International Journal of Research in Advent Technology, Vol.5, No.1, January 2017
E-ISSN: 2321-9637
Available online at www.ijrat.org

Symmetric Key Cryptography using Digital Circuit based on One Left Shift

Dr. U Ravi Babu
1Professor, Department of CSE, Narsimha Reddy Engg. College, HYD, TS, India
Uppu.ravibabu@gmail.com

Abstract-A session based symmetric key cryptographic technique has been proposed in this paper and it is termed as 1LS. The plain text is considered as a finite number of binary bits and is chopped into blocks with variable length. Bit positions into the block are shifted to generate the cipher text using the digital circuit based on one left shift. The session key is generated randomly from the chopping information of plain text. Results are computed using twenty files with different sizes and types to compare 1LS with standard symmetric key cryptographic techniques Triple-DES (168bits) and AES (128bits) with respect to the Encryption and Decryption times, Avalanche and Strict Avalanche values, Bit independence value, Chi-square values and some other statistical measures like median, mode, standard deviation and correlation coefficient.

Index Terms-Left Shift, AES, Symmetric key cryptography, Session key, Triple-DES.

1. INTRODUCTION

In modern era every computer is connected virtually. It is very important to secure our information from eavesdroppers. So data security becomes main concern of modern life. Cryptography is an important aspect for secure communication to protect important data. As a result continuous research works [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] are going on in this field of cryptography to enhance the network security.

Section 2 of this paper explains the proposed technique. Section 3 deals with the algorithms for encryption, decryption and session 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 and AES. Conclusions are drawn in section 6.

2. TECHNIQUE

1LS considers the input file as a finite number of binary bits. The binary bits are chopped dynamically to fit into blocks of length 8n where n ε N, N is the set of Natural numbers. The block size and number of block are written into file to generate session based key. The i^{th} position bit of the original block is mapped to the j^{th} position of encrypted block. This mapping is bijective in nature. The bit position value (say value i) of the original block having block length 2^k, varies from 0 to $(2^k -1)$, is converted into k-bit binary number and the corresponding binary bits are sent into k-input digital circuit. For each 2^k number of combination of inputs, the output of the circuit produces unique 2^k number of k-bit binary numbers. Figures 1 and 2 show the block diagram of circuit for encryption and decryption respectively. For encryption the input bits are identified by IE_1, IE_2, IE_3, …, IE_k(where IE_1 is the MSB and $IE_k$is the LSB) and output bits are identified by OE_1, OE_2, OE_3, …, OE_k(where OE_1 is the MSB and $OE_k$is the LSB). The output bits of the circuit for encryption are defined as

$$OE_j = IE_{(j+1)} \text{ for } j=1, 2, \dots, (k-1)$$
$$= IE_1 \text{ for } j=k$$

The bits are converted to the corresponding decimal to find the value of j. For decryption the input bits are identified by ID_1,ID_2, ID_3, …, ID_k(where ID_1 is the MSB and $ID_k$is the LSB) and output bits are identified by $OD_1OD_2OD_3…OD_k$(where OD_1 is the MSB and $OD_k$is the LSB). The output bits for decryption are represented as

$$OD_j = ID_{(j-1)} \text{ for } j=2, 3, \dots, k$$
$$= ID_k \text{ for } j=1$$

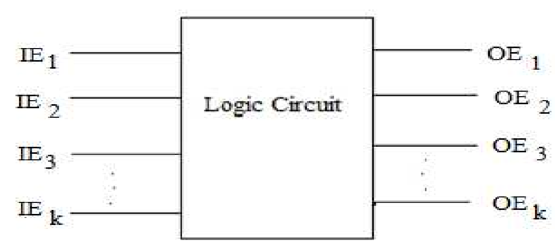

Figure 1: The block diagram of circuit for encryption

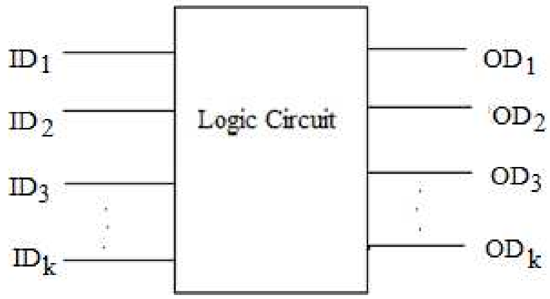

Figure 2: The block diagram of circuit for encryption

In this section Encryption, Decryption and Session key generation algorithms are explained in details.

