A Universal Session Based Bit Level Symmetric Key Cryptographic Technique 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 technical paper a session based symmetric key cryptographic technique, termed as SBSKCT, has been proposed. This proposed technique is very secure and suitable for encryption of large files of any type. SBSKCT considers the plain text as a string with finite no. of binary bits. This input binary string is broken down into blocks of various sizes (of 2^k order where k = 3, 4, 5, ….). The encrypted binary string is formed by shifting the bit position of each block by a certain values for a certain number of times and from this string cipher text is formed. Combination of values of block length, no. of blocks and no. of iterations generates the session based key for SBSKCT. For decryption the cipher text is considered as binary string. Using the session key information, this binary string is broken down into blocks. The decrypted binary string is formed by shifting the bit position of each block by a certain values for a certain number of times and from this string plain text is reformed. A comparison of SBSKCT with existing and industrially accepted TDES and AES has been done.

.Keywords

SBSKCT, Cryptography, Symmetric Key, Plain text, Cipher text, Session Based Key, TDES, AES.

1. INTRODUCTION

During this time when the Internet provides essential communication between more than tens of millions of people and is being increasingly used as a tool for commerce, security becomes a tremendously important issue to deal with. It is essential to secure our information from eavesdroppers. Hence network security is very much focused topic for researchers [1, 2, 3, 4, 5, 6, 7, 8, 9]. Many algorithms are available on this domain but each of them has their own merits and demerits. As a result continuous research works are going on in this field of cryptography.

In this paper a new algorithm based on symmetric key cryptography has been proposed where the plain text is considered as a stream of binary bits. Bit positions are shuffled to generate the cipher text. During encryption process a session key is generated. The plain text can be regenerated from the cipher text using the session key.

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 SBSKCT with TDES [10], AES [11]. Conclusions are drawn in the section 6.

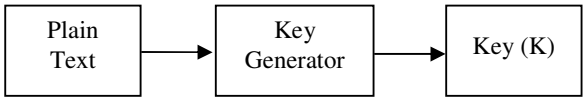
DOI : 10.5121/ijnsa.2012.4408

2. THE SCHEME

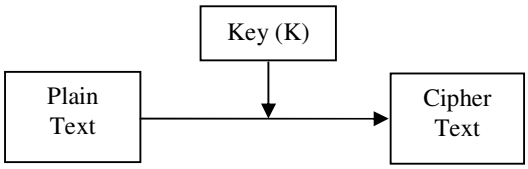
The SBSKCT algorithm consists of three major components:

- Key Generation
- Encryption Mechanism
- Decryption Mechanism

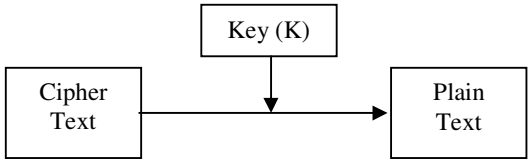
Key Generation:



Encryption Mechanism:



Decryption Mechanism:



3. PROPOSED ALGORITHM

3.1. Encryption Algorithm:

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

Step 2. This input binary string breaks into blocks with different lengths like 8 / 16 / 32 / 64 / 128 / 256 / 512 ……. (2^k order where k = 3, 4, 5, ….) as follows:

First n_1 no. of bits is considered as x_1 no. of blocks with block length y_1 bits 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 bits 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 bits where $n_m = x_m * y_m$ with $y_m = 8$. So no padding is required.

Step 3. For each block with length n, a unique number (ranging from 1 to 3*n) is generated for the position of each bit using the following function

$$f1(p) = p + n * [\{ n + p * (-1)^{\wedge} (n \% 3) \} \% 3] ;$$

where, n = block length under consideration
p = position of the p^{th} bit
(-1)^ (n % 3) means that (n% 3) is the power of (-1)

(n % 3) returns the remainder when n is divided by 3.

