RESEARCH ARTICLE OPEN ACCESS

MATLAB Implementation of 128-key length SAFER+ Cipher System

*Musaria K. Mahmood, **Lujain S. Abdulla, **Ahmed H. Mohsin, and **Hamza A. Abdullah

**Department of Communication and Electronics Engineering, Philadelphia University, Amman, Jordan*
*** Department of Electrical Engineering, University of Tikrit, Tikrit, Iraq*

ABSTRACT
Data security is a major challenge today. To protect data in the Internet or in private networks many measures exist. The most important security layer is the use of encryption standard to protect information from eavesdropper. Today many encryption standards exist failing in two categories: symmetric and asymmetric key algorithms. In this work Secure And Fast Encryption Routine (SAFER+) standard is implemented using MATLAB. The development in computer programming techniques and languages offers a good opportunity of the software implementation of encryption standards with moderate cost and good performance. MATLAB can be considered today as the first engineering programming language with powerful mathematical function and reliable programming procedures.

***Keywords*:** MATLAB, SAFER+, Symmetric key, software implementation

I. INTRODUCTION

The word today enters the era of data gathering, classification, sharing and then protection especially by the exponential increase of the use of computer networks and mobile system. Data protection becomes practically the important challenges to be faced. Cryptography is the most important countermeasure for securing data in general. SAFER+ is a symmetric key standard where the same key is shared by a sender and a receiver. It was one of the candidates for the last round for the Advanced Encryption standard (AES). Designed by James Massie (Messey) in Cylink in 1998, it is the new algorithm of the family SAFER (SAFER-64 SAFER-128) [1]. It is one of the symmetric key algorithms known by its speed of data encryption that made it capable to be used for real-time data encryption requirements [2]. SAFER+ presents a good hardware-software tradeoff orientation, simplicity, high throughput compared to other algorithms, and low memory requirement [3], [4]. This encryption standard carries attention by its use as security measure in Bluetooth and wireless communication [5]. Hardware implementation of SAFER+ encryption algorithm was presented in the beginning of the deployment of the encryption standard which was proven to be very efficient [6].

Bluetooth communication provides a short-run distance wireless communication between devices and other network with low cost and low power making it important part of the modern world where it [7]. SAFER+ is the most outstanding encryption standard in Bluetooth security architecture. A confrontation of SAFER+ standard with existing encryption algorithms proves the superiority of this algorithm in VLSI implementation in Bluetooth devices [8], [9] [10]. The technical values of SAFER+ are not limited to the hardware implementation and to the Bluetooth communication protection, but can be spreading to others areas [2]. In [11], a software evaluation of SAFER+ is implemented where its performances are evaluated based on the efficiency of the algorithm. In this paper a software platform is developed based on the implementation of SAFER+ cipher system on MATLAB with 128-bits key length. This platform can be used for data encryption from personal use to small institution. Results of the implementation show good performances in encryption of data in general. An innovative programming procedure is presented for the implementation of nonlinear functions which are the most difficult step of programming implementation. This paper is organized as follows. The module structures of SAFER+ are described in Section II. The software implementation of the algorithm and results discussion is presented in Section III. Finally in section IV a conclusion of the work is given.

II. SAFER+ ENCRYPTION AND DECRYPTION PROCEDURES

The encryption system is composed by three different modules: the encryption, decryption and the keys schedules modules. Each module is a combination of several logic functions with two non-linear functions based on logarithm and exponential computation.

2.1 SAFER+ ENCRYPTION METHOD

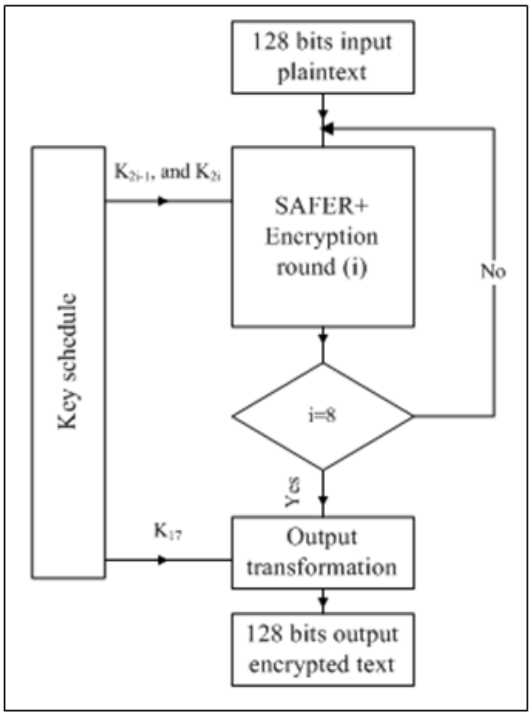
SAFER+ is a block cipher system which encrypts (decrypts) a block of 128 bits data at a time, using a key of 128 bits length (can also be of length 192 or 256 bytes). Input data (Plain text) is subject to a number of encryption rounds (R) giving encrypted output data. R is related to the keys length as:

