A Bit Level Session Based Encryption Technique to Enhance Information Security

Manas Paul
Asst. Prof., Dept. of Computer Application, JIS College of Engineering, Kalyani, West Bengal, India,
e-mail:manaspaul@rediffmail.com

Jyotsna Kumar Mandal
Prof., Dept. of C.S.E., Kalyani University, Kalyani, West Bengal, India,
e-mail: jkmandal@rediffmail.com

Abstract— **In this paper, a session based symmetric key cryptographic system has been proposed and it is termed as Bit Shuffle Technique (BST). This proposed technique is very fast, suitable and secure for encryption of large files. BST consider the plain text (i.e. the input file) as binary string with finite no. of bits. The input binary string is broken down into manageable-sized blocks to fit row-wise from left to right into a square matrix of suitable order. Bits are taken diagonally upward from the square matrix to form the encrypted binary string and from this string cipher text is formed. Combination of values of block length and no. of blocks of a session generates the session key for BST. For decryption the cipher text is considered as binary string. Using the session key information, this binary string is broken down into manageable-sized blocks to fit diagonally upward from left to right into a square matrix of suitable order. Bits are taken row-wise from left to right from the square matrix to form the decrypted binary string and from this string plain text is formed. A comparison of BST with existing and industrially accepted TDES and AES has been done.**

Keywords- ***Bit Shuffle Technique (BST), Cryptography, Symmetric Key, Plain text, Cipher text, Session Based Key, TDES, AES.***

I. INTRODUCTION

Number of users who are using the internet in their daily life is increasing day by day. Communication through Internet becomes very popular because it is faster and easier. But it is important to secure our information from unauthorized users. Hence network security is the most focused domain for researchers and continuous research works are going on for the development of network security [1, 2, 3]. Various encryption techniques are available and all of them have their own merits and demerits.

In this paper a new cryptographic technique based on symmetric key cryptography has been proposed where the plain text is considered as a stream of binary bits. After shuffling the bit positions the cipher text is generated. A session key is generated during the encryption process. The plain text can be regenerated from the cipher text using the session key.

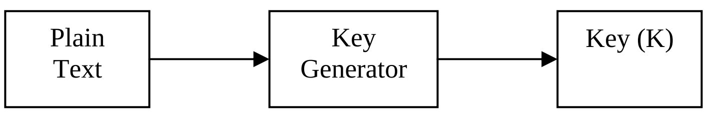
Section 2 of this paper contains the proposed scheme with block diagrams. Section 3 deals with the algorithms for encryption, decryption and key generation. Section 4 explains the proposed technique with an example. Section 5 shows the results and analysis on different files and the comparison of the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

II. THE SCHEME

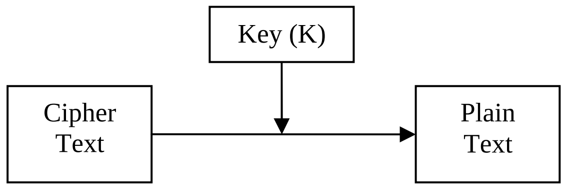
The BST algorithm consists of three major portions:

- Key Generation
- Encryption Mechanism
- Decryption Mechanism

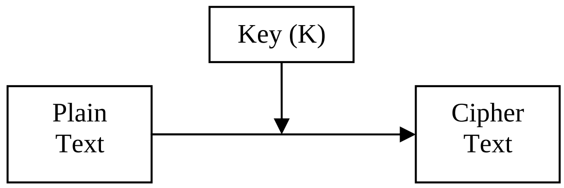
- *Key Generation:*



- *Encryption Mechanism:*



- *Decryption Mechanism:*



III. PROPOSED ALGORITHMS

Encryption Algorithm:

Step 1. The input file i.e. the plain text is considered as a stream of finite no. of binary bits.

Step 2. This binary string 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 where $n_m = x_m * y_m$ with $y_m = 4$. 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 row-wise from left to right into this square matrix.

Step 4. The encrypted binary string is generated after taking the bits diagonally upwards from left to right from that square matrix.

Step 5. The cipher text is formed after converting the encrypted binary string into characters.

Decryption Algorithm:

Step 1. The encrypted file i.e. the cipher text is considered as a binary stream.

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 diagonally upwards from left to right into this square matrix.
Step 4. The decrypted binary string is generated after taking the bits row-wise from left to right from the square matrix.

Step 5. The plain text is reformed after converting the decrypted binary string into characters.