Step 4. The new position of each bit is generated to form the next intermediate block using the given function

$f2(q) = (q + 2) / 3$; where, q = generated unique no. using f1(p) for p^{th} bit

Step 5. The block of length n ($=2^k$) be regenerated after n/4 ($=2^{k-2}$) no. of iteration. Any of the intermediate blocks generated in this process may be used as encrypted string.

Step 6. The cipher text is formed after converting the encrypted binary string into characters.

3.2. Decryption Algorithm:

Step 1. For decryption process, the input file i.e. the cipher text is considered as a binary stream.

Step 2. After processing the session key information, this binary string is broken down into blocks of different length as similar as encryption process.

Step 3. Since a block of length n ($=2^k$) is regenerated after n/4 ($=2^{k-2}$) no. of iteration, so the process is symmetric in nature. If n1 no. of iteration is used for encryption then (n/4 – n1) no. of iteration is used for decryption.

Step 4. The plain text is reformed after converting the decrypted binary string into characters.

3.3. Generation of Session Key:

A session key is generated for one time use in a session of transmission to ensure much more security to SBSKCT. The input plain text which is treated as binary bit stream is divided 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 blocks with block length 8 bits (i.e. $y_{16} = 8$). So padding is not required. Total length of the input binary string can be written as

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

The values of x and y are generated dynamically. The value of n1, no. of iteration performed to encrypt the block with n bits, is also generated dynamically. The session key contains the sixteen set of values of x, y and n1 respectively.

4. EXAMPLE

To illustrate the SBSKCT, let us consider a two letter's word "At". The ASCII values of "A" and "t" are 65 (01000001) and 116 (01110100) respectively. Corresponding binary bit representation of that word is "0100000101110100". Consider a block with length 16 bits as

0	1	0	0	0	0	0	1	0	1	1	1	0	1	0	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

A unique no. is generated for each bit position from MSB to LSB using the function f1(p) and from this unique no. the new position of the p-th bit for next intermediate block is generated using the function f2(q).Table 4.1 shows the above for 16 bits block (i.e. n=16)

Table 4.1
Bit positions of each bit within a block

Original bit position	Generated unique no. using the function f1(p)	New Bit position for next intermediate block using the function f2(q)
01	01	01
02	34	12
03	19	07
04	04	02
05	37	13
06	22	08
07	07	03
08	40	14
09	25	09
10	10	04
11	43	15
12	28	10
13	13	05
14	46	16
15	31	11
16	16	06

The source block, intermediate blocks and the final block which is nothing but the source blocks are shown in the Figure 4.1.

0	1	0	0	0	0	0	1	0	1	1	1	0	1	0	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Source block with bit positions from MSB to LSB

0	0	0	1	0	0	0	0	0	1	0	1	0	1	1	1
1	4	7	10	13	16	3	6	9	12	15	2	5	8	11	14

First intermediate block with old (i.e. previous) bit position

0	0	0	1	0	0	0	0	0	1	0	1	0	1	1	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

First intermediate block with new (i.e. present) bit position

0	1	0	1	0	1	0	0	0	1	1	0	0	0	0	1
1	4	7	10	13	16	3	6	9	12	15	2	5	8	11	14

Second intermediate block with old (i.e. previous) bit position

0	1	0	1	0	1	0	0	0	1	1	0	0	0	0	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Second intermediate block with new (i.e. present) bit position

0	1	0	1	0	1	0	1	0	0	0	1	0	0	1	0
1	4	7	10	13	16	3	6	9	12	15	2	5	8	11	14

Third intermediate block with old (i.e. previous) bit position

0	1	0	1	0	1	0	1	0	0	0	1	0	0	1	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Third intermediate block with new (i.e. present) bit position

0	1	0	0	0	0	0	1	0	1	1	1	0	1	0	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Fourth intermediate block i.e. regenerated source block

Fig. 4.1 Formation of the cycle of the process

For 16 (= n) bit block, total no of iteration is 4 (= n/4) i.e. after 4th iteration source block is regenerated. If n1 no. of iteration is used to encrypt the source block then (4 – n1) no. of iteration is used to decry pt the cipher block. Value of n1 is chosen dynamically to enhance the security.