R = 8 rounds if key length =128 bits (16 bytes),
R = 12 rounds if key length =192 bits (24 bytes),
R = 16 rounds if key length =256 bits (32 bytes).

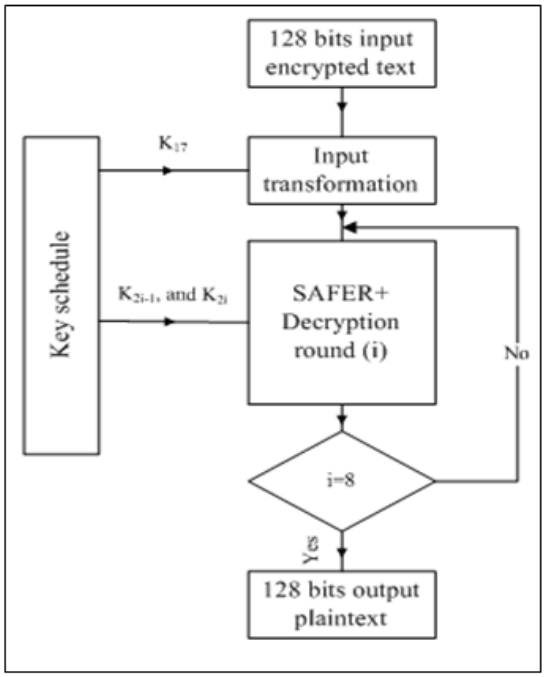
For the three cases the input plaintext is grouped into 128 bits blocks. In this work the cipher system with 128 bits key length is implemented because other options (192 or 256 bits key length) are found with certain key feebleness [12]. Fig. 1 presents the principals of SAFER+ algorithm mechanism.

2.2 KEYS SCHEDULE

Algorithm users own the 128 bits (16-bytes) principal key (K_1), from what a number of (16) other keys ($K_2 \dots K_{17}$) are generated using key schedule module with 128 bits length for each key. Two keys are used for each round and the last key (K_{17}) is used for the output transformer.



(a) Encryption



(b) Decryption

Figure 1: Structure of SAFER+ standards

The key schedule module uses sample arithmetic and logic functions like bit rotation, bit-by-bit exclusive-or of bytes, modulo 256 addition of bytes, and byte selection process.

The key schedule for the 16 bytes input key is exposed in Fig. 2. The 16 sub keys ($K_2 \dots K_{17}$) are computed in the following manner. The user secret key itself is used as the first sub key (K_1) and is also loaded into the first 16 byte positions of a 17-byte key register (KB_1 to KB_{16}). The last byte position of this register (KB_{17}) is then loaded with the bit-by-bit modulo two, sum of the 16 bytes of the user selected key. Each byte of the key register is then rotated leftwards by three bits positions. The second sub key K_2 is then calculated as modulo 256 -sum of the bytes of 16-byte bias word (B_2) with the bytes number KB_2, KB_3…, and KB_{17} respectively. Each byte of the key register is then again rotated leftwards by 3 bit positions. K_3 is calculated as the sum modulo 256 of the 16-byte bias word (B_3) with the bytes number KB_3, KB_4,....KB_{17}, and KB_1 respectively. *B* matrix is used by SAFER+ standard to randomize the round sub keys [1]. The first bias word (B_1) is a "dummy" never used but it is defined for programming purpose. Bi denotes the i^{th} bias word and let $B_{i,j}$ denote the j^{th} byte of this i^{th} bias word computed as follows:

$$B_{i,j} = 45^{(45^{17i+j} \bmod 257)} \bmod 257 \quad (1)$$

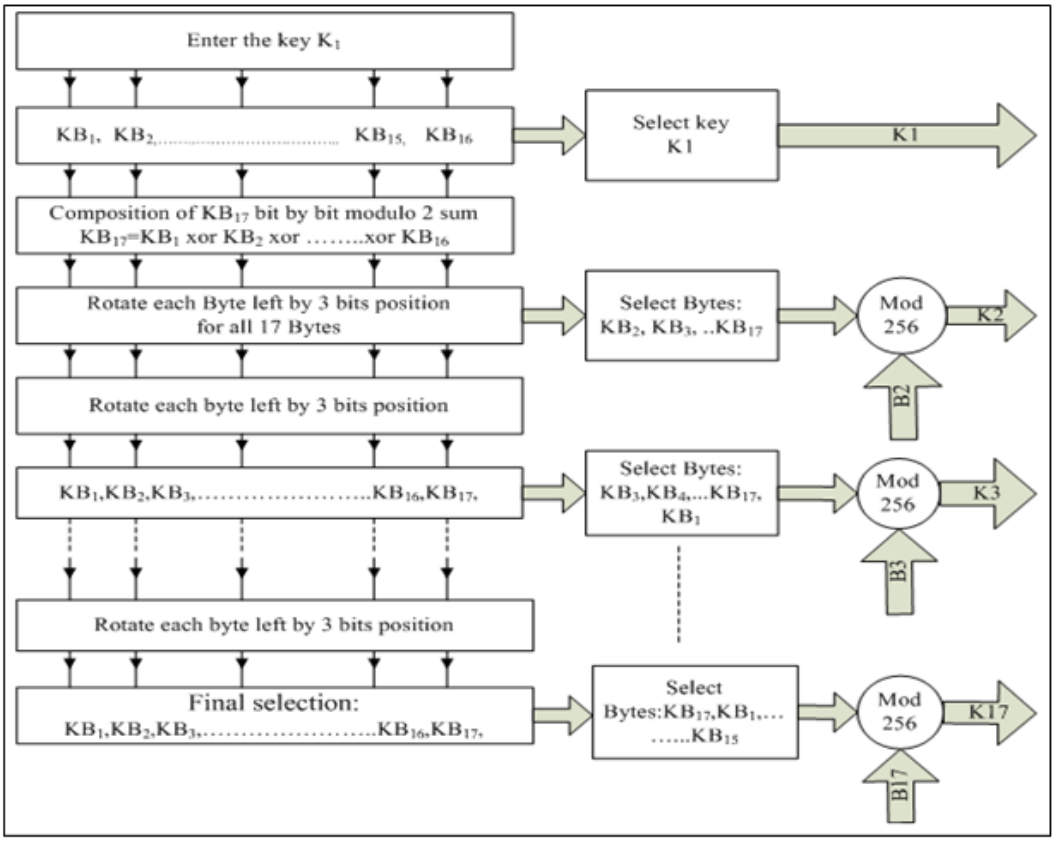


Figure 2: sub key production

2.3 ENCRYPTION PROCEDURE

SAFER+ is a symmetric block cipher standard. Fig. 3 shows SAFER+ encryption structure where the input is the plaintext block of 16-bytes ($PT_1 \ldots PT_{16}$). 17 Sub key produced by the key schedule (each sub key of 16 bytes length) are used in the encryption rounds as two keys for each round. Round (*i*) uses key K_{2i-1} and K_{2i}. K_{17} is used for the output transformation as bit-by-bit exclusive-or and modulo 256 byte addition. The 16-byte input data of round (*i*) ($RT_1 \ldots RT_{16}$) are bit-by-bit XOR, and modulo 256 addition of bytes with the key K_{2i-1}. Bytes 1, 4, 5, 8, 9, 12, 13, and 16 of round input data and round sub-key K_{2i-1} are added together bit-by-bit modulo two. Other bytes are added together modulo 256.