Generation of Session Key:

During the encryption process a session key is generated for one time use in a session of transmission to ensure much more security to BST. This technique divides the input binary bit stream dynamically into 16 portions (say), 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 (in bits) is

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

The values of x and y are generated dynamically. The session key contains the sixteen set of values of x and y respectively. The length of the session key is remains same if the no. of portions is fixed and the length will vary if the no. of portions varies session to session.

IV. EXAMPLE

To illustrate the BST, 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 row-wise from left to right into this square matrix of order 4 as follows:

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

The encrypted binary string is formed after taking the bits diagonally upwards from left to right from above the square matrix as follows:

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

The equivalent decimal no. of two 8 bit binary numbers 00101011 and 10111101 are 43 and 189 respectively. 43 and 189 are ASCII values of the characters "+" and "µ" respectively. So the word "Go" is encrypted as "+µ". For decryption, exactly reverse steps of the above are followed.

V. RESULTS AND ANALYSIS

In this section the comparative study between Triple-DES(168bits), AES(128bits) and BST has been done on 20 files of 8 different file types with different 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 the computer language JAVA.

A. Analysis of Encryption and Decryption Time

Table I & Table II shows the encryption time and decryption time for Triple-DES (168bits), AES (128bits) and proposed BST against the different files. Proposed BST takes quite 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 for the above three algorithms.

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

Sl. No.	Source File Size (in bytes)	File type	Encryption Time (in seconds)		
			TDES	AES	BST
1	330	dll	0.001	0.001	0.004
2	528	txt	0.001	0.001	0.007
3	96317	txt	0.034	0.004	0.018
4	233071	rar	0.082	0.011	0.059
5	354304	exe	0.123	0.017	0.077
6	536387	zip	0.186	0.023	0.126
7	657408	doc	0.220	0.031	0.226
8	682496	dll	0.248	0.031	0.063
9	860713	pdf	0.289	0.038	0.108
10	988216	exe	0.331	0.042	0.147
11	1395473	txt	0.476	0.059	0.164
12	4472320	doc	1.663	0.192	0.362
13	7820026	avi	2.626	0.334	0.638
14	9227808	zip	3.096	0.397	0.478
15	11580416	dll	4.393	0.544	0.782
16	17486968	exe	5.906	0.743	1.790
17	20951837	rar	7.334	0.937	1.499
18	32683952	pdf	10.971	1.350	1.973
19	44814336	exe	15.091	1.914	2.825
20	62657918	avi	21.133	2.689	5.577

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

Sl. No.	Source File Size (in bytes)	File type	Decryption Time (in seconds)		
			TDES	AES	BST
1	330	Dll	0.001	0.001	0.002
2	528	Txt	0.001	0.001	0.007
3	96317	Txt	0.035	0.008	0.030
4	233071	Rar	0.087	0.017	0.062
5	354304	Exe	0.128	0.025	0.075
6	536387	Zip	0.202	0.038	0.063
7	657408	Doc	0.235	0.045	0.216
8	682496	Dll	0.266	0.046	0.143
9	860713	Pdf	0.307	0.060	0.097
10	988216	Exe	0.356	0.070	0.143
11	1395473	Txt	0.530	0.098	0.328
12	4472320	Doc	1.663	0.349	0.530
13	7820026	Avi	2.832	0.557	0.653
14	9227808	Zip	3.377	0.656	0.493
15	11580416	Dll	4.652	0.868	0.958
16	17486968	Exe	6.289	1.220	1.738
17	20951837	Rar	8.052	1.431	1.983
18	32683952	Pdf	11.811	2.274	3.643
19	44814336	Exe	16.253	3.108	3.213
20	62657918	Avi	22.882	4.927	5.830

