A Universal Bit Level Block Encoding Technique Using Session Based Symmetric Key Cryptography to Enhance the Information Security

Manas Paul[1] and Jyotsna Kumar Mandal[2]

[1] Dept. of Comp. Application, JIS College of Engineering, Kalyani, West Bengal, India

manaspaul@rediffmail.com

[2]Dept. of C.S.E., Kalyani University, Kalyani, West Bengal, India

jkmandal@rediffmail.com

ABSTRACT

In this paper a session based symmetric key cryptographic algorithm has been proposed and it is termed as Matrix Based Bit Jumbling Technique (MBBJT). MBBJT consider the plain text (i.e. the input file) as a binary bit stream with finite number bits. This input bit stream is divided into manageable-sized blocks with different length. The bits of the each block fit diagonally upward starting from (n , n) cell in a right to left trajectory into a square matrix of suitable order n. Then the bits are taken from the square matrix column-wise from top to bottom to form the encrypted binary string and from this encrypted string cipher text is formed. Combination of the values of block length and the no. of blocks of a session generates the session key. For decryption the cipher text is considered as a stream of binary bits. After processing the session key information, this binary string is divided into blocks. The bits of the each block fit column-wise from top to bottom into a square matrix of order n. Now bits are taken diagonally upward starting from (n , n) cell in a right to left trajectory from the square matrix to form the decrypted binary string. Plain text is regenerated from this binary string. Comparison of MBBJT with existing and industrially accepted TDES and AES has been done.

KEYWORDS

Matrix Based Bit Jumbling Technique (MBBJT), Cryptography, Symmetric Key, Session Based Key, TDES, AES.

1. INTRODUCTION

Day by day our communication becomes faster and simpler for internet. Every computer is connected virtually to each other through internet. Securing electronic data is gradually becoming important with the increasing dependency on the data interchange by the internet. Hence network security is the most focused topic among the researchers [1, 2, 3, 4]. Various cryptographic algorithms are available but each of them has their own merits and demerits. As a result continuous research works are going on in this field of cryptography to enhance the network security.

DOI : 10.5121/ijait.2012.2203

Based on symmetric key cryptography a new technique has been proposed where the plain text is considered as a stream of binary bits. Bit positions are jumbled up to generate the cipher text. A session key is generated using plain text information. The plain text can be regenerated from the cipher text using the session key information.

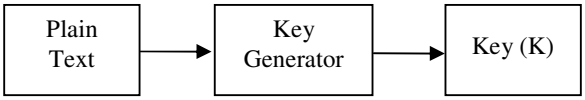
Section 2 of this paper contains the block diagram of the proposed scheme. Section 3 deals with the algorithms of encryption, decryption and key generation. Section 4 explains the proposed technique with an example. Section 5 shows the results and analysis on different files with different sizes and the comparison of the proposed MBBJT with TDES [5], AES [6]. Conclusions are drawn in the section 6.

2. THE SCHEME

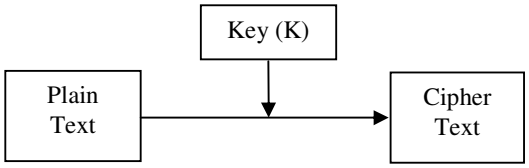
The MBBJT algorithm consists of three major components:

- Key Generation
- Encryption Mechanism
- Decryption Mechanism

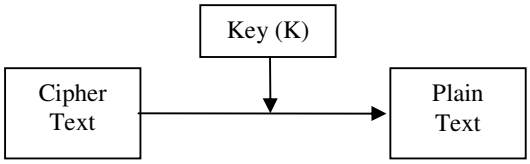
Key Generation:



Encryption Mechanism:



Decryption Mechanism:



3. PROPOSED ALGORITHM

3.1. Encryption Algorithm:

Step 1. The plain text i.e. the input file is considered as a binary bit stream of finite no. of bits.
Step 2. This binary stream breaks into manageable-sized blocks with different lengths like 4 / 16 / 64 / 144 / 256 / 400 / …….. [$(4n)^2$ for n = 1/2, 1, 2, 3, 4, 5, …….] as follows:
First n_1 no. of bits is considered as x_1 no. of blocks with block length y_1 where $n_1 = x_1 * y_1$. Next n_2 no. of bits is considered as x_2 no. of blocks with block length y_2 where $n_2 = x_2 * y_2$ and so on. Finally n_m no. of bits is considered as x_m no. of blocks with block length y_m (= 4) where $n_m = x_m * y_m$. So no padding is required.