If n1 = 1 then the encrypted block is

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

The equivalent decimal no. of two 8 bit binary numbers 00010000 and 01010111 are 16 and 87 respectively. 16 and 87 are ASCII values of the characters "►" and "W" respectively. So the word "At" is encrypted as "►W". For decryption, remaining 3 iterations are used.

If n1 = 2 then the encrypted block is

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

The equivalent decimal no. of two 8 bit binary numbers 01010100 and 01100001 are 84 and 97 respectively. 84 and 97 are ASCII values of the characters "T" and "a" respectively. So the word "At" is encrypted as "Ta". For decryption, remaining 2 iterations are used.

If n1 = 3 then the encrypted block is

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

The equivalent decimal no. of two 8 bit binary numbers 01010101 and 00010010 are 85 and 18 respectively. 85 and 18 are ASCII values of the characters "U" and "↕" respectively. So the word "At" is encrypted as "U↕". Only single iteration is used for decryption.

5. RESULTS AND ANALYSIS

In this section the results of analysis are given. The analysis includes the comparison of encryption time, decryption time, Character frequencies, Chi-square values, Avalanche and Strict Avalanche effects, Bit Independence. The comparative study between Triple-DES(168bits), AES(128bits) and SBSKCT has done on 20 files of 8 different types with different sizes varying from 330 bytes to 62657918 bytes (59.7 MB). 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 SBSKCT against the different files. Proposed SBSKCT 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 SBSKCT algorithms)

Sl. No.	Source File Size (in bytes)	File type	Encryption Time (in seconds)		
			TDES	AES	SBSKCT
1	330	Dll	0.001	0.001	0.004
2	528	Txt	0.001	0.001	0.008
3	96317	Txt	0.034	0.004	0.024
4	233071	Rar	0.082	0.011	0.067
5	354304	Exe	0.123	0.017	0.089
6	536387	Zip	0.186	0.023	0.143
7	657408	Doc	0.220	0.031	0.257
8	682496	Dll	0.248	0.031	0.073
9	860713	Pdf	0.289	0.038	0.125
10	988216	Exe	0.331	0.042	0.176
11	1395473	Txt	0.476	0.059	0.182
12	4472320	Doc	1.663	0.192	0.408
13	7820026	Avi	2.626	0.334	0.712
14	9227808	Zip	3.096	0.397	0.521
15	11580416	Dll	4.393	0.544	0.871
16	17486968	Exe	5.906	0.743	2.075
17	20951837	Rar	7.334	0.937	1.736
18	32683952	Pdf	10.971	1.350	2.285
19	44814336	Exe	15.091	1.914	3.271
20	62657918	Avi	21.133	2.689	6.452

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

Sl. No.	Source File Size (in bytes)	File type	Decryption Time (in seconds)		
			TDES	AES	SBSKCT
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	0.032
4	233071	Rar	0.087	0.017	0.064
5	354304	Exe	0.128	0.025	0.079
6	536387	Zip	0.202	0.038	0.067
7	657408	Doc	0.235	0.045	0.223
8	682496	Dll	0.266	0.046	0.147
9	860713	Pdf	0.307	0.060	0.101
10	988216	Exe	0.356	0.070	0.150
11	1395473	Txt	0.530	0.098	0.348
12	4472320	Doc	1.663	0.349	0.554
13	7820026	Avi	2.832	0.557	0.683
14	9227808	Zip	3.377	0.656	0.515
15	11580416	Dll	4.652	0.868	1.002
16	17486968	Exe	6.289	1.220	1.816

17	20951837	Rar	8.052	1.431	2.073
18	32683952	Pdf	11.811	2.274	3.809
19	44814336	Exe	16.253	3.108	3.390
20	62657918	Avi	22.882	4.927	6.092

