A Symmetric Key Cryptographic Technique Based on Frame Rotation of an Even Ordered Square Matrix

Joyita Goswami (Ghosh) and Manas Paul

Abstract A session based symmetric key cryptographic technique has been proposed in this paper. It considered the plain text as a stream of finite number of binary bits. Using frame rotation technique of square matrix, input bits are oriented to generate cipher text. A session key is generated from the plain text information. Results are generated to compare the proposed technique with Triple-DES (168 bits), AES (128 bits) with respect to different parameters.

Keywords FRSM · AES · Triple DES · Session based key · Avalanche · Strict avalanche · Bit independence · Chi-square

1 Introduction

In modern era every computer is connected virtually, so data security becomes main concern of modern life. Cryptography is an important aspect for secure communication to protect important data, so network security is the most important topic of researchers [1–5].

The proposed technique is a symmetric key cryptography and it is termed as FRSM where the plain text is considered as a stream of binary bits. The input bit stream chopped dynamically into manageable size of blocks such that bits of each block fitted into an even ordered square matrix. Bit positions are shifted to generate the cipher text using the concept of frame rotation of that square matrix. A session key is generated for one time use in a session of transmission using the information

J. Goswami (Ghosh) (✉) · M. Paul
Deptartment of Computer Application, JIS College of Engineering,
Kalyani, West Bengal, India
e-mail: joyitagoswami@gmail.com

M. Paul
e-mail: manaspaul@rediffmail.com

J.K. Mandal et al. (eds.), *Information Systems Design and Intelligent Applications*,
Advances in Intelligent Systems and Computing 339,
DOI 10.1007/978-81-322-2250-7_31

of block sizes. The plain text can be regenerated from the cipher text using the session key information during decryption.

Section 2 describes the proposed technique along with the algorithms for encryption, decryption and key generation. Section 3 explains FRSM with an example and Sect. 4 contains the results and analysis. Conclusions are drawn in Sect. 5.

2 Technique

FRSM consider the plain text as a stream of finite number of binary bits. This bit stream is chopped dynamically into blocks with variable sizes. The block size information of any particular session generates the session key. During encryption the bits of the blocks are taken MSB (Most Significant Bit) to LSB (Least Significant Bit) to fit row-wise into an even order ($2n \times 2n$) square matrix. Any square matrix of order 2n can be imagine as n number of square frames where inner most frame is denoted as frame 1 and outer most frame as frame n. Frame k contains $4(2k - 1)$ number of cells. Cells of each kth frame are shifted by circular fashion along clock-wise direction by k positions. After rotation bits are taken from the square matrix row-wise to form the encrypted block. Cipher text is generated from these encrypted blocks. For decryption the cipher text is considered as a stream of binary bits. Processing the session key the binary bits are sliced into desired sized blocks. The bits of the blocks are taken MSB to LSB to fit row-wise into a square matrix of order 2n. Cells of kth frame of that matrix are shifted circularly along anti clock-wise direction by k positions. Now bits are collected from the square matrix row-wise to form decrypted blocks. Plain text is regenerated from decrypted blocks.

Sections 2.1, 2.2 and 2.3 explain encryption algorithm, decryption algorithm and generation of session key respectively.

2.1 Encryption Algorithm

Input: Source stream i.e. plaintext.
Output: Encrypted stream i.e. ciphertext.
Method: The process takes binary stream and generates encrypted bit stream through a combination of S-Box and P-Box operations.

STEP 1: *The plain text i.e. the input file is considered as a stream of finite number of binary bits.*

STEP 2: *This binary string chopped dynamically 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 $\sqrt{y}$ (=2n) is generated for each block of length y. The binary bits of the block are taken from MSB to LSB to fit row-wise into this square matrix.*

STEP 4: *Any square matrix of even order (say $2n \times 2n$) can be imagine as n number of square frames like frame 1, frame 2, frame 3, ..., frame n where inner most frame is denoted as frame 1 and outer most frame as frame n. Frame k contains total $4(2k - 1)$ number of cells. Cells of frame k are shifted by circular fashion along clockwise direction by k positions.*

STEP 5: *Bits are taken row-wise from the square matrix to generate the encrypted block of length y.*

STEP 6: *The cipher text is formed converting the encrypted binary string into corresponding characters.*

2.2 Decryption Algorithm

Input: Encrypted stream i.e. ciphertext and session key.
Output: Source stream i.e. plaintext.
Method: Process takes encrypted binary stream and generates source stream.