Step 3. Square matrix of order √y is generated for each block of length y. The binary bits of the block from MSB to LSB fit diagonally upward starting from (√y , √y) cell in a right to left trajectory into this square matrix.
Step 4. From the square matrix bits are taken column-wise from top to bottom to generate the encrypted block of length y.
Step 5. The cipher text is formed after converting the encrypted binary string into characters.

3.2. Decryption Algorithm:

Step 1. The encrypted file i.e. the cipher text is considered as a stream of binary bits.
Step 2. After processing the session key information, this binary string breaks into manageable-sized blocks.
Step 3. Square matrix of order √y is generated for each block of length y. The binary bits of the block from MSB to LSB fit column-wise into the square matrix.
Step 4. The decrypted binary string is generated after taking the bits diagonally upward starting from (√y , √y) cell in a right to left trajectory from the square matrix.
Step 5. The plain text is reformed after converting the decrypted binary string into characters.

3.3. Generation of Session Key:

During encryption a session key is generated for one time use in a session of transmission to ensure much more security to MBBJT. This technique divides the input binary bit stream dynamically into 16 portions, each portion is divided again into x no. of blocks with block length y bits. The final (i.e. 16^{th}) portion is divided into x_{16} no. of block with block length 4 bits (i.e. y_{16} = 4). So no padding is required. Total length of the input binary string is

$$x_1 * y_1 + x_2 * y_2 + \ldots\ldots\ldots + x_{16} * y_{16}.$$

The values of x and y are generated dynamically. The session key contains the sixteen set of values of x and y respectively.

4. EXAMPLE

To illustrate the MBBJT, let us consider a two letter's word "Go". The ASCII values of "G" and "o" are 71 (01000111) and 111 (01101111) respectively. Corresponding binary bit representation of that word is "0100011101101111". Consider a block with length 16 bits as

0	1	0	0	0	1	1	1	0	1	1	0	1	1	1	1

Now these bits from MSB to LSB fit diagonally upward starting from (4 , 4) cell in a right to left trajectory into this square matrix of order 4 as follows:

1	1	1	1
1	0	0	1
1	1	0	0
1	0	1	0

The encrypted binary string is formed after taking the bits column-wise from top to bottom from above the square matrix as follows:

1	1	1	1	1	0	1	0	1	0	0	1	1	1	0	0

The equivalent decimal no. of two 8 bit binary numbers 11111010 and 10011100 are 250 and 156 respectively. 250 and 156 are the ASCII values of the characters ú (Latin small letter u with acute) and œ (Latin small ligature oe) respectively. So the word **Go** is encrypted as **úœ**.

For decryption, exactly reverse steps of the above are followed.

5. RESULTS AND ANALYSIS

In this section the comparative study between Triple-DES (168bits), AES (128bits) and MBBJT has done on 20 files of 8 different types with file sizes varying from 330 bytes to 62657918 bytes (59.7 MB). Analysis includes comparison of encryption time, decryption time, Character frequencies, Chi-square values, Avalanche and Strict Avalanche effects, Bit Independence. All implementation has been done using JAVA.

5.1. ANALYSIS OF ENCRYPTION & DECRYPTION TIME

Table I & Table II shows the encryption time and decryption time for Triple-DES (168bits), AES (128bits) and proposed MBBJT against the different files. Proposed MBBJT takes very less time to encrypt/decrypt than Triple-DES and little bit more time than AES. Fig. 1(a) and Fig. 1(b) show the graphical representation of encryption time and decryption time against file size in logarithmic scale.