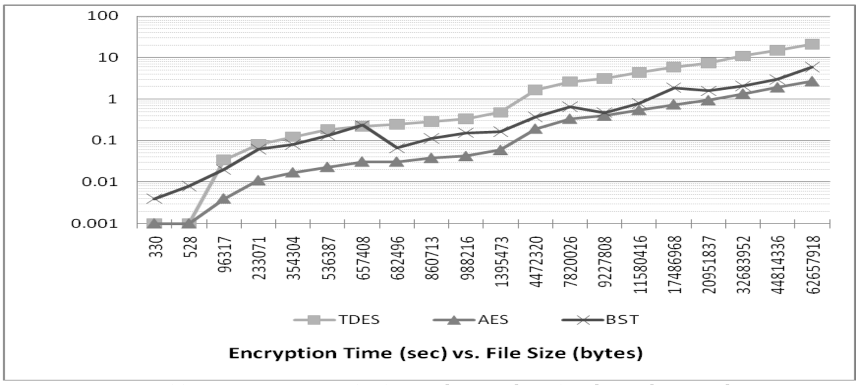


Fig. 1(a). Encryption Time (sec) vs. File Size (bytes) in logarithmic scale

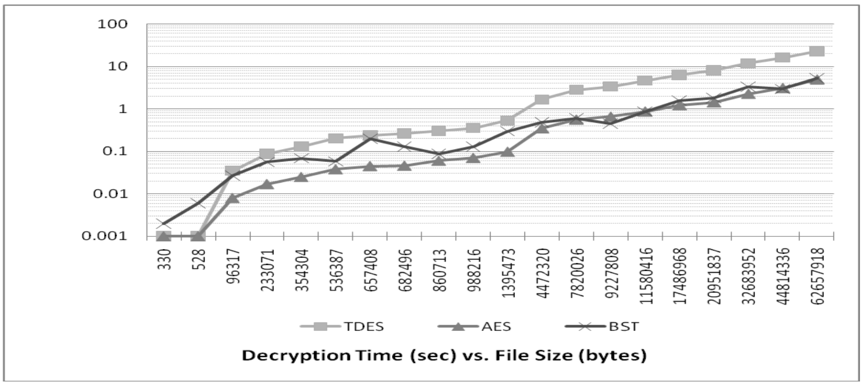


Fig. 1(b). Decryption Time (sec) vs. File Size (bytes) in logarithmic scale

B. Analysis of Character Frequencies

Analysis of Character frequencies for text file has been performed for T-DES, AES and proposed BST. 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 BST. All three algorithms show a distributed spectrum of characters. From the above observation it may be conclude that the proposed BST obtain very good security.