The results are then fed to a nonlinear layer. The value (*x*) of byte (*j*) is converted to (45^x mod 257) for j = 1, 4, 5, 8, 9, 12, 13, and 16 (with the Convention that when x = 128, then 45^{128} mod 257 = 256 is represented by a 0). The value (*x*) of byte (*j*) is converted to $\log_{45}(x)$ for j = 2, 3, 6, 7, 10, 11, 14, and 15 (with convention that when x = 0, then $\log_{45}(0) = 128$). The output of the nonlinear layer is then subjected to the same addition and XOR operation similar to the first block with key K_{2i} but with opposite order as shown in figure 4. At the end of round (*i*), matrix multiplication is used. The 16 bytes are multiplied by matrix *(M)* in mod 256 arithmetic. *M* is a 16 × 16 predefined matrix given as fixed input. *M* is used in the encryption process while M^{-1} is used in the decryption process. At the end of round (8) the message is processed by the output transformer to give the 16-bytes ($CT_1 \ldots CT_{16}$) cipher text as shown at the bottom of the fig. 3.

$$M=\begin{bmatrix} 2 & 2 & 1 & 1 & 16 & 8 & 2 & 1 & 4 & 2 & 4 & 2 & 1 & 1 & 4 & 4 \\ 1 & 1 & 1 & 1 & 8 & 4 & 2 & 1 & 2 & 1 & 4 & 2 & 1 & 1 & 2 & 2 \\ 1 & 1 & 4 & 4 & 2 & 1 & 4 & 2 & 4 & 2 & 16 & 8 & 2 & 2 & 1 & 1 \\ 1 & 1 & 2 & 2 & 2 & 1 & 2 & 1 & 4 & 2 & 8 & 4 & 1 & 1 & 1 & 1 \\ 4 & 4 & 2 & 1 & 4 & 2 & 4 & 2 & 16 & 8 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 2 & 1 & 2 & 1 & 4 & 2 & 8 & 4 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 4 & 2 & 4 & 2 & 16 & 8 & 2 & 1 & 2 & 2 & 4 & 4 & 1 & 1 \\ 1 & 1 & 2 & 1 & 4 & 2 & 8 & 4 & 2 & 1 & 1 & 1 & 2 & 2 & 1 & 1 \\ 2 & 1 & 16 & 8 & 1 & 1 & 2 & 2 & 1 & 1 & 4 & 4 & 4 & 2 & 4 & 2 \\ 2 & 1 & 8 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 4 & 2 & 4 & 2 & 4 & 4 & 1 & 1 & 2 & 2 & 1 & 1 & 16 & 8 & 2 & 1 \\ 2 & 1 & 4 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 8 & 4 & 2 & 1 \\ 4 & 2 & 2 & 2 & 1 & 1 & 4 & 4 & 1 & 1 & 4 & 2 & 2 & 1 & 16 & 8 \\ 4 & 2 & 1 & 1 & 1 & 1 & 2 & 2 & 1 & 1 & 2 & 1 & 2 & 1 & 8 & 4 \\ 16 & 8 & 1 & 1 & 2 & 2 & 1 & 1 & 4 & 4 & 2 & 1 & 4 & 2 & 4 & 2 \\ 8 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 1 & 2 & 1 & 4 & 2 \end{bmatrix} \quad (2)$$

$$M^{-1}=\begin{bmatrix} 2 & -2 & 1 & -2 & 1 & -1 & 4 & -8 & 2 & -4 & 1 & -1 & 1 & -2 & 1 & -1 \\ -4 & 4 & -2 & 4 & -2 & 2 & -8 & 16 & -2 & 4 & -1 & 1 & -1 & 2 & -1 & 1 \\ 1 & -2 & 1 & -1 & 2 & -4 & 1 & -1 & 1 & -1 & 1 & -2 & 2 & -2 & 4 & -8 \\ -2 & 4 & -2 & 2 & -2 & 4 & -1 & 1 & -1 & 1 & -1 & 2 & -4 & 4 & -8 & 16 \\ 1 & -1 & 2 & -4 & 1 & -1 & 1 & -2 & 1 & -2 & 1 & -1 & 4 & -8 & 2 & -2 \\ -1 & 1 & -2 & 4 & -1 & 1 & -1 & 2 & -2 & 4 & -8 & 1 & -8 & 16 & -4 & 4 \\ 2 & -4 & 1 & -1 & 1 & -2 & 1 & -1 & 2 & -2 & 4 & -8 & 1 & -1 & 1 & -2 \\ -2 & 4 & -1 & 1 & -1 & 2 & -1 & 1 & -4 & 4 & -8 & 16 & -2 & 2 & -2 & 4 \\ 1 & -1 & 1 & -2 & 1 & -1 & 2 & -4 & 4 & -8 & 2 & -2 & 1 & -2 & 1 & -1 \\ -1 & 1 & -1 & 2 & -1 & 1 & -2 & 4 & -8 & 16 & -4 & 4 & -2 & 4 & -2 & 2 \\ 1 & -2 & 1 & -1 & 4 & -8 & 2 & -2 & 1 & -1 & 1 & -2 & 1 & -1 & 2 & -4 \\ -1 & 2 & -1 & 1 & -8 & 16 & -4 & 4 & -2 & 2 & -2 & 4 & -1 & 1 & -2 & 4 \\ 4 & -8 & 2 & -2 & 1 & -2 & 1 & -1 & 1 & -2 & 1 & -1 & 2 & -4 & 1 & -1 \\ -8 & 16 & -4 & 4 & -2 & 4 & -2 & 2 & -1 & 2 & -1 & 1 & -2 & 4 & -1 & 1 \\ 1 & -1 & 4 & -8 & 2 & -2 & 1 & -2 & 1 & -1 & 2 & -4 & 1 & -1 & 1 & -2 \\ -2 & 2 & -8 & 16 & -4 & 4 & -2 & 4 & -1 & 1 & -2 & 4 & -1 & 1 & -1 & 2 \end{bmatrix} \quad (3)$$