A. Encryption Algorithm:

Step 1: The plain text i.e input file is considered as a finite number of binary bits.

Step 2: The bits are chopped dynamically into blocks of different lengths like 8 / 16 / 24 / 32 / 40 / 48 / 56 /….[i.e. 8n for n=1,2,3,4…] as follows. First n1 no. of bits is considered as x1 no. of blocks with block length y1 where n1 = x1 * y1. Next n2 no. of bits is considered as x2 no. of blocks with block length y2 where n2 = x2 * y2 and so on. Finally nm no. of bits is considered as xm no. of blocks with block length ym (= 8) where nm = xm * ym. So no padding is required.

Step 3: The bit position value (say value i) of the plain text block having block length 2^k, varies from 0 to (2^k - 1), is converted into k-bit binary number and the corresponding binary bits are sent into k-input digital circuit as $IE_1IE_2IE_3…IE_k$(where IE_1 is the MSB and $IE_k$is the LSB).

Step4: The output bits of the digital circuit $OE_1OE_2OE_3…OE_k$are expressed as

$$OE_j = IE_{(j+1)} \text{ for } j=1, 2, 3, …, (k-1)$$
$$= IE_1 \text{ for } j=k$$

Step 5: The output bits of the circuit are converted into decimal number to get the corresponding bit position value (say value j) in the encrypted block of length 8n.

Step 6: The i^{th} bit of the plain text block is placed to j^{th} bit of the encrypted block. The relationship between i and j for the block with block length 2^k can also be expressed using the function given below

$j = f(i) = 2*i + (1 - 2^k)(2*i / 2^k)$ where / gives the integer part of the quotient

The cipher text is formed by converting the encrypted block to its corresponding characters.

B. Decryption Algorithm:

Step 1: The cipher text is considered as a finite number of binary bits.

Step 2: Processing the session key the binary bits are sliced into manageable sized block.

Step 3: The bit position value (say value i) of the cipher text block having block length 2^k is converted into k-bit binary number and the corresponding binary bits are sent into k-input digital circuit as $ID_1ID_2ID_3…ID_k$(where ID_1 is the MSB and $ID_k$is the LSB)..

Step4: The output bits of the digital circuit $OD_1OD_2OD_3…OD_k$are expressed as

$$OD_j = ID_{(j-1)} \text{ for } j=2, 3, 4, …,k$$
$$= ID_k \text{ for } j=1$$

Step 5: The output bits of the circuit are converted into decimal number to get the corresponding bit position value (say value j) in the decrypted block of length 8n.

Step 6: The i^{th} bit of the cipher text block is placed to j^{th} bit of the decrypted block. The relationship between i and j for the block with block length 2^k can also be expressed using the function given below

$$j = f(i) = \{ i + (2^k - 1)*(i \% 2) \} / 2$$

where / gives the integer part of the quotient and % gives the remainder part.

The plain text is regenerated by converting the decrypted block to its corresponding characters.

C. Session Key Generation Algorithm:

1LS generates a session based key for one time use in a particular session. The input bit stream is divided into 16 portions where 1^{st} portion contains 20% of the total file size, 2^{nd} portion contains 20% of the remaining file size and so on. Each portion is divided into x no. of blocks with block length y (=8n) where value of n is selected dynamically for first fifteen portions. Finally last (i.e. 16^{th}) portion is divided into x_{16} no. of blocks with block length 8 bits (i.e. y_{16} = 8). So no padding is required. Total length of the input binary stream is = $x_1*y_1+x_2*y_2+…….. +x_{16}*y_{16}$. The value of n for each portion is stored as a character in the key file. So the key file contains sixteen characters.

EXAMPLE

Let consider the word "Ma". The 8 bit representation of the above characters "M" and "a" are '01001101' and '01100001' respectively. The bits are stored into an array from MSB to LSB as 8 bit or 16 bit block length randomly. Now the position of 8 or 16 bit is converted into binary and stored into another array and following the above logic bits are changed to generate the new position. Figure 2 shows the encryption steps for the above example.