STEP 1: *The cipher text is considered as a stream of binary bits.*

STEP 2: *Processing the session key information, this binary string breaks into manageable-sized blocks.*

STEP 3: *Square matrix of order $\sqrt{y}$ (2n) is generated for each block of length y. The binary bits of the block are taken from MSB to LSB to fit row-wise into this square matrix.*

STEP 4: *Any square matrix of even order (say $2n \times 2n$) can be imagine as n number of square frames like frame 1, frame 2, frame 3, ..., frame n where inner most frame is denoted as frame 1 and outer most frame as frame n. Frame k contains total 4(2 k–1) number of cells. Cells of each frame, say kth frame, are shifted by circular fashion along anti-clock-wise direction by k positions.*

STEP 5: *The decrypted binary string is generated taking the bits row-wise from the square matrix.*

STEP 6: *The plain text is regenerated converting the decrypted binary string into corresponding characters.*

2.3 Generation of Session Key

FRSM generates a session key for one time use in a particular session of transmission to ensure more security. The input binary bit stream is divided into 16 portions where 1st portion contains 20 % of total file size, 2nd portion contains 20 % of remaining file size and so on. Each portion is divided into x no. of blocks with block length y (=4n * n). The bits of this block are fitted into an even ordered (2n) square matrix where value of n is selected dynamically for first fifteen portions. Finally last (i.e. 16th) portion is divided into x_{16} no. of block with block length 4 bits (i.e. $y_{16} = 4$) and we generate 2×2 order matrix. So no padding is required. Total length of the input binary string is $= x_1 * y_1 + x_2 * y_2 + \cdots + x_{16} * y_{16}$. The value of n for each portion is stored as a character in the key file. So the key file contains fifteen characters.

3 FRSM with an Example

To illustrate FRSM, let us consider the word "Ma". The bit representation of the above word is '0100110101100001' which are taken from MSB to LSB into a block of length 16. Bits of the block fit row-wise into an even order (=4) square matrix. So matrix is imagined as two square frames, frame 1 and frame 2. Cells of

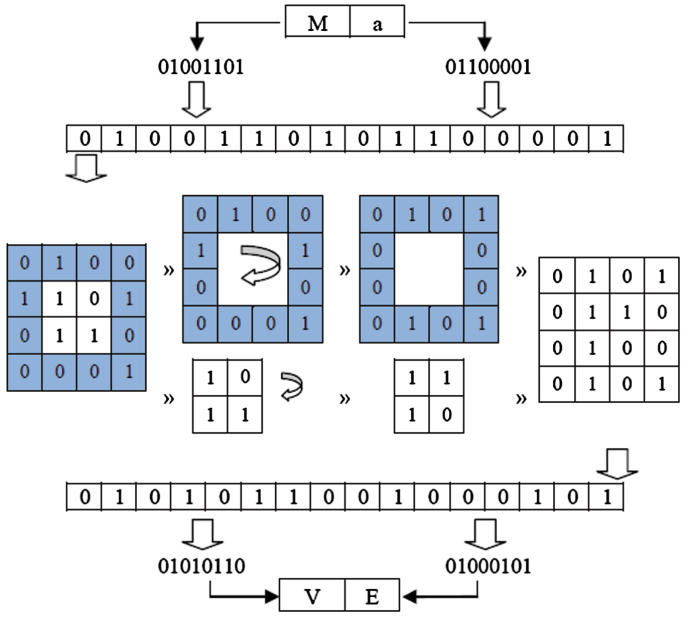

Fig. 1 Flow diagram of FRSM with the input word "Ma"

frame 1 and frame 2 are shifted by circular fashion along clock wise direction by one and two positions respectively. Now bits are taken row-wise to form the encrypted binary string. Two 8 bit binary numbers '01010110' and '01000101' are formed from the above encrypted string. The corresponding equivalent decimal values are 86 (≅character 'V') and 69 (≅character 'E') respectively. So the word "Ma" is encrypted as "VE". Figure 1 shows the flow diagram of FRSM with the input word "Ma".