TABLE I
File size v/s encryption time(for Triple-DES, AES and MBBJT algorithms)

Sl. No.	Source File Size (in bytes)	File type	Encryption Time (in seconds)		
			TDES	AES	MBBJT
1	330	Dll	0.001	0.001	0.003
2	528	Txt	0.001	0.001	0.005
3	96317	Txt	0.034	0.004	0.021
4	233071	Rar	0.082	0.011	0.054
5	354304	Exe	0.123	0.017	0.078
6	536387	Zip	0.186	0.023	0.131
7	657408	Doc	0.220	0.031	0.181
8	682496	Dll	0.248	0.031	0.198
9	860713	Pdf	0.289	0.038	0.215
10	988216	Exe	0.331	0.042	0.254
11	1395473	Txt	0.476	0.059	0.273
12	4472320	Doc	1.663	0.192	0.369
13	7820026	Avi	2.626	0.334	0.648
14	9227808	Zip	3.096	0.397	0.676
15	11580416	Dll	4.393	0.544	0.786
16	17486968	Exe	5.906	0.743	1.799
17	20951837	Rar	7.334	0.937	1.597
18	32683952	Pdf	10.971	1.350	1.992
19	44814336	Exe	15.091	1.914	2.934
20	62657918	Avi	21.133	2.689	5.288

TABL II
File size v/s decryption time (for Triple-DES, AES and MBBJT algorithms)

Sl. No.	Source File Size (in bytes)	File type	Decryption Time (in seconds)		
			TDES	AES	MBBJT
1	330	Dll	0.001	0.001	0.003
2	528	Txt	0.001	0.001	0.006
3	96317	Txt	0.035	0.008	0.024
4	233071	Rar	0.087	0.017	0.062
5	354304	Exe	0.128	0.025	0.089
6	536387	Zip	0.202	0.038	0.146
7	657408	Doc	0.235	0.045	0.198
8	682496	Dll	0.266	0.046	0.226
9	860713	Pdf	0.307	0.060	0.238
10	988216	Exe	0.356	0.070	0.277
11	1395473	Txt	0.530	0.098	0.298
12	4472320	Doc	1.663	0.349	0.407
13	7820026	Avi	2.832	0.557	0.712
14	9227808	Zip	3.377	0.656	0.535
15	11580416	Dll	4.652	0.868	0.866
16	17486968	Exe	6.289	1.220	1.974
17	20951837	Rar	8.052	1.431	1.768
18	32683952	Pdf	11.811	2.274	2.192
19	44814336	Exe	16.253	3.108	3.249
20	62657918	Avi	22.882	4.927	5.857