Case I: If block length is 8 then the encrypted string is '0111000100101001'. Two 8 bit binary numbers are '01110001' (=$[113]_{10}$) and '00101001' (=$[41]_{10}$) is encrypted from binary string and the corresponding characters are "q" and ")" respectively. So "Ma" is converted into "q)".

Case II: If block length is 16 then the encrypted string is '0011010010100011'. Two 8 bit binary numbers are '00110100' (=$[52]_{10}$) and '10100011' (=$[163]_{10}$) is encrypted from binary string and the corresponding characters are "4" and "£" respectively. So "Ma" is converted into "4£".

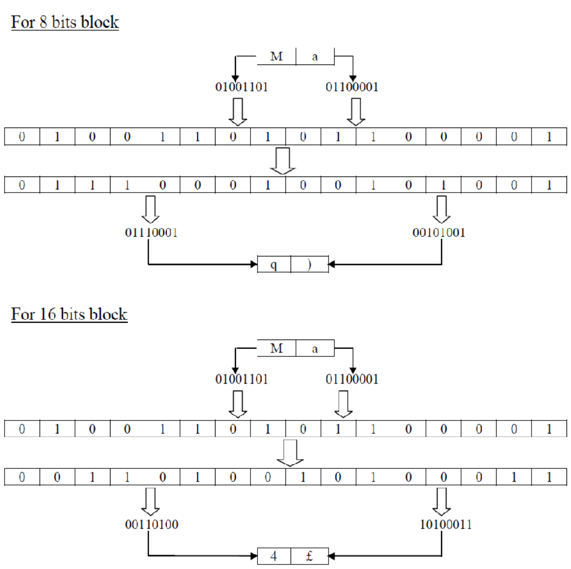

Figure 3: The encryption steps for the given example

3. RESULTS

Results are generated using twenty files with different file sizes varying from 50 bytes to 285 MB (approx.) and eleven different file types (like .txt, .dll, .doc etc). Comprehensive analysis and comparison has been made between the proposed technique 1LS, Triple-DES (168bits) and AES (128bits) with respect to the following parameters.

[1]Encryption and Decryption Times:

The encryption and decryption times are taken the differences between processor clock ticks at the starting of execution and ending of execution. The minimum time indicates the highest speed of execution. Encryption and Decryption times (in milliseconds) of twenty different files are calculated for Triple-DES, AES and 1LS. Tables 1 and 2 show the encryption and decryption times respectively of TDES, AES and 1LS for different source files. Files are taken in ascending order of their size. Figures 4 and 5 indicate the graphical representation of encryption times and decryption times respectively for TDES, AES and 1LS of different source files.

Table1: Encryption times for TDES,AES and1LS

Sl. No.	File type	File size (Bytes)	Encryption time (in m.sec)		
			TDES	AES	1LS
1	txt	50	0	0	0
2	zip	288	0	0	0
3	txt	500	16	0	0
4	txt	2410	0	0	0
5	jpg	5400	15	0	15
6	docx	12660	47	16	31
7	exe	21492	16	0	46
8	jpg	50735	16	0	62
9	rar	115595	31	0	125
10	dll	215416	47	15	203
11	exe	624128	125	32	577
12	docx	1224413	218	31	1107
13	dll	1409024	266	78	1279
14	jpg	3588725	592	94	3260
15	pdf	4446250	749	125	4040
16	avi	7355928	1341	203	6692
17	rtf	15766836	2652	421	14398
18	doc	43456000	7129	1154	39749
19	rar	77246022	12698	2060	70450
20	avi	144161826	23883	3791	131492

Table 2: Decryption times for TDES,AES and1LS

Sl. No.	File type	File size (Bytes)	Decryption time (in m.sec)		
			TDES	AES	1LS
1	txt	50	0	0	0
2	zip	288	0	0	16
3	txt	500	0	0	16

