_debajyoti@yahoo.com, jkmandal@sancharnet.in,

Books:

[2] W. Stallings, Cryptography and Network Security: Principles and Practices, Prentice Hall, Upper Saddle River, New Jersey, USA, Third Edition, 2003.

[3] Atul Kahate, Cryptography and Network Security, TMH, India,2nd Ed,2009

[4] Behroz Forouzan, Cryptography and Network Security, TMH, India, 4th Ed,2010

Proceedings Paper:

[5] Mandal, J. K., Sinha, S., Chakraborty, R., " A Microprocessor-based BlockCipher through Overlapped Modulo Arithmetic Technique (OMAT)",*Proceedings of 12th International Conference of IEEE on Advanced Computing and Communications* - ADCOM-2004, December 15-18,Ahmedabad, India, pp. 276 - 280, 2004.