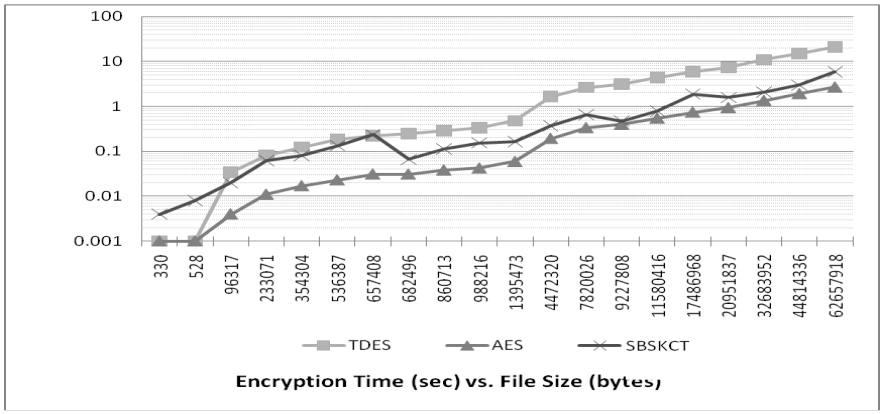


Fig. 1(a). Encryption Time (sec) vs. File Size (bytes) in logarithmic scale

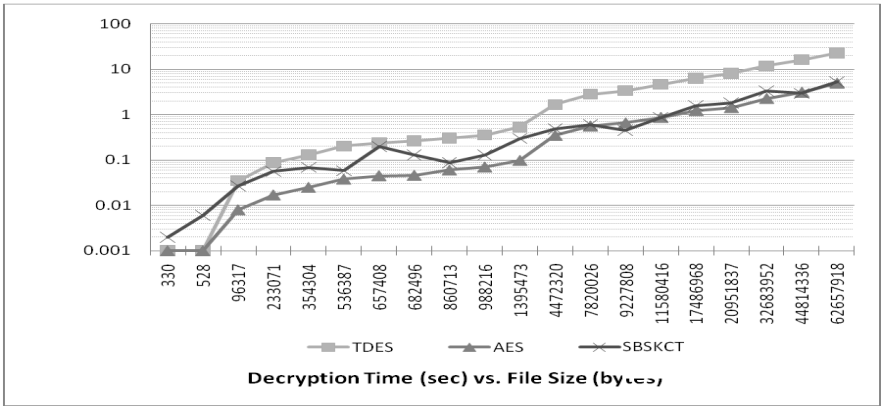


Fig. 1(b). Decryption Time (sec) vs. File Size (bytes) in logarithmic scale

5.2. ANALYSIS OF CHARACTER FREQUENCIES

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

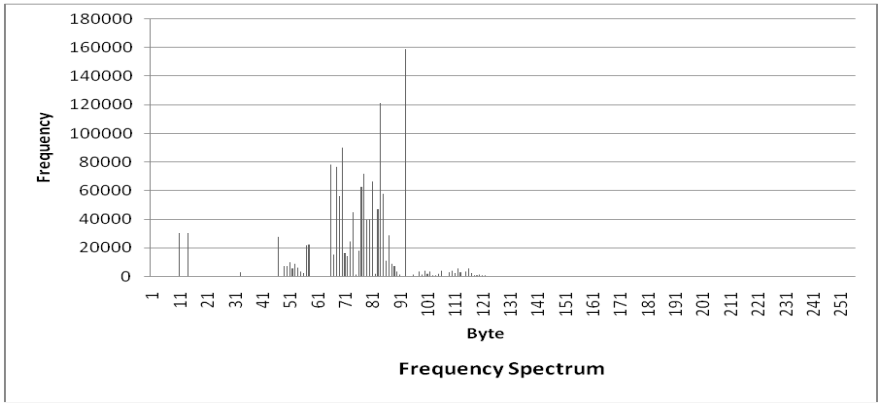


Fig. 2(a). Distribution of characters in source file

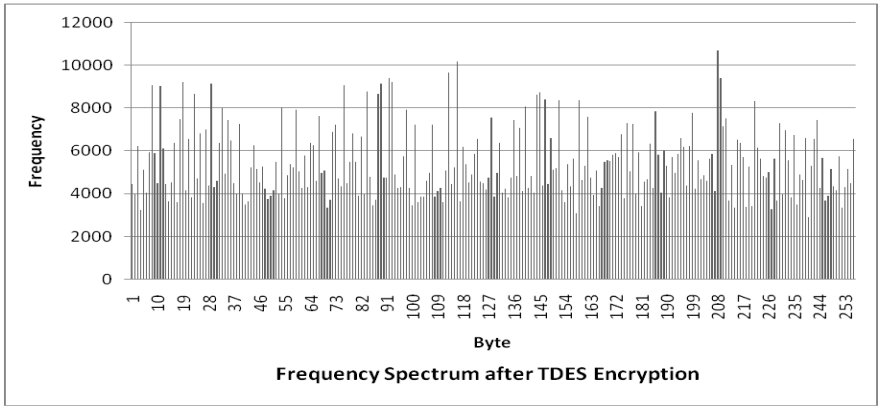