Sl. No.	File type	File size (Bytes)	Decryption time (in m.sec)		
			TDES	AES	1LS
4	txt	2410	0	0	0
5	jpg	5400	0	16	16
6	docx	12660	0	0	31
7	exe	21492	15	0	47
8	jpg	50735	15	15	78
9	rar	115595	31	15	125
10	dll	215416	46	31	187
11	exe	624128	124	62	562
12	docx	1224413	234	78	1107
13	dll	1409024	265	94	1295
14	jpg	3588725	718	202	3245
15	pdf	4446250	904	234	4024
16	avi	7355928	1451	374	6661
17	rtf	15766836	3105	905	14243
18	doc	43456000	8455	2636	39296
19	rar	77246022	15179	4399	69873
20	avi	144161826	28221	8642	130385

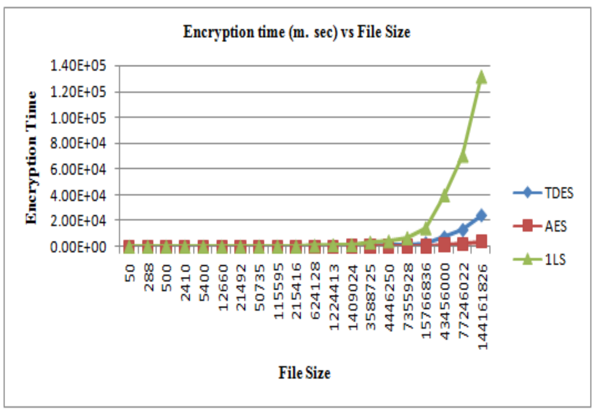

Figure 4:Graphical Representation of encryption times against file size in logarithmic scale

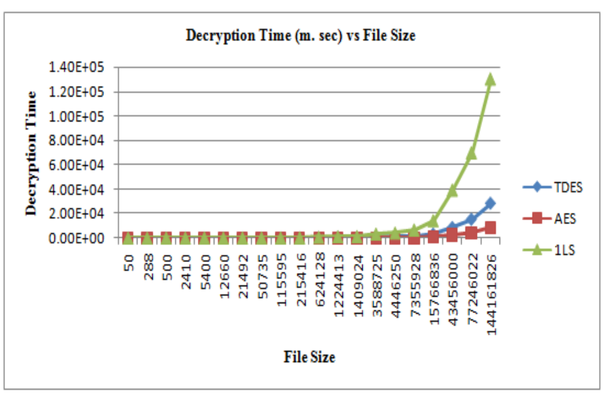

Figure 5:Graphical Representation of decryption times against file size in logarithmic scale

[2]Avalanche & Strict Avalanche values and Bit Independence Criterion:

Avalanche, Strict avalanche and Bit Independence are cryptographic test mechanisms which measure the degree of security of cryptographic technique. The bit changes among encrypted bytes for a single bit change in the original message sequence for the entire or a large number of bytes. The values of Avalanche and Strict Avalanche closer to 1.0 may indicate the high degree of security.Tables 3, 4 and 5 show the Avalanche & Strict Avalanche values and Bit Independence values respectively for Triple-DES, AES and 1LS which are closer to 1. Figures 6, 7 and 8 represent the graphical representation of Avalanche and Strict Avalanche and Bit Independence values respectively with respect to different files where files are taken in ascending order of its sizes. This analysis indicates that 1LS may provide good security.

Table 3: Avalanche Values for TDES,AES and1LS

Sl. No.	File type	File size (Bytes)	Avalanche achieved		
			TDES	AES	1LS
1	txt	50	0.99048	0.99324	0.70732
2	zip	288	0.98564	0.99835	0.25000
3	txt	500	0.99565	0.99374	0.25000
4	txt	2410	0.99956	0.99950	0.25000
5	jpg	5400	0.99971	0.99957	0.91036
6	docx	12660	0.99990	0.99985	0.25000
7	exe	21492	0.99965	0.99970	0.93251
8	jpg	50735	0.99997	0.99971	0.99856
9	rar	115595	0.99985	0.99997	0.99905
10	dll	215416	0.99989	0.99994	0.98640
11	exe	624128	0.99996	0.99995	0.97756
12	docx	1224413	0.99998	0.99999	0.98663
13	dll	1409024	0.99998	0.99999	0.95852
14	jpg	3588725	0.99998	0.99998	0.99686
15	pdf	4446250	0.99999	0.99999	0.99143
16	avi	7355928	0.99999	0.99998	0.99469
17	rtf	15766836	0.99996	0.99992	0.95068
18	doc	43456000	0.99994	0.99906	0.96923
19	rar	77246022	0.99998	0.99999	0.96758
20	avi	144161826	0.99998	0.99999	0.82106