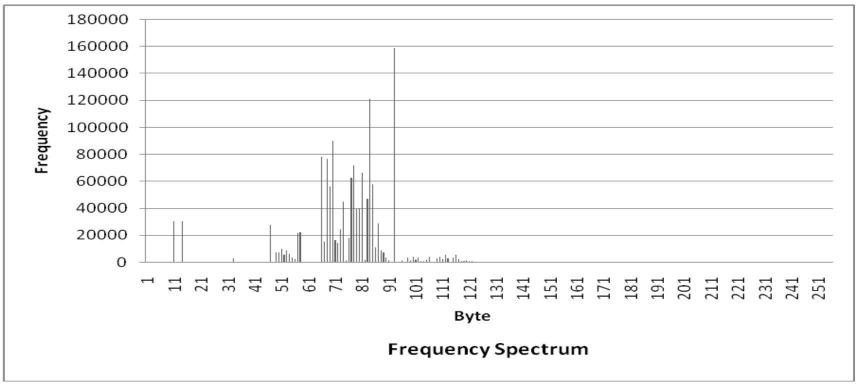


Fig. 2(a): Distribution of characters in source file

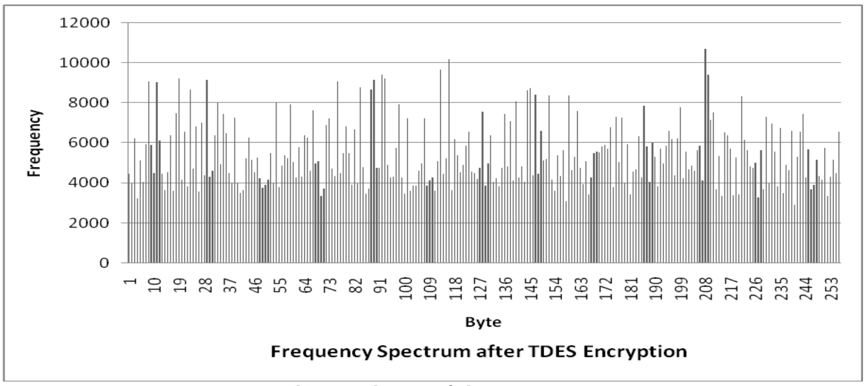


Fig. 2(b): Distribution of characters in TDES

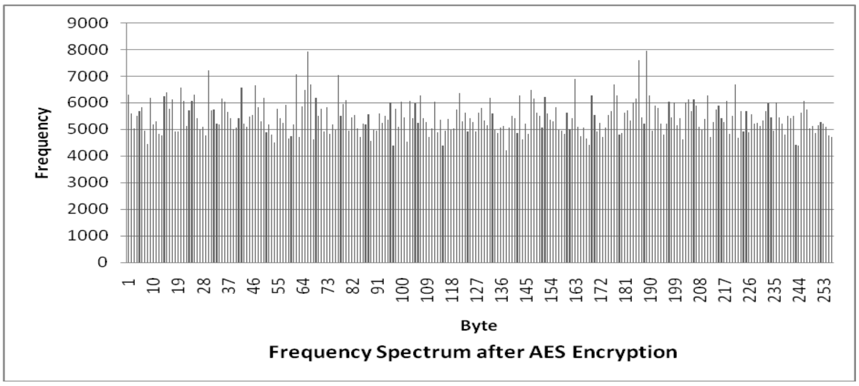


Fig. 2(c): Distribution of characters in AES

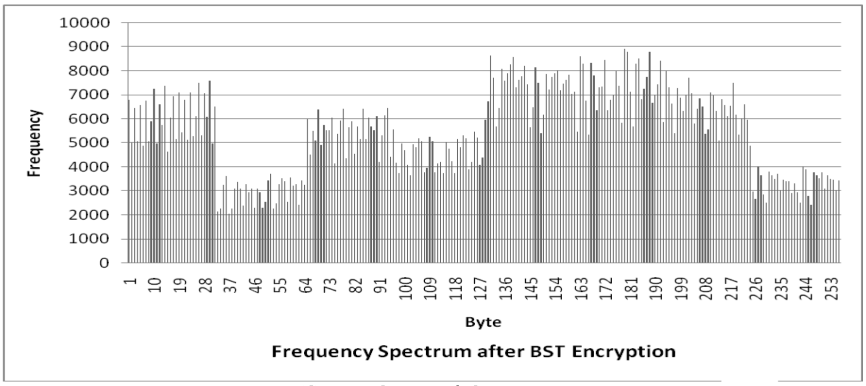


Fig. 2(d): Distribution of characters in BST

C. Tests for Non-homogeneity

The test for goodness of fit (Pearson χ^2) has been performed between the source files and the encrypted files to measure the degree no heterogeneity. 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 BST against the different files.

From Table III it may conclude that the Chi-Square values of BST are at par with T-DES and AES. Fig. 3 shows the graphical representation of Chi-Square values on logarithmic scale for T-DES, AES & BST.

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

Sl. No.	Source File Size (bytes)	File type	Chi-Square Values		
			TDES	AES	BST
1	330	dll	922	959	897
2	528	txt	1889	1897	1944
3	96317	txt	23492528	23865067	20234952
4	233071	rar	997	915	978
5	354304	exe	353169	228027	177261
6	536387	zip	3279	3510	3369
7	657408	doc	90750	88706	88250
8	682496	dll	29296	28440	26726
9	860713	pdf	59797	60661	56302
10	988216	exe	240186	245090	257941
11	1395473	txt	5833237390	5545862604	6791970526
12	4472320	doc	102678	102581	100173
13	7820026	avi	1869638	1326136	810169
14	9227808	zip	37593	37424	36829
15	11580416	dll	28811486	17081530	13800582
16	17486968	exe	8689664	8463203	8019008
17	20951837	rar	25615	24785	26570
18	32683952	pdf	13896909	13893011	15344561
19	44814336	exe	97756312	81405043	501345414
20	62657918	avi	3570872	3571648	4907916

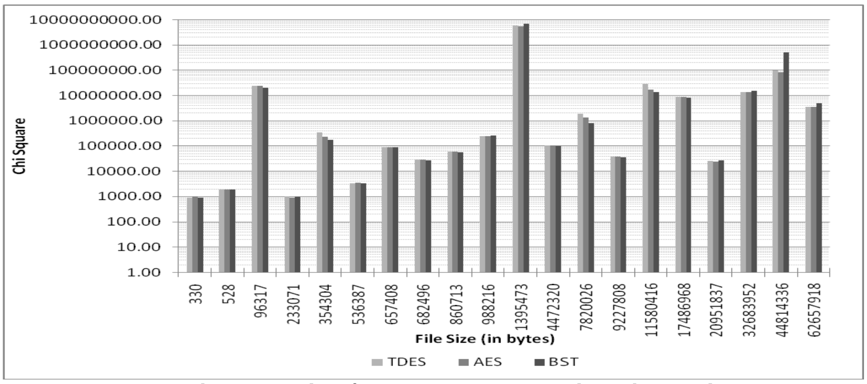


Fig.3 Chi-Square values for TDES, AES & BST in logarithmic scale.

