Encrypt Medical Image using CSalsa20 Stream Algorithm

Kian Raheem Qasim[1], Sara Salman Qasim[2]

[1]Lecturer, University of Information Technology and Communications, [2]Assist. Lecturer, University of Middle Technical - Mansour Technical Medical Institute - Computer & Internet Center. Sarah

Abstract

As a result of the tremendous development in the field of information technology and the internet and exposure to a flood of violations to circumvent and steal information organized and unorganized. The urgent need for the emergence of data protection technologies and data encryption techniques. As a result of the tremendous development in the field of information technology and the internet and exposure to a flood of violations to circumvent and steal information organized and unorganized. The urgent need for the emergence of data protection technologies and data encryption techniques one of these methods, The Salsa20 encryption stream and all of its reduced versions Salsa20 / 7 and Salsa20 / 12 are among the fastest stream ciphers today. In this paper, the Salsa20 method is therefore improved by adding a new variable by using chaos theory which can achieve faster propagation than the original Salsa20 and has been applied by encrypting medical images that need confidentiality because some patients do not want anyone to know about a disease so the patient's medical data is encrypted and no one can access it. This method has been tested and measured with the original Salsa20 with a series of tests. Most tests show that the proposed messy salsa is faster than the original salsa.

Keywords— *salsa20, medical image, chaotic map.*

Introduction

In clinical computer systems, medical images are critical and sensitive information [1]. To transfer medical images through a nonsecurity network, a reliable encryption algorithm must be developed. Among the three main features of security services (CIA),the most important characteristics of the exchange between doctors of medical images are confidentiality [2]. It is important to protect the confidentiality of image data from unauthorized access in conjunction with the rapid development of electronic data exchange [3]. Violations of security can affect the privacy and reputation of users. Data encryption is therefore widely used in open networks such as the Internet to ensure security [4]. Due, to the, significant increase, in digital data transmission via public channels, digital image security has become more popular and attracting a great deal of attention in the modern digital world. Throughout our culture, the advancement of multimedia technology has enabled digital images to play a greater role than conventional documents, which require serious privacy protection for all applications [5]. Each data type has its own advantages, so different techniques must be used to protect confidential image data from unauthorized access. Most, available, encryption algorithms use text data. Nevertheless, it is not feasible to use traditional methods of encryption due to the large data volume and real-time requirements. The last major trend is therefore to reduce the computational requirements for stable multimedia distribution [6]. Several authors have suggested various image coding schemes to overcome problems with image coding [7]. A classification of the schemes proposed in open literature is provided in [8]. In this paper, we, scanned the image encryption application for Salsa 20. Salsa20 has an interesting complete structure that appears to be a good choice for encoding images. A series of tests were used to justify the visual coding efficiency of Salsa20

