

AN ODD-EVEN BLOCK CIPHER BASED CRYPTOSYSTEM THROUGH MODULO ARITHMETIC TECHNIQUE (OEMAT)

Debajyoti Guha¹, Alok Basu²

¹Assistant Professor, Dept of IT, ²Assistant Professor, Dept of CSE, Siliguri Institute of Technology, West Bengal, India
 guha_debajyoti@yahoo.com, basualok@rediffmail.com

Abstract

In this paper, a new Cryptosystem based on block cipher has been proposed where the encryption is done through **Odd Even Modulo Arithmetic Technique (OEMAT)**. 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 the penultimate blocks are then added where the modulus of addition is 2^n . The result replaces the penultimate block (say $(N-1)^{th}$ block), first block remaining unchanged. In the next attempt the second and the ultimate blocks (say N^{th} block) are added and the result replaces the N^{th} block. This process continues until all the blocks are executed. 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: MFBOMAT, FBOMAT, Symmetric block cipher, Cryptosystem

1. 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 [4]. 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 [5] 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 2 of this paper deals with the proposed scheme. A concept of key-generation is given in Section 3. Results and comparisons are illustrated in Section 3. Conclusions are drawn in Section 4, Acknowledgements are given in section 5 and Section 6 lists the references.

2. THE ODD EVEN MODULO ARITHMETIC TECHNIQUE (OEOMAT)

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 ($n=2^k$, $k=1,2,3,\dots,8$) so that $S = B_1B_2B_3\dots B_m$ where $m=512/n$. Starting from the MSB, the blocks are paired as (B_1, B_{m-1}) , (B_2, B_m) , (B_3, B_{m-2}) , (B_4, B_{m-3}) and so on. So block pairing is done between non adjacent odd numbered blocks and even numbered blocks separately and hence the name given to the technique. The OEMAT 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.

2.1. The Algorithm for OEOMAT

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 3** where the block size is 256 and we get the encrypted bit-stream. The operation 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, between the odd and even block pairs until all the remaining blocks are decrypted.

2.2. 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.

2.3 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.

2.3.1 The Encryption Scheme

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

Input

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

Output

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

Operation: (B1, B7) mod4 Change B7;
 (B2, B8) mod4 Change B8;
 (B3, B5) mod4 Change B5;
 (B4, B6) mod4 Change B6;

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

Input

1101	0011	0000	0100
B1	B2	B3	B4

Output:

1101	0011	1101	0111
B1	B2	B3	B4

Operation: (B1, B3) mod16, Change B3;
 (B2, B4) mod16, Change B4;

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

Input

11010011	11010111
B1	B2

Output

11010011	10101010
B1	B2

Operation: (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. $S' = 1101001110101010$. For decryption the opposite method i.e. modular subtraction is used to get back the original bit stream in S.

2.3.2 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

11010011	10101010
B1	B2

Output

11010011	11010111
B1	B2

Operation: (B1, B2) mod 256, Change B2

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

Input

1101	0011	1101	0111
B1	B2	B3	B4

Output

1101	0011	0000	0100
B1	B2	B3	B4

Operation: (B1, B3) mod16, Change B3;
 (B2, B4) mod16, Change B4;

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

Input

11	01	00	11	00	00	01	00
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

Operation: (B1, B7) mod4 ChangeB7;
 (B2, B8) mod4 ChangeB8;
 (B3, B5) mod4 ChangeB5;
 (B4, B6) mod4 ChangeB6;

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

3. 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.

3.1. 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 each round. The binary strings are concatenated together to form the 128-bit binary string: 0000000001001010000000101010100000010000101101101011011000110010101101100110111011110111011101101100010110011100001101100100. 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.

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

3.2. 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

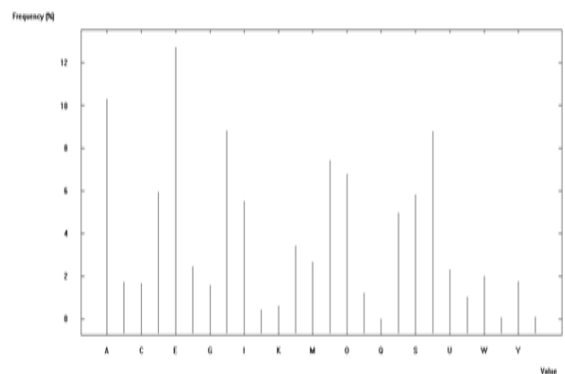


Fig. 1: Frequency Distribution of ASCII characters in the source file.