Fig. 2(b): Distribution of characters in TDES

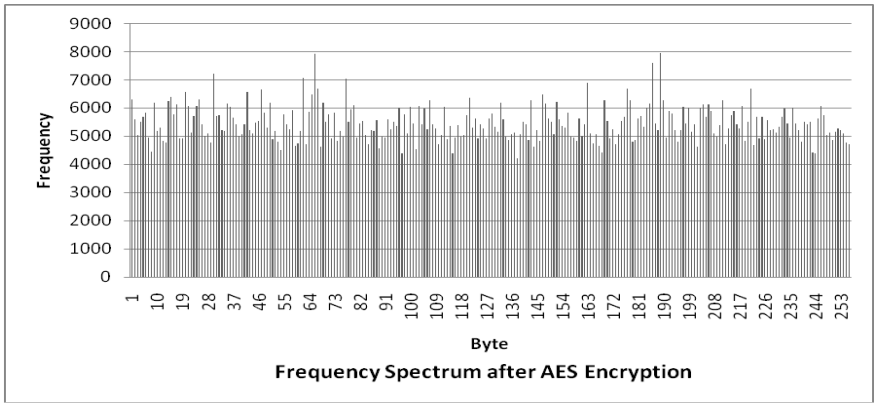


Fig. 2(c). Distribution of characters in AES

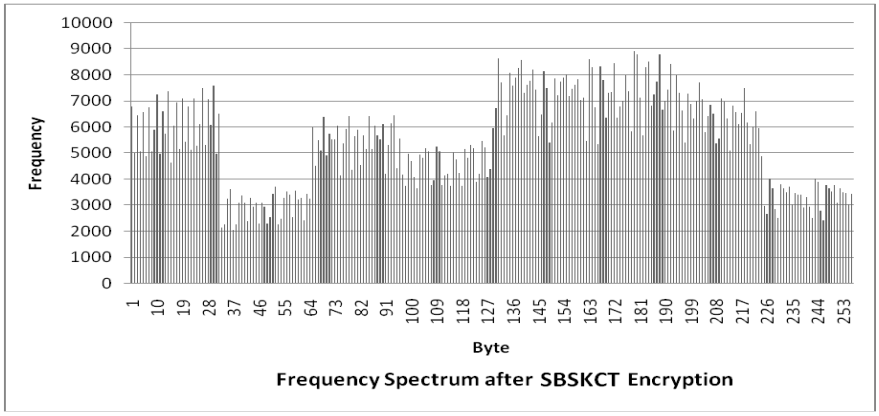


Fig. 2(d). Distribution of characters in SBSKCT

5.3. TESTS FOR NON-HOMOGENEITY

The test for goodness of fit (Pearson χ^2) has been performed between the source files (expected) and the encrypted files (observed). 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 SBSKCT against the different files.