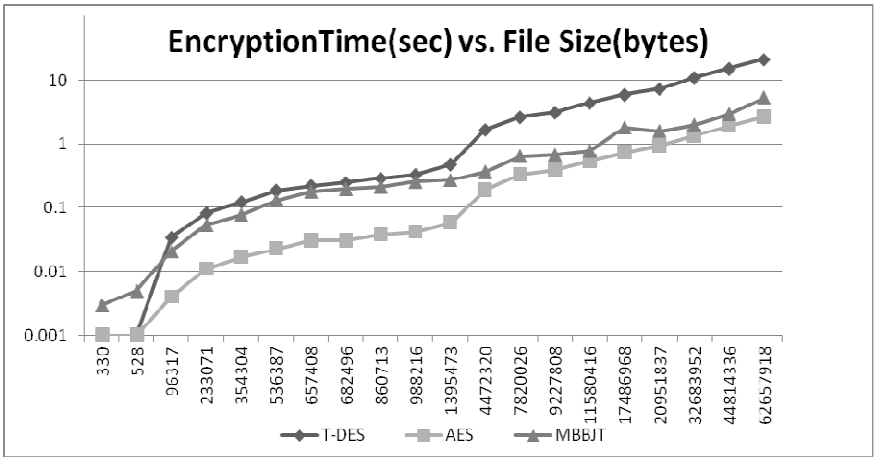


Fig. 1(a). Encryption Time (sec) vs. File Size (bytes) in logarithmic scale

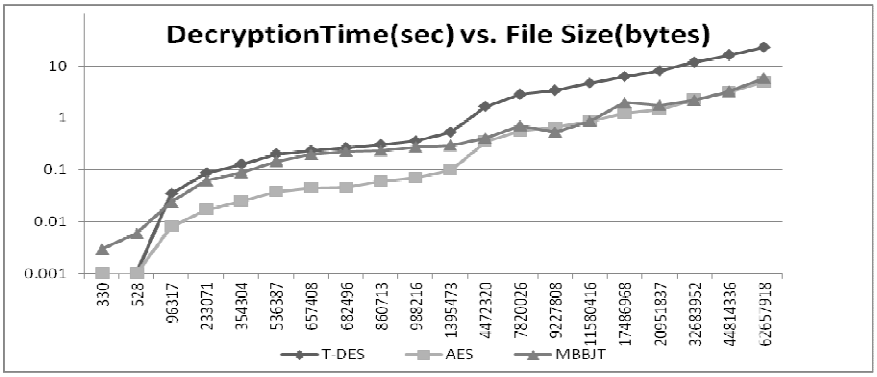


Fig. 1(b). Decryption Time (sec) vs. File Size (bytes) in logarithmic scale

5.2. ANALYSIS OF CHARACTER FREQUENCIES

Analysis of Character frequencies for text file has been performed for T-DES, AES and proposed MBBJT. Fig.2(a) shows the distribution of characters in the plain text. Fig.2(b), 2(c), 2(d) show the characters distribution in cipher text for T-DES, AES and proposed MBBJT. All three algorithms show a distributed spectrum of characters. From the above observation it may be conclude that the proposed MBBJT may obtain very good security.