Table 4: Strict Avalanche Values for TDES,AES and1LS

Sl. No.	File type	File size (Bytes)	Strict Avalanche achieved		
			TDES	AES	1LS
1	txt	50	0.99048	0.99324	0.70732
2	zip	288	0.98564	0.99835	0.25000
3	txt	500	0.99565	0.99374	0.25000
4	txt	2410	0.99956	0.99950	0.25000
5	jpg	5400	0.99971	0.99957	0.91036
6	docx	12660	0.99990	0.99985	0.25000

Sl. No.	File type	File size (Bytes)	Strict Avalanche achieved		
			TDES	AES	1LS
7	exe	21492	0.99965	0.99970	0.93251
8	jpg	50735	0.99997	0.99971	0.99856
9	rar	115595	0.99985	0.99997	0.99905
10	dll	215416	0.99989	0.99994	0.98640
11	exe	624128	0.99996	0.99995	0.97756
12	docx	1224413	0.99998	0.99999	0.98663
13	dll	1409024	0.99998	0.99999	0.95852
14	jpg	3588725	0.99998	0.99998	0.99686
15	pdf	4446250	0.99999	0.99999	0.99143
16	avi	7355928	0.99999	0.99998	0.99469
17	rtf	15766836	0.99996	0.99992	0.95068
18	doc	43456000	0.99994	0.99906	0.96923
19	rar	77246022	0.99998	0.99999	0.96758
20	avi	144161826	0.99998	0.99999	0.82106

Table 5: Bit Independence Values for TDES,AES and1LS

Sl. No.	File type	File size (Bytes)	Bit Independence achieved		
			TDES	AES	1LS
1	txt	50	0.15643	0.25978	0.00000
2	zip	288	0.39354	0.35724	0.00000
3	txt	500	0.41180	0.39897	0.00000
4	txt	2410	0.48006	0.48396	0.00000
5	jpg	5400	0.97112	0.97528	0.77979
6	docx	12660	0.97565	0.97208	0.00000
7	exe	21492	0.63366	0.60980	0.67265
8	jpg	50735	0.99751	0.99773	0.99070
9	rar	115595	0.99778	0.99734	0.99815
10	dll	215416	0.75333	0.75456	0.77753
11	exe	624128	0.74603	0.74036	0.77150
12	docx	1224413	0.99086	0.99095	0.98326
13	dll	1409024	0.72549	0.72959	0.85618
14	jpg	3588725	0.99470	0.99470	0.99277
15	pdf	4446250	0.97532	0.96328	0.99115
16	avi	7355928	0.99326	0.99187	0.99139
17	rtf	15766836	0.37358	0.33894	0.35744
18	doc	43456000	0.34031	0.22074	0.43120
19	rar	77246022	0.99979	0.99969	0.96842
20	avi	144161826	0.98842	0.98788	0.79361

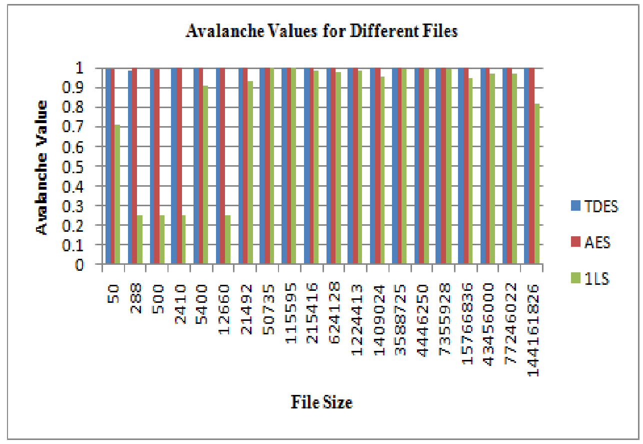

Figure 6:Graphical Representation of Avalanche value against file size in logarithmic scale