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

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

Sl. No.	Source File Size (bytes)	File type	Chi-Square Values		
			TDES	AES	SBSKCT
1	330	dll	922	959	895
2	528	txt	1889	1897	1939
3	96317	txt	23492528	23865067	20174362
4	233071	rar	997	915	972
5	354304	exe	353169	228027	176731
6	536387	zip	3279	3510	3359
7	657408	doc	90750	88706	87988
8	682496	dll	29296	28440	26642
9	860713	pdf	59797	60661	56134
10	988216	exe	240186	245090	257169
11	1395473	txt	5833237390	5545862604	6771635306
12	4472320	doc	102678	102581	99874
13	7820026	avi	1869638	1326136	807744
14	9227808	zip	37593	37424	36715
15	11580416	dll	28811486	17081530	13759262
16	17486968	exe	8689664	8463203	7994999
17	20951837	rar	25615	24785	26491
18	32683952	pdf	13896909	13893011	15298623
19	44814336	exe	97756312	81405043	499844380
20	62657918	avi	3570872	3571648	4893222

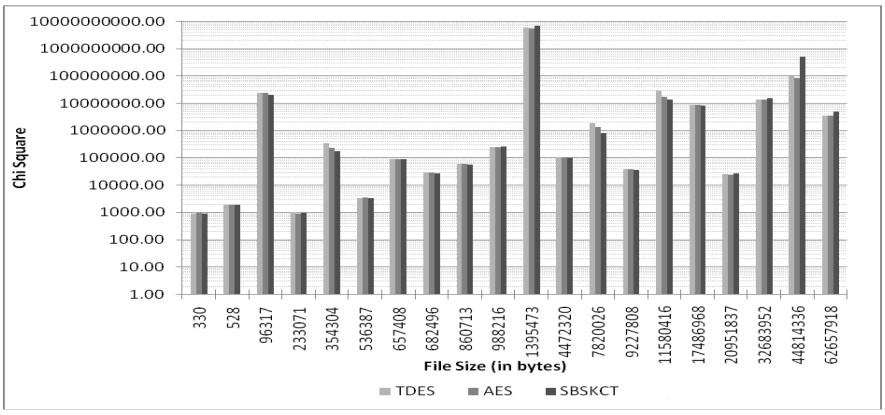


Fig.3 Chi-Square values for TDES, AES & SBSKCT in logarithmic scale.

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 0.9 – 1.0 for better visual interpretation.

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

Sl. No.	Source File Size (in bytes)	File type	Avalanche achieved		
			TDES	AES	SBSKCT
1	330	dll	0.99591	0.98904	0.96444
2	528	txt	0.99773	0.99852	0.97661
3	96317	txt	0.99996	0.99997	0.98984
4	233071	rar	0.99994	0.99997	0.99456
5	354304	exe	0.99996	0.99999	0.99102
6	536387	zip	0.99996	0.99994	0.99576
7	657408	doc	0.99996	0.99999	0.99339
8	682496	dll	0.99998	1.00000	0.99188
9	860713	pdf	0.99996	0.99997	0.99386
10	988216	exe	1.00000	0.99998	0.98686
11	1395473	txt	1.00000	1.00000	0.99353
12	4472320	doc	0.99999	0.99997	0.99001
13	7820026	avi	1.00000	0.99999	0.99455
14	9227808	zip	1.00000	1.00000	0.99899
15	11580416	dll	1.00000	0.99999	0.99865

16	17486968	exe	1.00000	0.99999	0.99908
17	20951837	Rar	1.00000	1.00000	0.99900
18	32683952	pdf	0.99999	1.00000	0.99896
19	44814336	exe	0.99997	0.99997	0.99986
20	62657918	Avi	0.99999	0.99999	0.99994