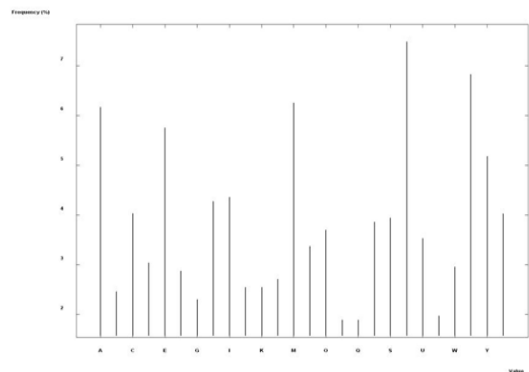


Fig.2: Frequency Distribution of ASCII characters in the OEMAT encrypted file.

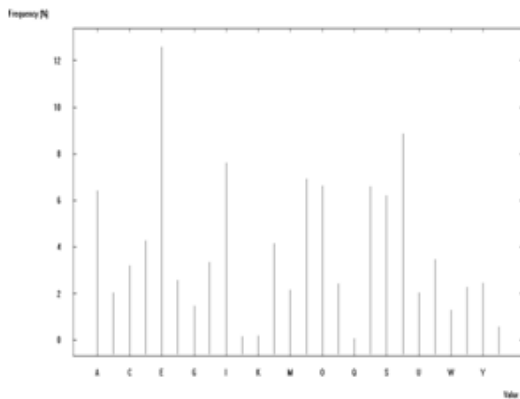


Fig.3: Frequency Distribution of ASCII characters MFBOMAT encrypted file

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.

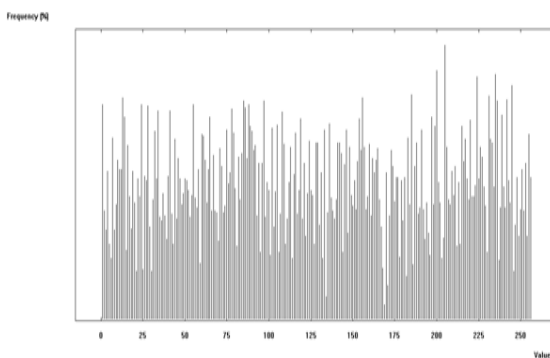


Fig. 4: Frequency Distribution of ASCII characters in the RSA encrypted file.

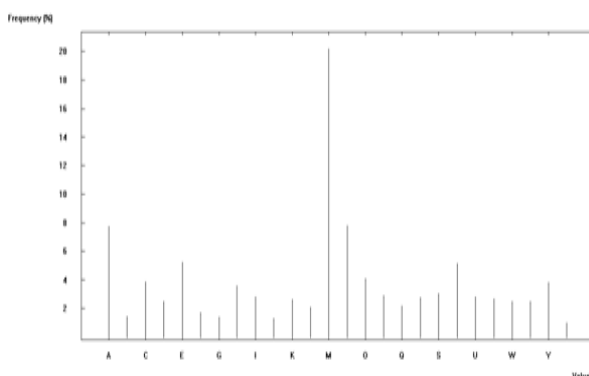


Fig.5: Frequency Distribution of ASCII characters in the FBOMAT encrypted file.

Although ten different text files were encrypted and decrypted using both RSA and OEMAT, only one such file is considered here for analyzing the results. Figs. 1,2,3 and 4 illustrate the frequencies of occurrence of all the 256 ASCII characters in the source file, encrypted file with OEMAT and encrypted file with MFBOMAT, RSA and FBOMAT. A close observation will reveal that the characters in the encrypted file using OEMAT are fairly well distributed throughout the character space. Hence the OEMAT scheme may be comparable with MFBOMAT, 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 OEMAT, 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	CHIVALUE OEMAT	CHIVALUE MFBOMAT	CHIVALUE FBOMAT	CHIVALUE RSA
17	41356	41142	36237	65293
21	546547	515344	521449	280724
28	12456	18657	11273	14495
101	3223456	3563896	2951266	1459936
104	7343	7698	8472	13140
107	73487	75407	71047	91297
125	116758	107983	104619	258734
202	208745	237854	194465	186974
203	6186754	5922387	6051580	3343845
215	1108023	1412765	1109781	2297243

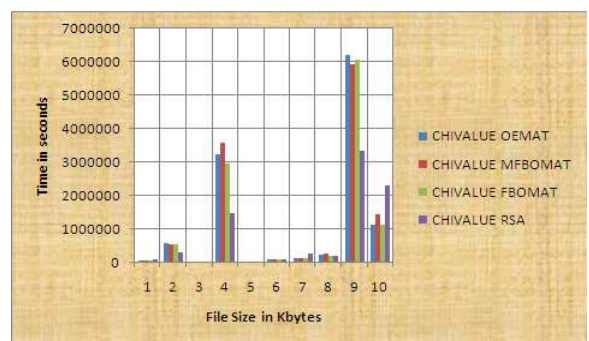


Chart 1: Graph showing Chi-Square values for MFBOMAT, FBOMAT and RSA

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 OEMAT, MFBOMAT, FBOMAT and RSA