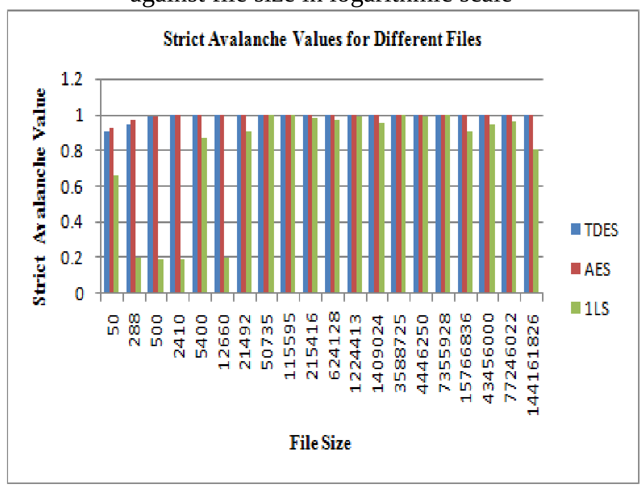

Figure 7:Graphical Representation of StrictAvalanche value against file size in logarithmic scale

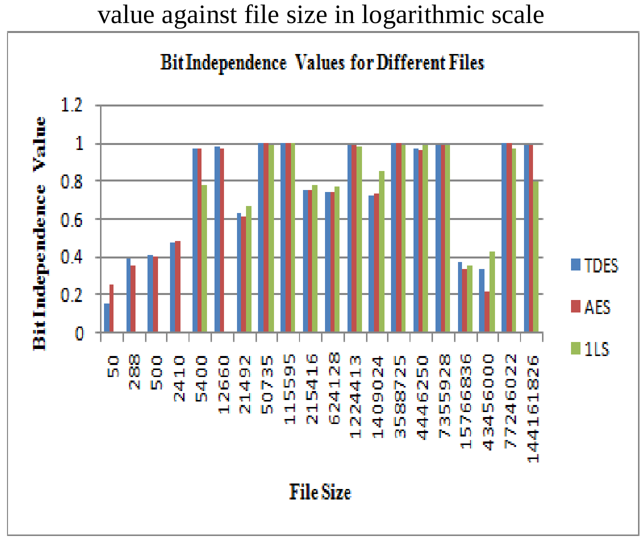

Figure 8:Graphical Representation of Bit Independence value against file size in logarithmic scale

[3]Chi-Square Values:

The large Chi-square value compared with tabulated value may indicate a high degree of non-homogeneity among source and encrypted files. Table 6 shows the Chi-square values for Triple-DES (168bits), AES (128bits) and 1LS. Average chi-square values of Triple-DES (168bits), AES (128bits) and 1LS are 34143114280, 32603653459 and 1.01805E+11 respectively. Figure 9 shows the comparison of the Chi-square values of all three techniques against the twenty source files. From the figures it is observed that the degree of non-homogeneity of the encrypted files with respect to source files using the technique 1LS is very high. Therefore it may conclude that 1LS provides good security.

Table 6: Chi-Square Values for TDES,AES and1LS

Sl. No	File type	File size (Bytes)	Chi-Square values		
			TDES	AES	1LS
1	txt	50	114	111	164
2	zip	288	503	529	357
3	txt	500	1470	1546	8242
4	txt	2410	24059	20981	124028
5	jpg	5400	936	946	865
6	docx	12660	18333	9343	1068
7	exe	21492	1044334	481174	163688
8	jpg	50735	1373	1301	5033
9	rar	115595	1031	1038	761
10	dll	215416	530985	473027	777200
11	exe	624128	2027106	1848171	2776952
12	docx	1224413	54964	55574	132742
13	dll	1409024	3219751	3139562	3308113
14	jpg	3588725	78928	79292	106741
15	pdf	4446250	413610	369563	829913
16	avi	7355928	438208	442887	320848
17	rtf	15766836	6.825E+11	6.517E+11	2.035E+12
18	doc	43456000	288821670	267709342	880702598
19	rar	77246022	61298	61037	7410
20	avi	144161826	15912744	15646387	12659237
Average			3.4E+10	3.2E+10	1.0E+11