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

Sl. No.	Source File Size (in bytes)	File type	Strict Avalanche achieved		
			TDES	AES	SBSKCT
1	330	Dll	0.98645	0.98505	0.89548
2	528	Txt	0.99419	0.99311	0.96732
3	96317	Txt	0.99992	0.99987	0.97625
4	233071	Rar	0.99986	0.99985	0.99137
5	354304	Exe	0.99991	0.99981	0.98773
6	536387	Zip	0.99988	0.99985	0.99456
7	657408	Doc	0.99989	0.99990	0.99022
8	682496	Dll	0.99990	0.99985	0.98507
9	860713	Pdf	0.99990	0.99993	0.98701
10	988216	Exe	0.99995	0.99995	0.97168
11	1395473	Txt	0.99990	0.99996	0.99024
12	4472320	Doc	0.99998	0.99995	0.98222
13	7820026	Avi	0.99996	0.99996	0.99068
14	9227808	Zip	0.99997	0.99998	0.99897
15	11580416	Dll	0.99992	0.99998	0.99685
16	17486968	Exe	0.99996	0.99997	0.99747
17	20951837	Rar	0.99998	0.99996	0.99896
18	32683952	Pdf	0.99997	0.99998	0.99892
19	44814336	Exe	0.99991	0.99990	0.99986
20	62657918	Avi	0.99997	0.99998	0.99989

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

Sl. No.	Source File Size (in bytes)	File type	Bit Independence achieved		
			TDES	AES	SBSKCT
1	330	Dll	0.49180	0.47804	0.39143
2	528	Txt	0.22966	0.23056	0.20942
3	96317	Txt	0.41022	0.41167	0.42818
4	233071	Rar	0.99899	0.99887	0.98266
5	354304	Exe	0.92538	0.92414	0.93376
6	536387	Zip	0.99824	0.99753	0.99166
7	657408	Doc	0.98111	0.98030	0.97182
8	682496	Dll	0.99603	0.99560	0.96489
9	860713	Pdf	0.97073	0.96298	0.96629
10	988216	Exe	0.91480	0.91255	0.92836
11	1395473	Txt	0.25735	0.25464	0.24598
12	4472320	Doc	0.98881	0.98787	0.95295
13	7820026	Avi	0.98857	0.98595	0.96716
14	9227808	Zip	0.99807	0.99817	0.99707

15	11580416	Dll	0.86087	0.86303	0.85963
16	17486968	Exe	0.83078	0.85209	0.85420
17	20951837	Rar	0.99940	0.99937	0.99834
18	32683952	Pdf	0.95803	0.95850	0.95689
19	44814336	Exe	0.70104	0.70688	0.82535
20	62657918	Avi	0.99494	0.99451	0.99564