File Size	OEMAT Encryption	MFBOMAT Encryption	FBOMAT Encryption	RSA Encryption
17	0.03	0.04	0.05	0.04
21	0.04	0.04	0.05	0.06
28	0.05	0.06	0.06	0.07
101	0.3	0.26	0.28	0.28
104	0.29	0.27	0.27	0.26
107	0.16	0.17	0.17	0.25
125	0.32	0.31	0.33	0.31
202	0.5	0.46	0.49	0.53
203	0.46	0.44	0.44	0.54
215	0.45	0.42	0.44	0.57

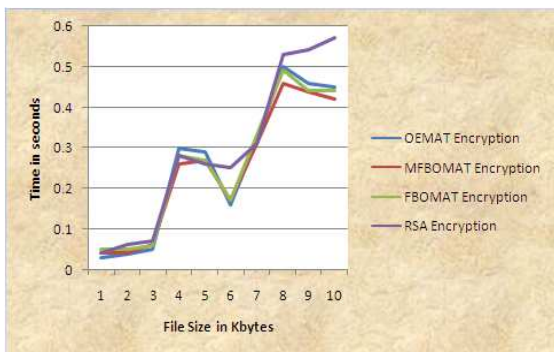


Chart 2: shows graphically Time Complexity Analysis among MFBOMAT, FBOMAT & RSA for Encryption.

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

File Size	OEMAT Decryption	MFBOMAT Decryption	FBOMAT Decryption	RSA Decryption
17	0	0	0	0.56
21	0	0	0	0.73
28	0.05	0.04	0.05	0.86
101	0.17	0.18	0.17	2.76
104	0.2	0.22	0.22	1.86
107	0.14	0.15	0.16	3.25
125	0.25	0.23	0.22	2.95
202	0.5	0.45	0.44	3.93
203	0.37	0.38	0.38	4.28
215	0.4	0.4	0.44	4.15

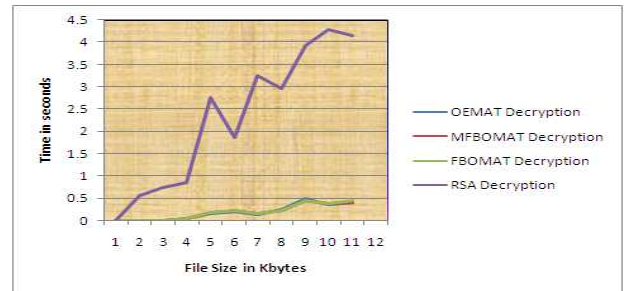


Chart 3: shows graphically Time Complexity analysis among OEMAT, MFBOMAT, FBOMAT and RSA for Decryption

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

CONCLUSIONS

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.

ACKNOWLEDGEMENTS

The authors express their deep sense of gratitude to the Department of Computer Science & Engineering and Department of Information Technology, Siliguri Institute of Technology.

REFERENCES

[1] Debajyoti Guha, Rajdeep Chakraborty, and Abhirup Sinha, "A Block Cipher Based Cryptosystem Through Modified Forward Backward Overlapped Modulo Arithmetic Technique (MFBOMAT)", published in International Organization of Scientific Research Journal of Computer Engineering (IOSR-JCE), e-ISSN: 2278-0661, p- ISSN: 2278-8727 accepted and published in Volume 13-Issue 1 (Jul. - Aug. 2013) issue, Article number 22, PP- 138-146.
 Email:rajdeep_chak@indiatimes.com, guha_debajyoti@yahoo.com, sinha.abhirup@gmail.com.
 [2] 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.
 Email:rajdeep_chak@indiatimes.com, guha_debajyoti@yahoo.com, jkmandal@sancharnet.in,

[3] 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.

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

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

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

BIOGRAPHIES:



Debajyoti Guha has obtained B.Tech.in Information Technology and M.Tech.in Computer Science from West Bengal University of Technology. Presently he is working as an Assistant Professor in the dept. of IT in Siliguri Institute of Technology. His research interest is in Computer Networking, Cryptography and Network Security.



Alok Basu has started his career as a Mechanical Engineer, at Alloy Steels Plant, SAIL in 1983. He worked there in different sections of his field in various capacities. Later he got a scope to be trained in COBOL and to work with the in-house software development and maintenance team for more than seven years. He was mainly involved in the development of Production Information & Order Servicing System in COBOL on MV 15000 (AOS/VS-II) platform. He joined in Siliguri Institute of Technology after completion of M. Tech in Information Technology. He is serving this institute for more than 10 years and presently holding the chair of HOD.