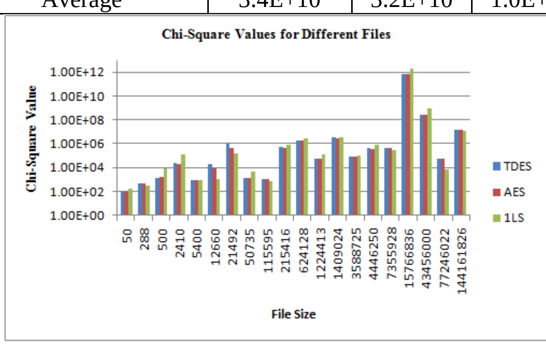

Figure 9:Graphical Representation of Chi-Square value against file size in logarithmic scale

[4]Other Statistical Measures:

Measure of Central tendency in terms of median, mode and measure of Dispersion in terms of standard deviation has been performed as a measure of non-homogeneity. Values of median, mode and standard deviation of source stream and encrypted stream using 1LS for three different files has given in Table 7. The correlation coefficient between the source stream and cipher stream is measured using Karl Pearson's Product Moment Correlation Coefficient formula. In Table 7 product moment correlation coefficient of three types of source streams and the corresponding encrypted streams has been also presented from which it is observed that there is negligible correlation between the source stream and the cipher stream. This result indicates that 1LS may provide good security.

Table 7: Median, Mode, standard Deviation and Correlation coefficient values using 1LS

Value of	**Stream**	**S08.png**	**S10.dll**	**S17.rtf**
Median (character with ASCII value)	Source	123	102	99
	Encrypted	121	96	57
Mode (character with ASCII value)	Source	0	0	92
	Encrypted	0	0	60
Standard Deviation	Source	0.93E2	0.24E4	0.22E6
	Encrypted	0.47E2	0.17E4	0.15E6
Correlation Coefficient	Source & Encrypted	0.69	0.83	0.09

4. CONCLUSIONS

The proposed technique 1LS in this paper is simple to understand and easy to implement using various high level languages. The performance of 1LS is quite satisfactory because of high processing speed and the measure of the degree of security is at par with Triple-DES and AES. It is applicable in message transmission of any form and any size. Some of the salient features of 1LS can be summarized as follows:

1. High degree of security.
2. Session based key implementation.
3. Block size independency.

REFERENCES

[1] Mandal, B.K., Bhattacharyya, D. and Bandyopadhyay, S.K. 2013. Designing and performance analysis of a proposed symmetric cryptography algorithm. In: International Conference on Communication Systems and Network Technologies (CSNT 2013), Gwalior, India, pp. 453–461, 6–8 April 2013.

[2] Paul, M. and Mandal, J.K. 2013. A Novel Generic Session Based Bit Level Cryptographic Technique based on Magic Square Concepts, International Conference on Global Innovations in Technology and Sciences (ICGITS 2013), 4-6 April 2013, Kottayam, Kerala, India, pp. 156-163.

[3] Niemiec, M. and Machowski, L. 2012. A new symmetric block cipher based on key-dependent S-boxes, 4th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT 2012), 3-5 October 2012, St. Petersburg, Russia, pp. 474 – 478.

[4] Cheng H. and Ding Q. 2012. Overview of the Block Cipher, Second International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC 2012), 8-10 December 2012, Harbin, China, pp. 1628 – 1631.

[5] Paul, M. and Mandal, J.K. 2012. A Universal Session Based Bit Level Symmetric Key Cryptographic Technique to Enhance the Information Security. International Journal of Network Security & Its Application (IJNSA) 4(4), 123–136.

[6] Navin, A.H., Oskuei, A.R., Khashandarag, A.S. and Mirnia, M, 2011. A Novel Approach Cryptography by using Residue Number System. In: 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT 2011), Seogwipo, South Korea, November 29-December 01, pp. 636–639.

[7] Paul, M. and Mandal, J. K. 2011. A Novel Generic Session Based Bit Level Cryptographic Technique to Enhance Information Security, International Journal of Computer Science and Network Security (IJCSNS), December 2011, Vol. 11, No. 12, pp. 117-122.

[8] Som, S., Chatergee N. S. and Mandal, J. K. 2011. Key Based Bit Level Genetic Cryptographic Technique (KBGCT), 7th International Conference on Information Assurance and Security (IAS), 5-8 December 2011, Melaka, Malaysia, pp. 240-245.

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

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