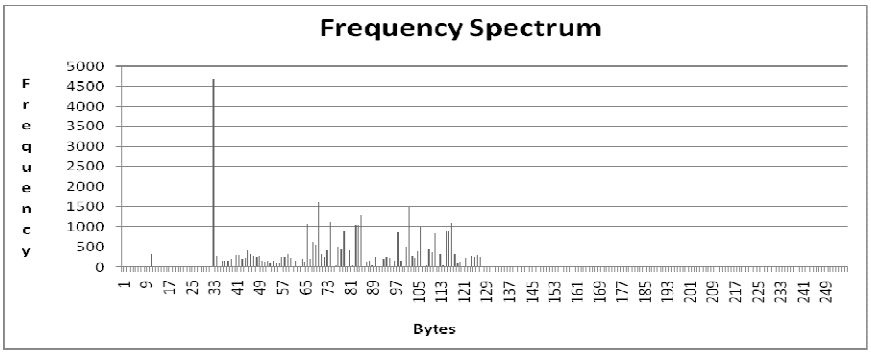


Fig. 2(a). Distribution of characters in source file

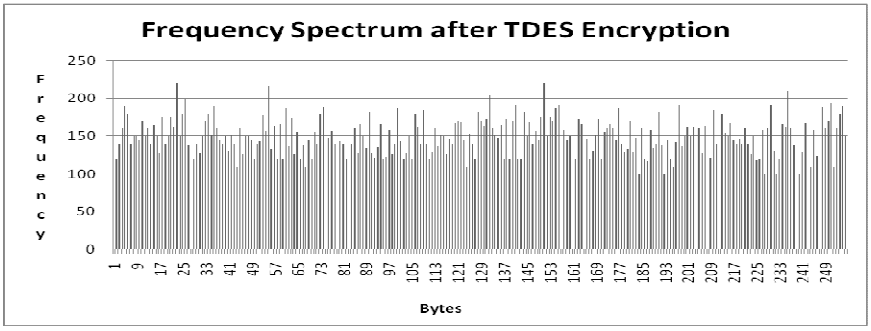


Fig. 2(b): Distribution of characters in TDES

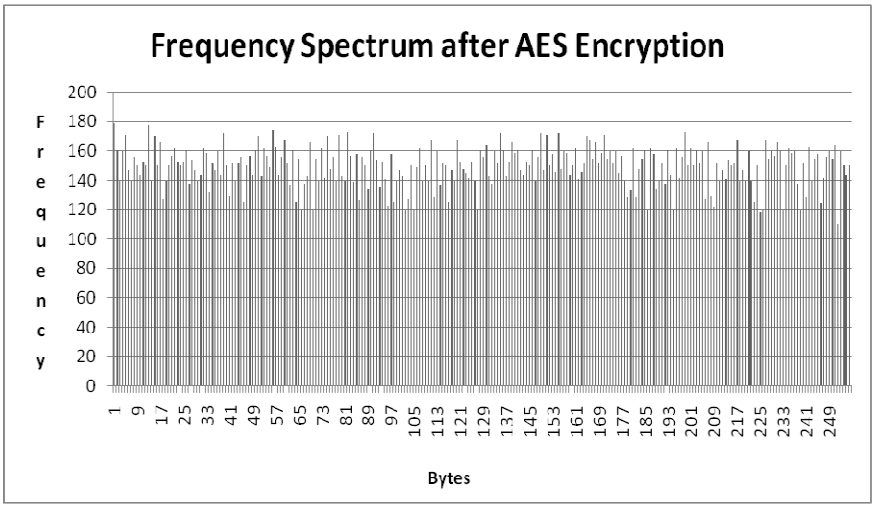


Fig. 2(c). Distribution of characters in AES

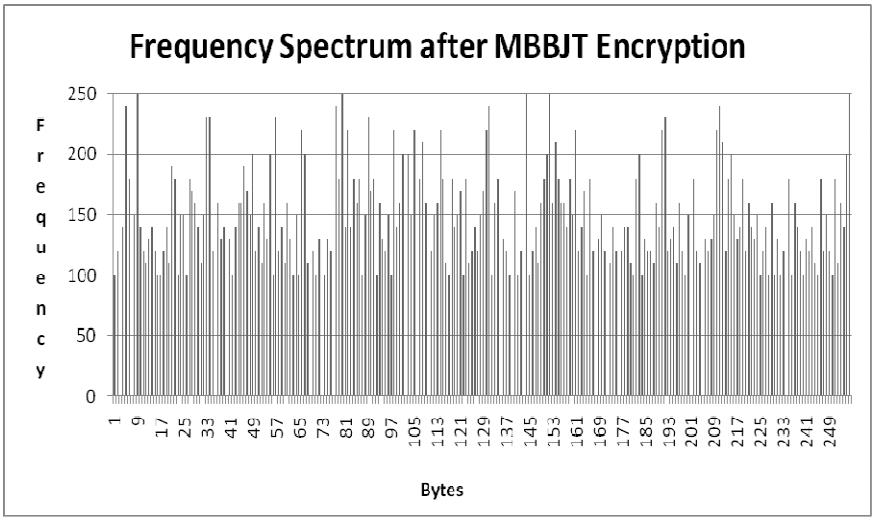


Fig. 2(d). Distribution of characters in MBBJT

5.3. TESTS FOR NON-HOMOGENEITY

The test for goodness of fit (Pearson χ^2) has been performed between the source files and the encrypted files. The large Chi-Square values (compared with tabulated values) may confirm the high degree of non-homogeneity between the source files and the encrypted files. Table III shows the Chi-Square values for Triple-DES (168bits), AES (128bits) and proposed MBBJT against the different files.

From Table III it may conclude that the Chi-Square values of MBBJT are at par with & sometimes better than that of T-DES and AES. Fig. 3 graphically represents the Chi-Square values on logarithmic scale for T-DES, AES & MBBJT.

Table III
Chi-Square values for Triple-DES, AES and MBBJT algorithms

Sl. No.	Source File Size (bytes)	File type	Chi-Square Values		
			TDES	AES	MBBJT
1	330	dll	922	959	874
2	528	txt	1889	1897	1938
3	96317	txt	23492528	23865067	20949798
4	233071	rar	997	915	963
5	354304	exe	353169	228027	214087
6	536387	zip	3279	3510	3308
7	657408	doc	90750	88706	87099
8	682496	dll	29296	28440	26536
9	860713	pdf	59797	60661	57051
10	988216	exe	240186	245090	256043
11	1395473	txt	5833237390	5545862604	5686269895
12	4472320	doc	102678	102581	99695
13	7820026	avi	1869638	1326136	1144840
14	9227808	zip	37593	37424	36682
15	11580416	dll	28811486	17081530	16699315
16	17486968	exe	8689664	8463203	8137730
17	20951837	rar	25615	24785	26267
18	32683952	pdf	13896909	13893011	15054022
19	44814336	exe	97756312	81405043	77350891
20	62657918	avi	3570872	3571648	3854424