4 Results and Analysis

The comparative study between Triple-DES (168 bits), AES (128 bits) and the proposed technique FRSM has done on twenty files of twelve different types with file sizes varying from 50 bytes to 137 MB (approx.). Results are generated for analysis and comparison of Encryption and Decryption times (in Sect. 4.1), Avalanche, Strict Avalanche and Bit Independence values (in Sect. 4.2), Chi-square values (in Sect. 4.3) and Character frequencies (in Sect. 4.4).

4.1 Analysis of Encryption and Decryption Times

Time taken is the difference between processor clock ticks at the starting and at the end of execution. The lower time indicates the higher speed of execution of the technique. Encryption and decryption times (in milliseconds) of different files calculated for Triple-DES (168 bits), AES (128 bits) and FRSM. Figures 2 and 3 show the graphical representation of encryption and decryption times against the

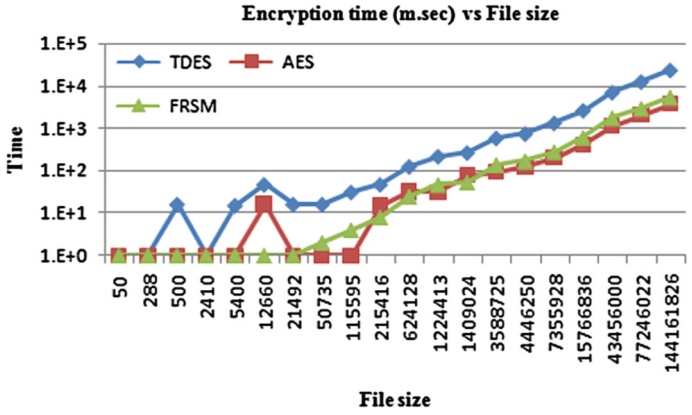

Fig. 2 Graphical representation of encryption times against file sizes in logarithmic scale

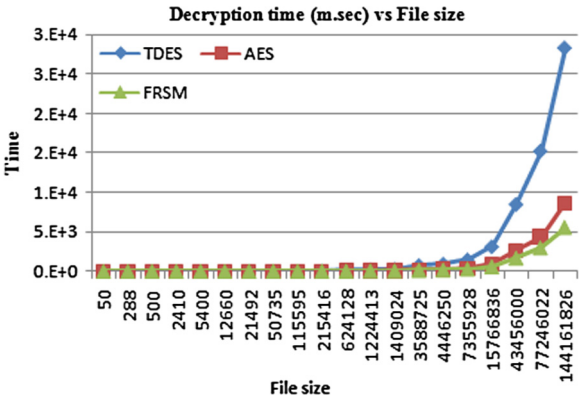

Fig. 3 Graphical representation of decryption times against file sizes in logarithmic scale

input files respectively. Techniques AES and FRSM take near same time to encrypt and decrypt the files where as TDES takes maximum time for both. In both the figures, the slopes of the curves for TDES are higher for larger source files.

4.2 Avalanche, Strict Avalanche and Bit Independence Values

Avalanche, Strict Avalanche and Bit Independence are cryptographic tests which measures 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 relative large number of bytes. The values of Avalanche, Strict Avalanche and Bit Independence closer to 1.0 indicate the high degree of security. Table 1 shows the Avalanche, Strict Avalanche and Bit Independence values for all three techniques. For maximum files, all three measures using TDES, AES and FRSM are very near equal to 1. This analysis may indicate that FRSM provides very good security.