Salsa20 Algorithm

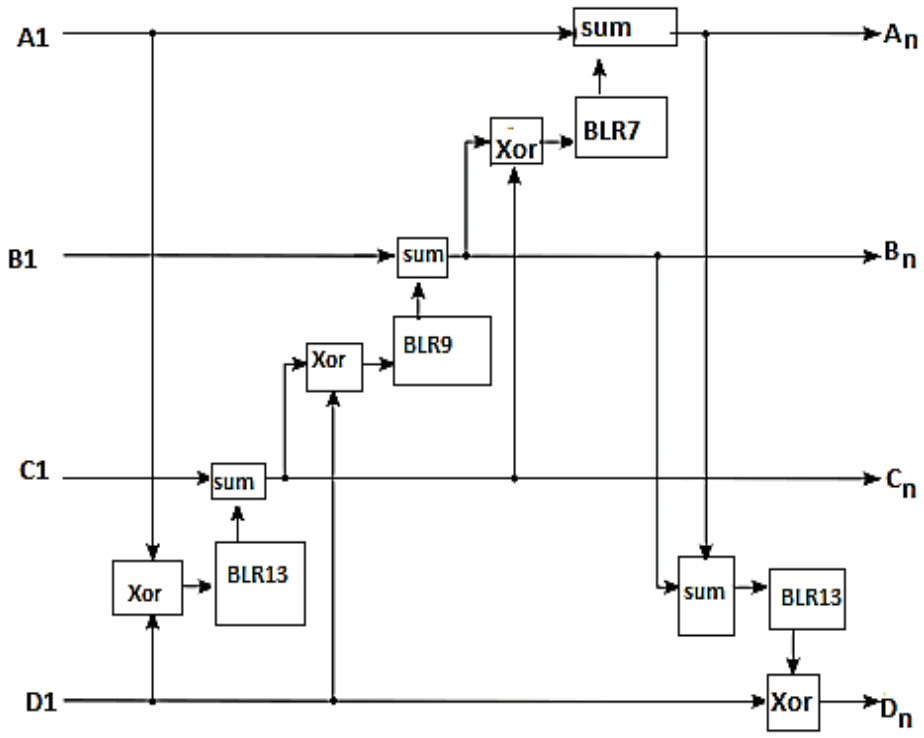

Figure (1) salsa20 work

Salsa 20 is a stream cipher designed depend on rotation operations 32-bit addition, and adding 32-bit bitwise (XOR). The salsa algorithm assigns block size 4 X4 to consist of a key (256-bit), nonce (64-bit), position (64-bit), and four output key constant from (32 to 512) bits. The Salsa20 core is a hash function for 64-byte input and 64-byte output that is stream cipher that works in counter mode. The producer of the 64-byte keystream through hash key, block, nonce, and the results are XOR operation with 64-byte plaintext. The Salsa20 hash function indicates a quarter function. In the figure (1), shown the method Salsa20 works, where you perform several operations that will be explained later. Where this method is implemented on the image and to implement it will need several stages first generating the key using the method of Salsa and encrypt the image through it.

Salsa20 operation

Salsa20 algorithm includes several processes the functions offered in order from lower-level to a higher level that is more complexity:

1.Summation between Two words

Add four byte words are indicated through of $a + b$ algorithm and divided result into a parameter of 2^{32} (thus, can be stored that the maximum value that in a single word). The summation of the words a and b is equal $((a + b) \bmod 2^{32})$. $((a + b) \bmod 2^{32})$. The result is a valid word of 4 bytes long.

2.Xor Operation between Two Words

Xor Operation is referred to for two words of 4 bytes in the description of the methods as *a XOR b*. To perform an XOR add-in, comparison *bit by bit* between two words, and the addition of XOR is executed for each pair. The consequence is an accepted word length of 4 bytes.

3.Binary Left Rotation

Left bit rotation of the word 4-byte wrd is indicated in the algorithm as ***wrd <<< at***. Move left-most to the positions rightmost. The rotation of the bits to the left can be represented by a word of 4 bytes in the form of multiplication: $((2^{at} \cdot wrd \bmod (2^{32}-1))$. The results are an accepted word 4-byte long.

4.Function of Quarter Round

In this method take four words for input and return other string of four words.

If v is a four words input:

$v = (v_0, v_1, v_2, v_3)$

The function can be defined as:

Quarter (v) = (u_0, u_1, u_2, u_3)

Where:

$u_1 = v_1$ Xor $((v_0 + v_3) <<< 7)$

$u_2 = v_2$ Xor $((u_1 + v_0) <<< 9)$

$u_3 = v_3$ Xor $((u_2 + u_1) <<< 13)$

$u_0 = v_0$ Xor $((u_3 + u_2) <<< 18)$

Quarter functionality may be implemented, without having to allocate any Extra memory. First, u1 alteration to v1, then u2 alteration to v2, the next u3 alteration to v3, and u0 alteration to v0. This method is reversed cause of the above adjustments are reversible.

5.Function of Row Round

This method takes input 16 words and convert it and return a 16-word string. The method similar to the column round function but doing different order of words.

If u input 16-word:

$u = (u_0, u_1, u_2, ..., u_{15})$

The method can be defined as:

Row round (u) = $(v_0, v_1, v_2, ..., v_{15})$

Where:

(v_0, v_1, v_2, v_3) = Quarterly (u_0, u_1, u_2, u_3)

(v_5, v_6, v_7, v_4) = Quarterly (u_5, u_6, u_7, u_4)

$(v_{10}, v_{11}, v_8, v_9)$ = Quarterly $(u_{10}, u_{11}, u_8, u_9)$

$(v_{15}, v_{12}, v_{13}, v_{14})$ = Quarterly $(u_{15}, u_{12}, u_{13}, u_{14})$

The square matrix displayed 16-word inputs:

u_0	u_1	u_2	u_3
u_4	u_5	u_6	u_7
u_8	u_9	u_{10}	u_{11}
u_{12}	u_{13}	u_{14}	u_{15}

in the array, the Rows may be alteration at the same time. Each of them is changed through the Quarter round function.

The first row, the word is alteration order as follow:

u_1, u_2, u_3, u_0

The second row, the word is alteration order as follow:

u_6, u_7, u_4, u_5

The third row, the word is adjusted in the order:

u_{11}, u_8, u_9, u_{10}

Finally, words are altered in order in the last and fourth rows:

$u_{12}, u_{13}, u_{14}, u_5$.