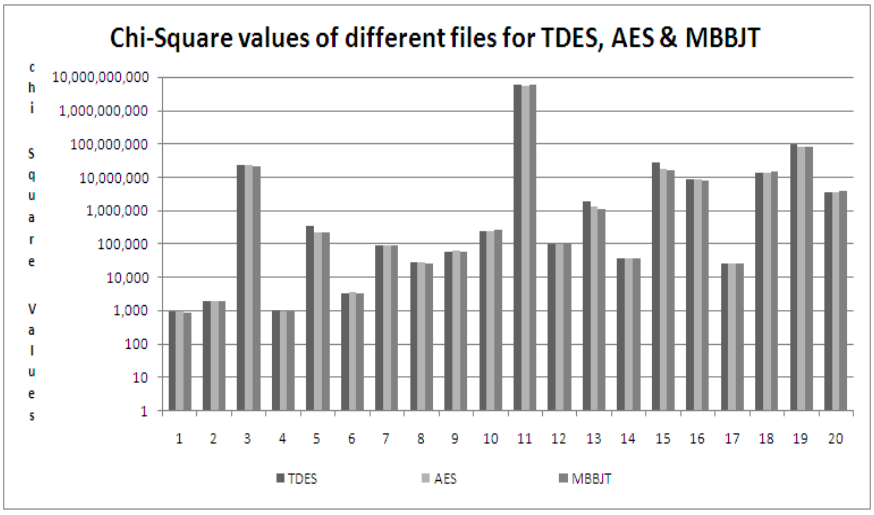


Fig.3 Chi-Square values for TDES, AES & MBBJT in logarithmic scale.

5.4. STUDIES ON AVALANCHE EFFECTS, STRICT AVALANCHE EFFECTS AND BIT INDEPENDENCE CRITERION

Avalanche & Strict Avalanche effects and Bit Independence criterion has been measured by statistical analysis of data. The bit changes among encrypted bytes for a single bit change in the original message sequence for the entire or a relative large number of bytes. The Standard Deviation from the expected values is calculated. The ratio of calculated standard deviation with expected value has been subtracted from 1.0 to get the Avalanche and Strict Avalanche effect on a 0.0 – 1.0 scale. The value closer to 1.0 indicates the better Avalanche & Strict Avalanche effects and the better Bit Independence criterion. Table IV, Table V & Table VI show the Avalanche effects, the Strict Avalanche effects & the Bit Independence criterion respectively. Fig.4(a), Fig.4(b) & Fig4(c) show the above graphically. In Fig.4(a) & Fig.4(b), the y-axis which represent the Avalanche effects & the Strict Avalanche effects respectively has been scaled from 0.97 – 1.0 for better visual interpretation.

Table IV
Avalanche effects for T-DES, AES and MBBJT algorithms

Sl. No.	Source File Size (in bytes)	File type	Avalanche achieved		
			TDES	AES	MBBJT
1	330	dll	0.99591	0.98904	0.98558
2	528	txt	0.99773	0.99852	0.98588
3	96317	txt	0.99996	0.99997	0.99643
4	233071	rar	0.99994	0.99997	0.99789
5	354304	exe	0.99996	0.99999	0.99774
6	536387	zip	0.99996	0.99994	0.99846
7	657408	doc	0.99996	0.99999	0.99795
8	682496	dll	0.99998	1.00000	0.99872
9	860713	pdf	0.99996	0.99997	0.99848
10	988216	exe	1.00000	0.99998	0.99873
11	1395473	txt	1.00000	1.00000	0.99896
12	4472320	doc	0.99999	0.99997	0.99824
13	7820026	avi	1.00000	0.99999	0.99863
14	9227808	zip	1.00000	1.00000	1.00000
15	11580416	dll	1.00000	0.99999	0.99898
16	17486968	exe	1.00000	0.99999	0.99964
17	20951837	rar	1.00000	1.00000	0.99967
18	32683952	pdf	0.99999	1.00000	0.99978
19	44814336	exe	0.99997	0.99997	0.99962
20	62657918	avi	0.99999	0.99999	0.99989