Table 1 Avalanche and strict avalanche values for TDES, AES and FRSM

Sl. No	File type	Avalanche values			Strict avalanche values			Bit independence values		
		TDES	AES	FRSM	TDES	AES	FRSM	TDES	AES	FRSM
1	Txt	0.990	0.993	0.250	0.903	0.926	0.183	0.156	0.260	0.018
2	Zip	0.986	0.998	0.902	0.944	0.972	0.868	0.394	0.357	0.688
3	Txt	0.996	0.994	0.978	0.989	0.989	0.967	0.412	0.399	0.428
4	Txt	0.999	0.999	0.968	0.997	0.999	0.953	0.480	0.484	0.498
5	Jpg	0.999	0.999	0.992	0.999	0.999	0.990	0.971	0.975	0.983
6	docx	0.999	0.999	0.999	0.999	0.999	0.998	0.976	0.972	0.988
7	Exe	0.999	0.999	0.949	0.999	0.999	0.942	0.634	0.610	0.676
8	Png	0.999	0.999	0.998	0.999	0.999	0.998	0.990	0.991	0.991
9	Rar	0.999	0.999	0.973	0.999	0.999	0.968	0.998	0.997	0.983
10	Dll	0.999	0.999	0.990	0.999	0.999	0.989	0.753	0.755	0.751
11	Exe	0.999	0.999	0.949	0.999	0.999	0.942	0.746	0.740	0.752
12	docx	0.999	0.999	0.997	0.999	0.999	0.997	0.991	0.991	0.990
13	Dll	0.999	0.999	0.972	0.999	0.999	0.970	0.725	0.730	0.791
14	Jpg	0.999	0.999	0.971	0.999	0.999	0.968	0.995	0.995	0.976
15	Pdf	0.999	0.999	0.993	0.999	0.999	0.992	0.975	0.963	0.987
16	Avi	0.999	0.999	0.974	0.999	0.999	0.972	0.993	0.992	0.979
17	Rtf	0.999	0.999	0.991	0.999	0.999	0.986	0.374	0.339	0.356
18	Doc	0.999	0.999	0.996	0.999	0.998	0.993	0.340	0.221	0.415
19	Rar	0.999	0.999	0.995	0.999	0.999	0.994	0.999	0.999	0.996
20	Avi	0.999	0.999	0.768	0.999	0.999	0.737	0.988	0.988	0.631

4.3 Chi-square Values for Non-homogeneity Test

The large Chi-square values (compared with tabulated values) may confirm the high degree of non-homogeneity between the source and the encrypted streams. Table 2 shows the Chi-square values using Triple-DES (168 bits), AES (128 bits) and proposed FRSM. Average Chi-square values of all twenty files using TDES, AES and FRSM are 34,143,114,516, 32,603,653,699 and 33,190,935,529 respectively. It may indicates the security of FRSM be good and comparable with that of TDES and AES.

Table 2 Chi-square values using TDES, AES and FRSM

Sl. No.	File type	File size (Bytes)	Chi-square values		
			TDES	AES	FRSM
1	Txt	50	114	111	141
2	Zip	288	503	529	542
3	Txt	500	1,470	1,546	1,837
4	Txt	2,410	24,059	20,981	51,264
5	Jpg	5,400	936	946	764
6	Docx	12,660	18,333	9,343	1,044
7	Exe	21,492	1,044,334	481,174	61,421
8	Png	50,952	6,085	6,086	4,679
9	Rar	115,595	1,031	1,038	806
10	Dll	215,416	530,985	473,027	275,729
11	Exe	624,128	2,027,106	1,848,171	1,348,006
12	Docx	1,224,413	54,964	55,574	43,742
13	Dll	1,409,024	3,219,751	3,139,562	15,85,202
14	Jpg	3,588,725	78,928	79,292	67,299
15	Pdf	4,446,250	413,610	369,563	652,379
16	Avi	7,355,928	438,208	442,887	225,162
17	Rtf	15,766,836	682,549,634,194	651,782,727,380	663,551,614,764
18	Doc	43,456,000	288,821,670	267,709,342	255,293,527
19	Rar	77,246,022	61,298	61,037	5,265
20	Avi	144,161,826	15,912,744	15,646,387	7,477,006
Average Chi-square values			34,143,114,516	32,603,653,699	33,190,935,529

4.4 Character Frequencies Test

Analysis of Character frequencies for text file (serial number of file is 4) has been performed for TDES, AES and proposed FRSM. Figure 4a shows the spectrum of frequency distribution of characters in the plain text. Figure 4b–d show the spectrums of frequency distribution of encrypted characters in cipher text using TDES, AES and FRSM. In encrypted files, all characters with ASCII values ranging from 0 to 255 appear with certain frequencies. Therefore it is very difficult to detect the actual message for a cryptanalyst. It may indicate that the degree of security of proposed FRSM is very high and is comparable with that of TDES and AES.