6.Function of Column round

This method takes input 16 words and returns the sequence of 16-word. The method similar to the Row round function but doing different order If u is the input of 16-word:

$u = (u_0, u_1, u_2, ..., u_{15})$

The method may be defined as follows:

Column (u) = $(u_0, u_1, u_2, ..., u_{15})$

Where:

(v_0, v_4, v_8, v_{12}) = Quarterly (u_0, u_4, u_8, u_{12})

(v_5, v_9, v_{13}, v_1) = Quarterly (u_5, u_9, u_{13}, u_1)

$(v_{10}, v_{14}, v_2, v_6)$ = Quarterly $(u_{10}, u_{14}, u_2, u_6)$

$(v_{15}, v_3, v_7, v_{11})$ = Quarterly $(u_{15}, u_3, u_7, u_{11})$

The square matrix displayed 16-word inputs:

u_0	u_1	u_2	u_3
u_4	u_5	u_6	u_7
u_8	u_9	u_{10}	u_{11}
u_{12}	u_{13}	u_{14}	u_{15}

in the array, Columns may be alteration at the same time. Each of them is changed through the Quarter round function.

The first column, the word is alteration order as follow:

u_4, u_8, u_{12}, u_0

The second column, the word is alteration order as follow:

u_9, u_{13}, u_1, and u_5

The third column, the word is alteration in order:

u_{14}, u_2, u_6, u_{10}

The last four columns, the word is alteration in order:

u_3, u_7, u_{11}, u_{15}

7.Function of Double Round

This method input 16 words and returns the sequence of 16-word.

If x is input of 16-word, then the Function is defined as follow:

double round (u) = row round (column round(u))

8.Function of Little Endian

This method can be changed the sequence order is 4 bytes.

If bt is 4byte sequence:

$bt = (bt_0, bt_1, bt_2, bt_3)$

then the method definition as:

$$\text{little endian (bt)} = bt_0 + 28bt_1 + 216bt_2 + 224bt_3$$

The function of little-endian is reversible. It simply modified the word order of bytes.

9.Function of Salsa20 Hash

The Function of Salsa20 Hash takes input 64 bytes and returns the sequence of a 64-byte.

If the input is 64byte sequence

input = $(bt_0, bt_1, bt_2, ..., bt_{63})$ Then 16 words are created

wd_0 = little endian(bt_0, bt_1, bt_2, bt_3)

wd_1 = little endian(bt_4, bt_5, bt_6, bt_7)

[...]

wd_{15} = little endian$(bt_{60}, bt_{61}, bt_{62}, bt_{63})$

$(u_0, u_1, ..., u_{15}) = \text{double round}^{10}(wd_0, wd_1, ..., wd_{15})$

$$\text{Output} = lE^{-1}(u0 + wd0) + lE^{-1}(u1 + wd1) + \ldots + lE^{-1}(u15 + wd15)$$

10.Function of Salsa20 Expansion

The Function of Salsa20 Expansion takes two bytes' sequences. The first sequence maybe 16 or 32 bytes and the second sequence (N) is 16 bytes long. The method returns a sequence other than 64 bytes. If 32-bytes long represent the first sequence, then divided 16 bytes of two shorter sequences (T_0 and T_1). The definition of function Salsa20 expansion by utilizing the function of the Salsa20 hash, as shown below:

Salsa20 Expansion T_0, T_1(N) = Salsa20 Hash (at_0, T_0, at_1, N, at_2, T_1, at_3)

where:

at_0 = (97,120,101,112), at_1 = (32,51,100, 110)

at_2 = (121, 98, 45, 50), at_3 = (101,107, 32,116)

If 16-bytes long (T) represent first sequence, then Function definition by utilizing the Function of Salsa20 Hash as below:

Salsa20 Expansion T(N) = Salsa20 Hash (bt_0, T, bt_1, N, bt_2, T, bt_3)

where:

bt_0 = (97,120,101,112), bt_1 = (32,49,100, 110)

bt_2 = (121, 98, 45, 54), bt_3 = (101,107, 32,116)

The vector of constant values (at_0, at_1, at_2, at_3) means32-byte T' expansion in ASCII code. Similarly, the second vector of constant values (bt_0, bt_1, bt_2, bt_3)

means 16-byte T' expansion in ASCII.