2.4 DECRYPTION PROCEDURES

The decryption module has the reverse process that used in encryption module. The encrypted data is injected to the input transformation where key K_{17} is used. The input transformation uses the same functions of the output transformation but with modulo 256 subtraction of bytes instead of bytes addition as in Fig. 3. Decryption round, mathematical functions are simply the inverse operations from the encryption round. The first operation is a multiplication of input data (here encrypted data) by the matrix M^{-1}which is modulo 256 inverse of *M*. The round sub-key (K_{18-2i}) is then "subtracted" from the input round cipher text (*x*) in the manner that the round sub key bytes 1, 4, 5, 8, 9, 12, 13, and 16 are subtracted modulo 256 from the corresponding bytes of (*x*) while round sub key bytes 2, 3, 6, 7, 10, 11, 14, and 15 are added bit-by bit modulo 2 to the corresponding bytes of (*x*). The resulting bytes are directed to the non linear stage with $\log_{45}(x)$ and 45^x as in the encryption process but with reverse order of log and exponential. The round sub-key (K_{17-2i}) is then subtracted from the 16-bytes result in the reverse manner of the previous subtraction stage.

III. MATLAB IMPLEMENTATION AND RESULTS

The algorithm implementation is achieved; many case studies are used to evaluate the performance of the encryption routine. SAFER+ is a block symmetric key encryption algorithm which can encrypt (decrypt) data of any kinds as images, video, numerical data, or speech. The fact that the algorithm is proved to be a fast routine presents the possibility to encrypt data in real time. The software implementation of SAFER+ requires a high performances computer for running the algorithm in real time conditions.

Encryption, Decryption, and sub-keys production modules have to calculate some values many times during the program execution. To avoid the repetition at every step, calculations are done once; results are stored, called when needed in the program. These sub-programs are conversion of binary to decimal, decimal to binary, logarithmic function, exponential function, and bias matrix (B).

3.1 Implementation of Logic Functions

MATLAB is a powerful language, possesses a large library of logic and mathematical function. Functions used to implement SAFER+ are:

- Add: P+K
- Sub: C-K
- Modulo:
- mod(x,y)
- Matrix multiplication: PT*M
- Xor: xor(p(i), k(i))
- Rotate: R(:,1:5)=B(:,4:8); R(:,6:8)=B(:,1:3);