Table V
Strict Avalanche effect for T-DES, AES & MBBJT algorithms

Sl. No.	Source File Size (in bytes)	File type	Strict Avalanche achieved		
			TDES	AES	MBBJT
1	330	dll	0.98645	0.98505	0.97884
2	528	txt	0.99419	0.99311	0.97896
3	96317	txt	0.99992	0.99987	0.98434
4	233071	rar	0.99986	0.99985	0.99576
5	354304	exe	0.99991	0.99981	0.99798
6	536387	zip	0.99988	0.99985	0.99783
7	657408	doc	0.99989	0.99990	0.99797
8	682496	dll	0.99990	0.99985	0.99884
9	860713	pdf	0.99990	0.99993	0.99992
10	988216	exe	0.99995	0.99995	0.99878
11	1395473	txt	0.99990	0.99996	0.99882
12	4472320	doc	0.99998	0.99995	0.99898
13	7820026	avi	0.99996	0.99996	0.99886
14	9227808	zip	0.99997	0.99998	0.99967
15	11580416	dll	0.99992	0.99998	0.99896
16	17486968	exe	0.99996	0.99997	0.99959
17	20951837	rar	0.99998	0.99996	0.99962
18	32683952	pdf	0.99997	0.99998	0.99971
19	44814336	exe	0.99991	0.99990	0.99988
20	62657918	avi	0.99997	0.99998	0.99979

Table VI
Bit Independence criterion for T-DES, AES & MBBJT algorithms

Sl. No.	Source File Size (in bytes)	File type	Bit Independence achieved		
			TDES	AES	MBBJT
1	330	Dll	0.49180	0.47804	0.43945
2	528	Txt	0.22966	0.23056	0.21867
3	96317	Txt	0.41022	0.41167	0.42849
4	233071	Rar	0.99899	0.99887	0.98972
5	354304	Exe	0.92538	0.92414	0.93688
6	536387	Zip	0.99824	0.99753	0.99594
7	657408	Doc	0.98111	0.98030	0.98583
8	682496	Dll	0.99603	0.99560	0.98990
9	860713	Pdf	0.97073	0.96298	0.97957
10	988216	Exe	0.91480	0.91255	0.94074
11	1395473	Txt	0.25735	0.25464	0.24994
12	4472320	Doc	0.98881	0.98787	0.96787
13	7820026	Avi	0.98857	0.98595	0.98766
14	9227808	Zip	0.99807	0.99817	0.98985
15	11580416	Dll	0.86087	0.86303	0.86868
16	17486968	Exe	0.83078	0.85209	0.85963
17	20951837	Rar	0.99940	0.99937	0.98858
18	32683952	Pdf	0.95803	0.95850	0.96881
19	44814336	Exe	0.70104	0.70688	0.82894
20	62657918	Avi	0.99494	0.99451	0.99788