D. Studies on Avalanche, Strict Avalanche Effects and Bit Independence Criterion:

Avalanche & Strict Avalanche effects and Bit Independence criterion has been measured using the 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 the calculated standard deviation with the expected value has been subtracted from 1.0 to get the Avalanche and Strict Avalanche effect in a 0.0 – 1.0 scale. The value closer to 1.0 indicates the better Avalanche & Strict Avalanche effect 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 0.9 – 1.0 for better visual interpretation.

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

Sl. No.	Source File Size (in bytes)	File type	Avalanche achieved		
			TDES	AES	BST
1	330	dll	0.99591	0.98904	0.96730
2	528	txt	0.99773	0.99852	0.97957
3	96317	txt	0.99996	0.99997	0.99281
4	233071	rar	0.99994	0.99997	0.99757
5	354304	exe	0.99996	0.99999	0.99399
6	536387	zip	0.99996	0.99994	0.99870
7	657408	doc	0.99996	0.99999	0.99637
8	682496	dll	0.99998	1.00000	0.99484
9	860713	pdf	0.99996	0.99997	0.99684
10	988216	exe	1.00000	0.99998	0.98983
11	1395473	txt	1.00000	1.00000	0.99651
12	4472320	doc	0.99999	0.99997	0.99298
13	7820026	avi	1.00000	0.99999	0.99754
14	9227808	zip	1.00000	1.00000	1.00000
15	11580416	dll	1.00000	0.99999	0.99859
16	17486968	exe	1.00000	0.99999	0.99812
17	20951837	rar	1.00000	1.00000	0.99862
18	32683952	pdf	0.99999	1.00000	0.99951
19	44814336	exe	0.99997	0.99997	0.99815
20	62657918	avi	0.99999	0.99999	0.99926

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

Sl. No.	Source File Size (in bytes)	File type	Strict Avalanche achieved		
			TDES	AES	BST
1	330	dll	0.98645	0.98505	0.89817
2	528	txt	0.99419	0.99311	0.97023
3	96317	txt	0.99992	0.99987	0.97918
4	233071	rar	0.99986	0.99985	0.99434
5	354304	exe	0.99991	0.99981	0.99070
6	536387	zip	0.99988	0.99985	0.99751
7	657408	doc	0.99989	0.99990	0.99319
8	682496	dll	0.99990	0.99985	0.98803
9	860713	pdf	0.99990	0.99993	0.98998
10	988216	exe	0.99995	0.99995	0.97454
11	1395473	txt	0.99990	0.99996	0.99321
12	4472320	doc	0.99998	0.99995	0.98517
13	7820026	avi	0.99996	0.99996	0.99365
14	9227808	zip	0.99997	0.99998	0.99980
15	11580416	dll	0.99992	0.99998	0.99887
16	17486968	exe	0.99996	0.99997	0.99924
17	20951837	rar	0.99998	0.99996	0.99861
18	32683952	pdf	0.99997	0.99998	0.99966
19	44814336	exe	0.99991	0.99990	0.99924
20	62657918	avi	0.99997	0.99998	0.99982

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

Sl. No.	Source File Size (in bytes)	File type	Bit Independence achieved		
			TDES	AES	BST
1	330	dll	0.49180	0.47804	0.39264
2	528	txt	0.22966	0.23056	0.21005
3	96317	txt	0.41022	0.41167	0.42947
4	233071	rar	0.99899	0.99887	0.98561
5	354304	exe	0.92538	0.92414	0.93652
6	536387	zip	0.99824	0.99753	0.99464
7	657408	doc	0.98111	0.98030	0.97474
8	682496	dll	0.99603	0.99560	0.96779
9	860713	pdf	0.97073	0.96298	0.96919
10	988216	exe	0.91480	0.91255	0.93115
11	1395473	txt	0.25735	0.25464	0.24670
12	4472320	doc	0.98881	0.98787	0.95581
13	7820026	avi	0.98857	0.98595	0.97007
14	9227808	zip	0.99807	0.99817	0.98021
15	11580416	dll	0.86087	0.86303	0.86221
16	17486968	exe	0.83078	0.85209	0.85677
17	20951837	rar	0.99940	0.99937	0.97225
18	32683952	pdf	0.95803	0.95850	0.95977
19	44814336	exe	0.70104	0.70688	0.82787
20	62657918	avi	0.99494	0.99451	0.99863