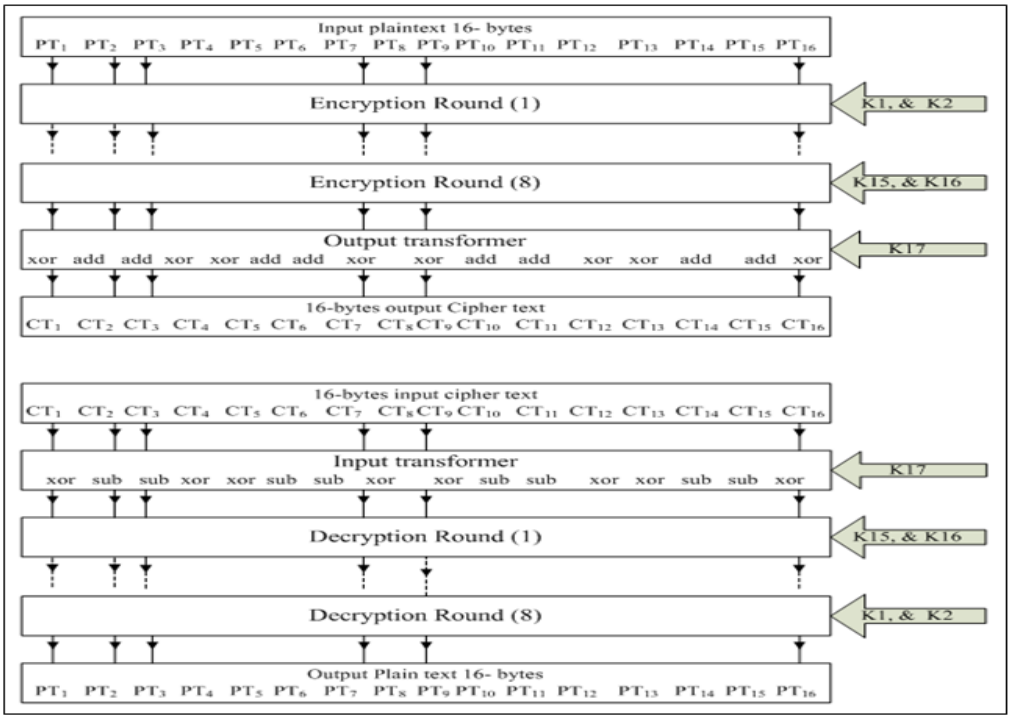


Figure 3: Encrypting and Decrypting structure of the SAFER+ algorithm [1]

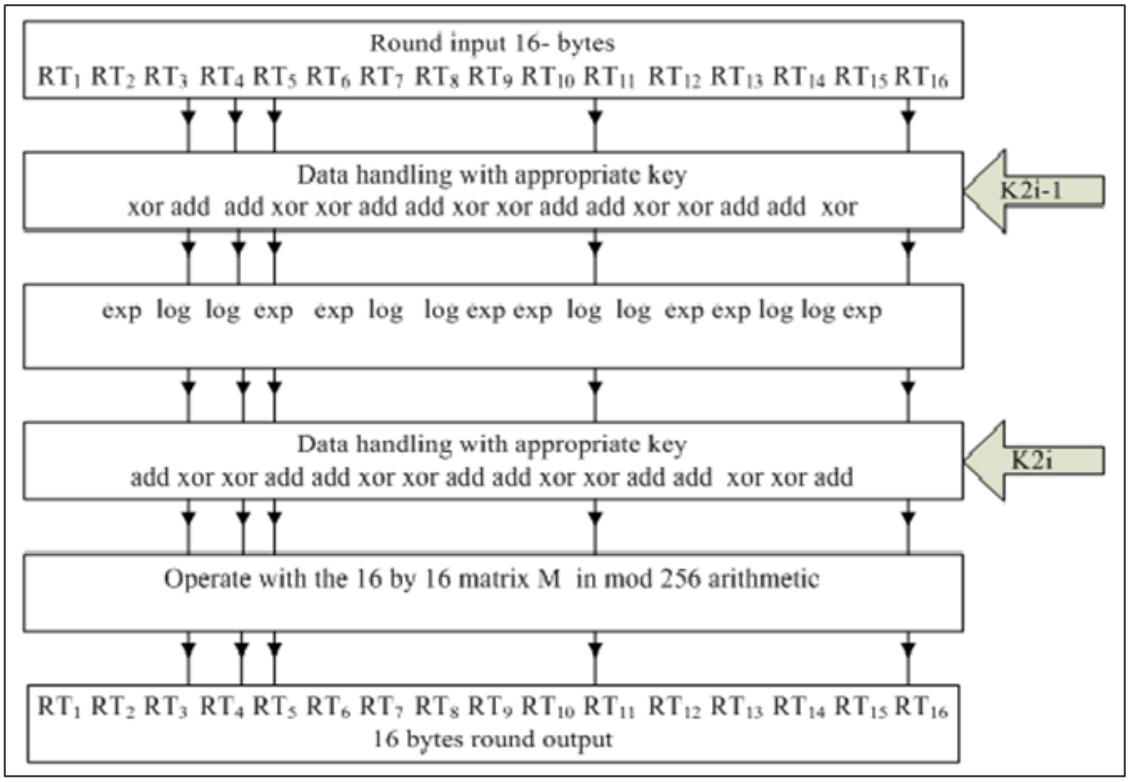


Figure 4: Encrypting round structure

3.2 Logarithmic and Exponential Functions

The most difficult programming step in SAFER+ is the implementation of nonlinear functions which are here the exponential and logarithmic functions. These functions are defined as:

Exponential function:

$$f(x) = \begin{cases} 45^x \bmod 257 & \text{if } x \neq 128 \\ 0 & \text{if } x = 128 \end{cases} \quad (4)$$

Logarithmic function:

$$g(x) = \begin{cases} \log_{45}(x) & \text{if } x \neq 0 \\ 128 & \text{if } x = 0 \end{cases} \quad (5)$$

The difficulty of implementing exponential function comes from the fact that 45^x can be a huge number impossible for MATLAB and other programming tools to handle it. The power of the number 45 gives a very large value for ($x > 9$) where the output cannot calculated directly. A sub program is developed in this work to overcome the limitation of MATLAB to handle big numbers as presented in figure (5). Three cases are present in this procedure:

1- if $x = 128$, then $f(x) = 0$
2- if $x \leq 8$, then $f(x)$ is computed directly using (4)
3- if $x > 8$ put $x = n \times 8 + r$,

$45^8 \bmod 257 = 120$, then

$$f(x) = [\{(120^n) mod 257\} \times 45^r] mod\ 257$$

Logarithmic function given (5) is the inverse function of the exponential given in (4). The above is used to calculate $\log_{45}(x)$, for $0 \leq x \leq 255$ one time, stored in a one row matrix to be used by the program.

3.3 Image Encryption/Decryption

The implemented algorithm can be used for images encryption by following these steps:

- Read and convert the image into a three-dimensional matrix with each element value ranging from 0 to 255.

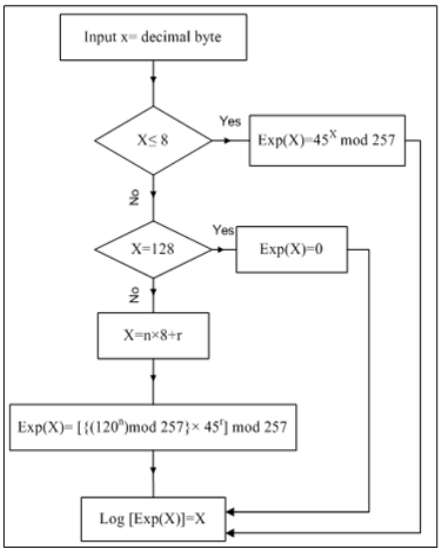


Figure 5: exponential and logarithmic functions

- Convert the matrix into a serial flow of numbers given as one row.
- Add a number of redundant elements with fixed value to get a total elements number that can be divided by (16) which is encrypted block size. The added elements will be removed after decryption and before the regeneration of the original images.
- Convert data into two- dimensional matrix with number of columns equal to (16). One row of (16) elements is encrypted at a time presenting one block.
- At the receiver side the reverse operations are implemented to regenerate the original messages.

3.4 Results of Implemented Algorithm

Running the implemented program requires a principal 16-byte key. Input data can be blocks of 16-byte blocks or primitive data such as an image. Each byte is given as a number between 0 and 255 and can be coded on 8- bits binary word.

The key: 41-35-190-132-225-108-214-174-82-144-73-241-241-187-233-235.

Plaintext: 179-166-219-60-135-12-62-153-36- 94-13-28 6-183-71-222

After running the program a cipher text of 16-byte is produced:

Cipher text block is: 224-31-182-10-12-255-84-70-127-13-89-249-9 -57-165-220

At the receiver side the original message is regenerated by the decryption and key schedule modules and using the same principal key:

Regenerated message: 179-166-219-60-135-12-62-153-36- 94-13-28 6-183-71-222.

In the case of image or any other primitive data, first data must be grouped into 16-byte blocks as explained in (3.3), and then presented at the encryption module input. Fig. 6-a presents image (A) to be encrypted. Fig. 6-b, the same image after encryption is represented, and looks as a pure undefined noise. Fig. 6-c the original image after the application of decryption module with the appropriate principal key is regenerated. If another key is used or just on bit error in the principal key the image cannot be regenerated and still appearing as noise as in fig. 6-d.