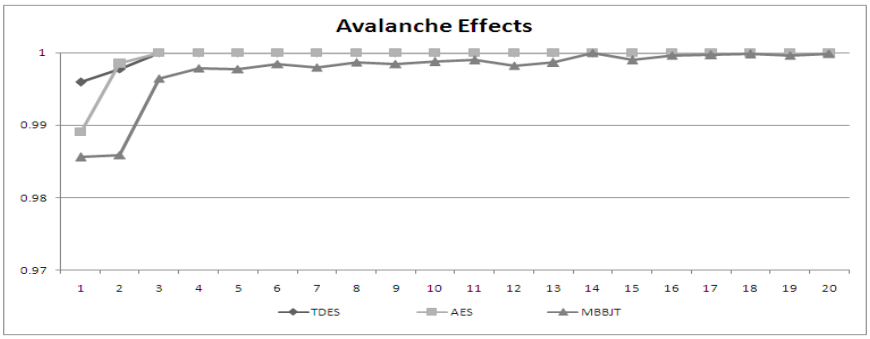


Fig.4(a) Comparison of Avalanche effect between T-DES, AES and MBBJT

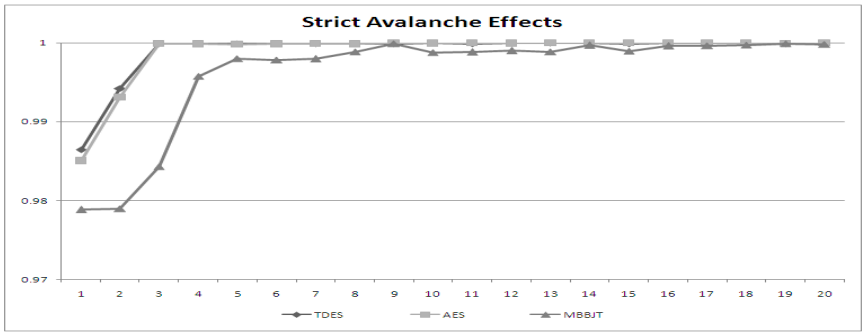


Fig4(b) Comparison of Strict Avalanche effect between TDES, AES and MBBJT

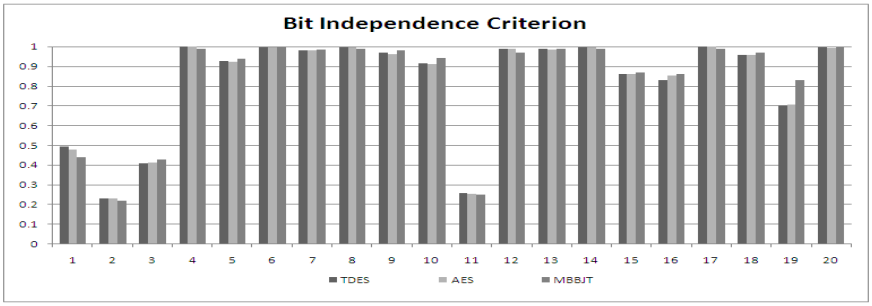


Fig.4(c) Comparison of Bit Independence criterion between TDES, AES and MBBJT

6. CONCLUSION

MBBJT, the proposed technique in this paper is simple, easy to understand and easy to implement. The key information varies from session to session for any particular file which may enhance the security features. Results and Analysis section indicates that the MBBJT is comparable with industry accepted standards T-DES and AES. The performance of MBBJT is significantly better than T-DES algorithm. For large files, MBBJT is at par with AES algorithm. Therefore the proposed technique is applicable to ensure high security in message transmission of any form and is suitable for any sort of file transfer.

REFERENCES

[1] J.K. Mandal, P.K. Jha, Encryption through Cascaded Arithmetic Operation on Pair of Bits and Key Rotation (CAOPBKR), National Conference of Recent Trends in Intelligent Computing (RTIC-06), Kalyani Government Engineering College, Kalyani, Nadia, India, 17-19 November 2006.

[2] M.Paul, J.K.Mandal, "A Permutative Cipher Technique (PCT) to Enhance the Security of Network Based Transmission", in Proceedings of 2nd National Conference on Computing for Nation Development, Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi, pp. 197-202,08th -09th February 2008

[3] S. Som, D. Mitra, J. Halder, Session Key Based Manipulated Iteration Encryption Technique (SKBMIET), International Conference on Advanced Computer Theory and Engineering (ICACTE 2008), Phuket, Thailand, 20-22 December 2008.

[4] S. Som, K. Bhattacharyya, R. Roy Guha, J. K. Mandal, Block Wise Bits Manipulations Technique (BBMT), International Conference on Advanced Computing, Tiruchirappalli, India, 6-8 August 2009.

[5] "Triple Data Encryption Standard" FIPS PUB 46-3 Federal Information Processing Standards Publication, Reaffirmed, 1999 October 25 U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology.

[6] "Advanced Encryption Standard", Federal Information Processing Standards Publication 197, November 26, 2001

Authors

Mr. Manas Paul received his Master degree in Physics from Calcutta University in 1998 and Master degree in Computer Application with distinction in 2003 from Visveswariah Technological University. Currently he is pursuing his PhD in Technology from Kalyani University. He is the Head and Assistant Professor in the Department of Computer Application, JISCE, West Bengal, India. His field of interest includes Cryptography and Network Security, Operation Research and Optimization Techniques, Distributed Data Base Management System, Computer Graphics.

Dr. JYOTSNA KUMAR MANDAL received his M.Tech. and PhD degree from Calcutta University. He is currently Professor of Computer Science & Engineering & Dean, Faculty of Engineering, Technology & Management, University of Kalyani, Nadia, West Bengal India. He is attached with several AICTE projects. He has 25 years Teaching & Research Experiences. His field of interest includes Coding Theory, Data and Network Security, Remote Sensing & GIS based Applications, Data Compression error corrections, Watermarking, Steganography and Document Authentication, Image Processing, Visual Cryptography.