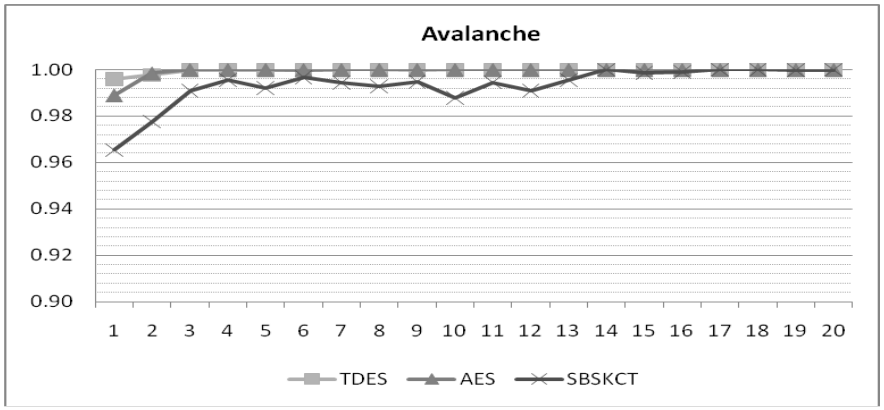


Fig.4(a) Comparison of Avalanche effect between T-DES, AES and SBSKCT

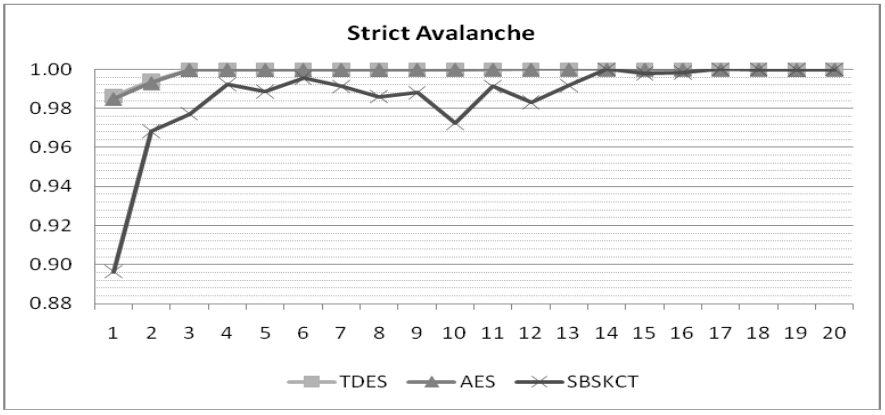


Fig4(b) Comparison of Strict Avalanche effect between TDES, AES and SBSKCT

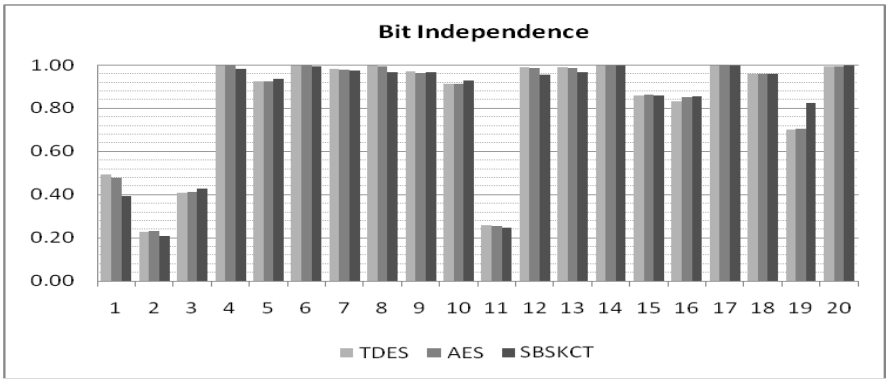


Fig.4(c) Comparison of Bit Independence criterion between TDES, AES and SBSKCT

6. CONCLUSION

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

REFERENCES

[1] M. Bellare and P. Rogaway, "*On the construction of variable length input Ciphers*", in Proceedings of Fast Software Encryption. LNCS, vol. 1636, pp. 231–244. Springer, Heidelberg, 1999.

[2] S.Patel, Z.Ramzan and G.Sundaram, "*Efficient constructions of variable-input-length block ciphers*", in Proceedings of Selected Areas in Cryptography 2004. LNCS, vol. 3357. Springer, Heidelberg , 2004**.**

[3] 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.

[4] Rajeev Chatterjee and J. K. Mandal , "*Authentication of PCSs with Cascaded Encryption technique (CE Technique)*", in proceedings of Fourth International Conference on Bridging the Digital Divide , Asian Applied Computing Conference (AACC 2007), Kathmandu, Nepal, 13-15th December 2007.

[5] 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.

[6] Som S., Mandal J. K., (2008) "*A Session Key Based Secure-Bit Encryption Technique (SBET)*", in Proceedings of 2nd National Conference on Computing for Nation Development, Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi, 08th -09th February 2008.

[7] Jayanta Kumar Pal, J. K. Mandal, and Somsubhra Gupta, "*Composite Transposition Substitution Chaining Based Cipher Technique*" in proceedings of 16th International Conference on Advanced Computing and Communication(ADCOM 2008), MIT Campus, Anna University Chennai, India, pp. 433-439, 14th-17th December 2008.

[8] 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.

[9] 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.

[10] *"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.

[11] *"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.