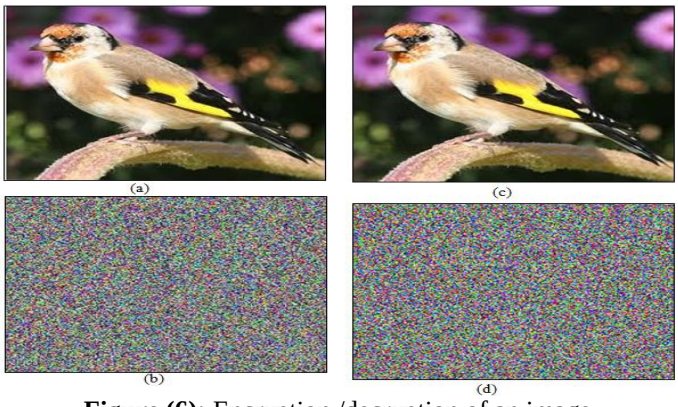


Figure (6): Encryption /decryption of an image

IV. CONCLUSION

SAFER+ presents an encryption decryption algorithm with good hardware software implementation. An encryption decryption platform is implemented by SAFER+ using MATLAB. This platform can be used for data encryption from personal use or for small institution with insignificant cost. Results of the implementation show a good performance in encryption of pictures and data in general. The speed of SAFER+ routine presents an opportunity to use it for the encryption of all type of data.

ACKNOWLEDGEMENTS

This research was supported by the Institute of International Education- SRF. We are thankful for their help.

REFERENCES

[1] J. Messy, H. Kh and K. Kuregian, Nomination of SAFER+ as a candidate algorithm for the AES, 1998. http://csrc.nist.gov/archive/aes/round1.

[2] Musaria K. Mahmood, and Fawzi M. Al-Naima, Developing a multi-layer strategy for securing control systems of oil refineries, *wireless Sensor Network, 2, 2010, 520-527.*

[3] A. Schubert, and W. Anheier, Efficient VLSI implementation of modern symmetric block ciphers, *Proc. 6th IEEE International Conf. on Electronics, Circuits and Systems, Cyprus Sep. 1999, Vol. 2, 757 – 760.*

[4] Swarnendu M., Debashis G., and Somnath N., A New Generation Cryptographic Technique, *International Journal of Computer Theory and Engineering, 1(3), 2009, 284-287.*

[5] I. S. Ashour, Online Data and Voice Encryption System Based on FPGA, *Proc. 24th National Radio Science Conf., Cairo, National Republican Senatorial Committee, 2007, 1-7.*

[6] P. Kitsos; N. Sklavos; O. Koufopavlou, Hardware implementation of the SAFER+ encryption algorithm for the Bluetooth system, *Proc. IEEE International Symposium on Circuits and Systems, 2002, Vol. 4, 878 -881.*

[7] U.L. Muhammed, S. Mosharani, S. Amuthapriya, M.M. Mufthas, M. Hezretov, and D. Dhammearatchi, Bluetooth security analysis and solution, *International Journal of scientific and research publication, 6(4), 2016, 333-338.*

[8] B. J. Babu, D. Kishore, and R. V. V. Krishna, Design of SAFER+ encryption algorithm for Bluetooth transmission, *International Journal of innovative technology and research, 3(1), 2015, 1864-1867.*

[9] B. Manthan A., and A. S. Shingh, Multilevel security algorithm for Bluetooth technology, *International Journal for Research in Technological Studies, 1(1), 2013, 1-7.*

[10] V. P. Babu, B. Sreenivas, T. P. Kumar, and R. J. Lai, Cracking Bluetooth security, *International Journal of applied sciences and Engineering Research, 3(2), 2014, 540-545.*

[11] D. Sharmila and R. Neelaveni, Performance Analysis of SAFER+ and Triple DES Security Algorithms for Bluetooth Security System, *International Journal of Computer Science and Network Security, 9(2), 2009, 74-87.*

[12] J. Kelsey, B. Schneier and D. Wagner, Key Schedule Weaknesses in SAFER+, *Proc. 2ndAdvanced Encryption Standard Candidate Conf., Rome, 1999, 155- 167.*