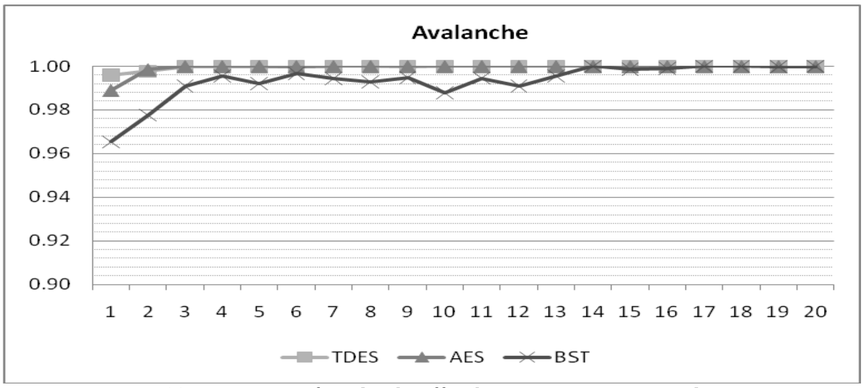


Fig.4(a) Comparison of Avalanche effect between T-DES, AES and BST

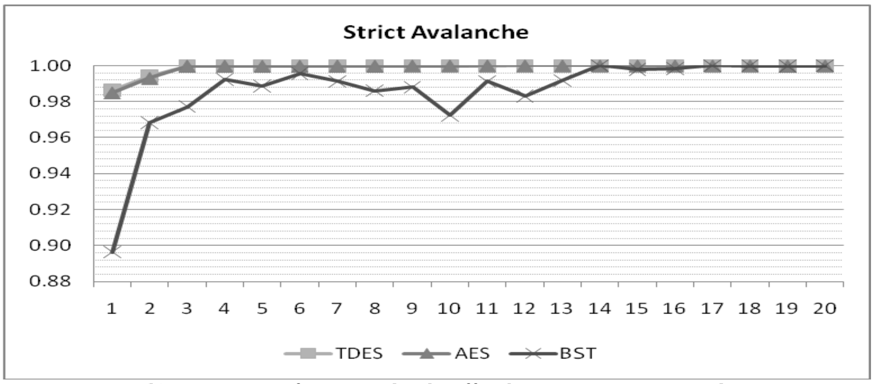


Fig4(b) Comparison of Strict Avalanche effect between TDES, AES and BST

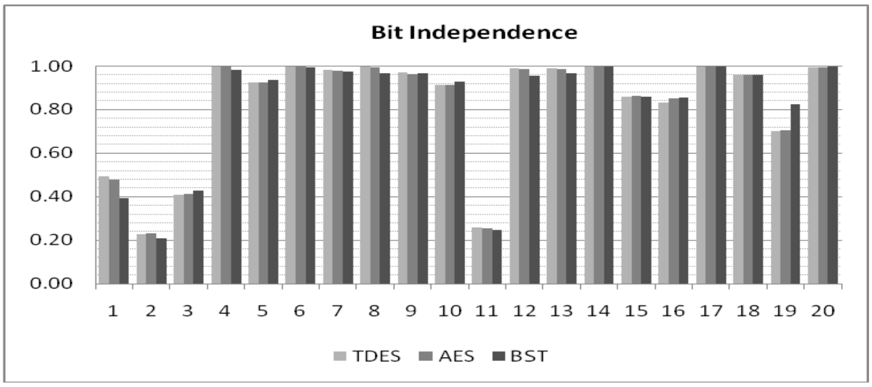


Fig.4(c) Comparison of Bit Independence criterion between TDES, AES and BST

VI. CONCLUSION

The proposed technique BST, presented in this paper is very simple to understand and easy to implement. The key length varies session to session for any particular file which certainly enhances the security features. Results and Analysis section indicates that the BST is definitely comparable with T-DES and AES. The performance of BST is significantly better than T-DES algorithm. For large files, BST is at par with AES algorithm. The proposed technique is applicable to ensure high security in transmission of any file of any size.

REFERENCES

[1] J.K. Mandal, S. Dutta, A Universal Bit-level Encryption Technique, *Seventh Vigyan Congress*,Jadavpur University, India, 28Feb to Ist March, 2000.

[2] 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, 17-19 Nov. 2006, India, pp212-220.

[3] J.K. Mandal, P.K. Jha, Encryption Through Cascaded Recursive Key Rotation and Arithmetic Operation (CRKRAO) of a Session Key, *14th International Conference on Advanced Computing and Communication*, NITK, Surathkal, Mangalore, 20-23 Dec., India, 2006.

[4] *"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.

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