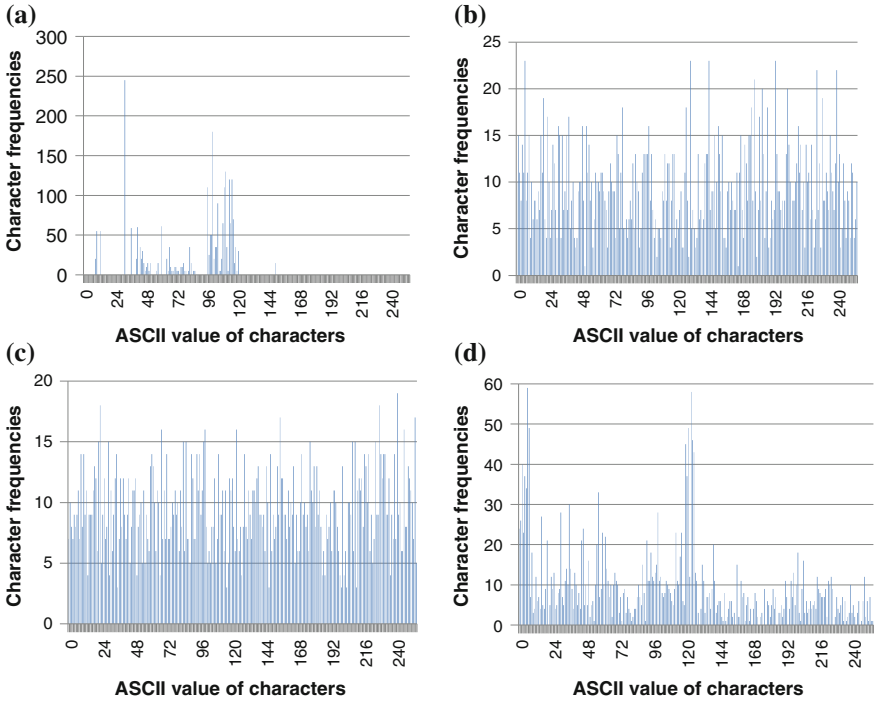

Fig. 4 **a** Graphical representation of the spectrum of frequency distribution of characters for source file. **b** Graphical representation of the spectrum of frequency distribution of characters for encrypted file using TDES. **c** Graphical representation of the spectrum of frequency distribution of characters for encrypted file using AES. **d** Graphical representation of the spectrum of frequency distribution of characters for encrypted file using FRSM

5 Conclusion

The proposed FRSM is simple to understand and easy to implement using various high level languages. The performance of FRSM is quite satisfactory because of high processing speed and the degree of security of FRSM may at par with other standard techniques like Triple DES and AES. So the proposed FRSM technique is comparable with the standard available encryption techniques and it is applicable to ensure good security in message transmission of any form through communication network.

References

1. Navin, A.H., Oskuei, A.R., Khashandarag, A.S., Mirnia, M.: A novel approach cryptography by using residue number system. In: 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT 2011), pp. 636–639. Seogwipo, 29 Nov–01 Dec 2011
2. Niemiec, M., Machowski, L.: A new symmetric block cipher based on key-dependent S-Boxes. In: 4th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT 2012), pp. 474–478. St. Petersburg, 3–5 Oct 2012
3. Verma, H.K., Singh, R.K.: Enhancement of RC6 block cipher algorithm and comparison with RC5 & RC6. In: 3rd IEEE International Advance Computing Conference (IACC 2013), pp. 556–561. Ghaziabad, 22–23 Feb 2013
4. Mandal, B.K., Bhattacharyya D., Bandyopadhyay, S.K.: Designing and performance analysis of a proposed symmetric cryptography algorithm. In: International Conference on Communication Systems and Network Technologies (CSNT 2013), pp. 453–461. Gwalior, 6–8 Apr 2013
5. Paul, M., Mandal, J.K.: A novel generic session based bit level cryptographic technique based on magic square concepts. In: International Conference on Global Innovations in Technology and Sciences (ICGITS 2013), pp. 156–163. SAINTGITS College of Engineering, Kottayam, 4–6 Apr 2013