4. Chaotic Function and Logistic Map

Chaotic maps usually utilized in the investigation of dynamical frameworks [13]. Every now and again, Chaotic maps produce fractals. In spite of the fact that a rehashed procedure may develop the fractal, a few fractals are considered assets as opposed to regarding maps that produce them [14]. Thisais on theagrounds that there are many different iterative methods to generate a similar fractal. The logisticamap is a simple quadratic mathematical polynomial demonstrating complex chaoticabehavior. due to its simplicity, the logistic map remains useful as a testbed for new ideas in the theory of chaos and for the application of chaos in cryptography [15]. The basic form of the logisticamap can be mathematically interpreted, as shown in Eq. (1):

$$X_{n+1} = rx_n(1 - x_n)X_{n+1} = rx_n(1 - x_n) \quad \text{... (1)}$$

Where x is a stateavariableathat can, have any value between 0 and 1, and r is the system parameter that falls at the interval [1, 4]. Figure (2) shown the logistic map bifurcation diagram. The bifurcation variable r is shown on the plot's horizontal axis and the vertical axis displays the logistic function's possible long-term population values. Each of these bifurcation points is a bifurcation that doubles over time.

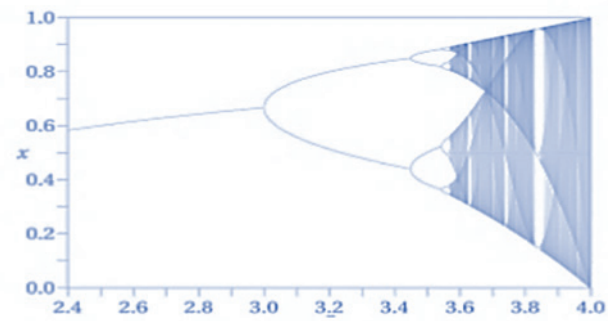

Figure (2) Bifurcation diagram for the logistic map

4. The Proposed Method of Csalsa20 algorithm

The proposed method consists of two main stages: - firstly, a random key is generated using the Salsa 20 algorithm with a logistic map secondly, encryption for the medical image after partition and return original image as shown in Figure (3).

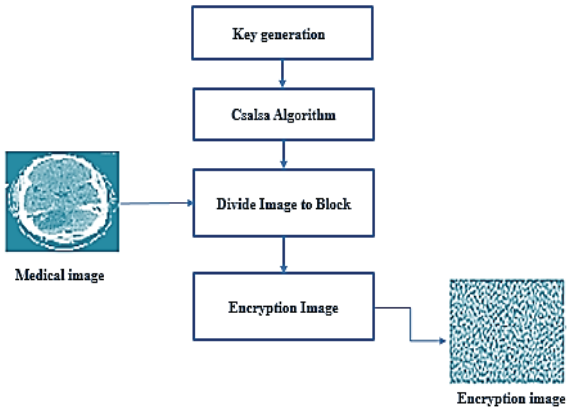

Figure (3): Block diagram of Proposed Extraction Method

A. Key generation CSalsa20 algorithm

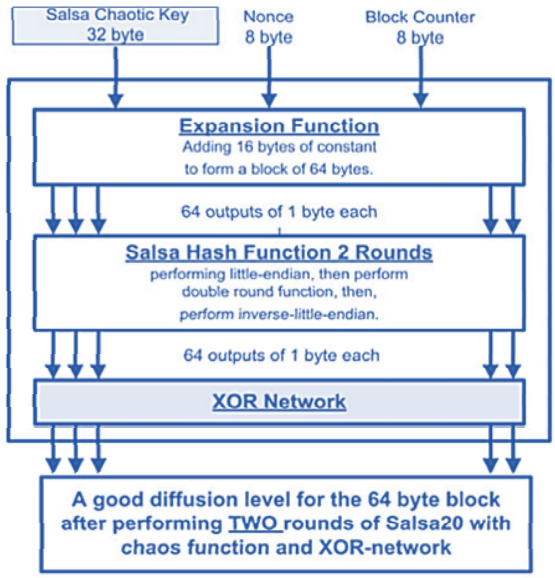

Figure (4) System block diagram

chaotic key generation can improve the resistance of Salsa20 against various attacks and help remove the statistical leakage that may be detected in the first rounds. The application of the chaotic logistics map to create the messy key will increase the input differences between any two successive periods from 1 bit to 33 bits because each block has four new bytes of clutter. These 33 new bits came from the 1-bit counter bit of the block, and the other new 32-bits come from the secret key's four chaotic bytes. Figure (5) illustrates how the chaotic key is produced. The first byte of the original key is the transformed integer within the [0, 255] range to the appropriate range for the logistic map, which falls within the [0, 1] range. The resulting float value is then fed as input to the logistic map. Because of its

speed, the logistic map was chosen because it has the minimum calculations needed to achieve the chaotic behaviour. The sequences of random numbers are generated. Each sequence depends on the Initial value and control parameter. The sequences are very sensitive to change of these two initial values. This consists of a series of 64-byte keys that are applied to the grayscale image after dividing this image into a set of blocks (8*8) where each block of the image takes 64 bytes of the key and then work on the rest of the block, but every time it is encrypted with a key, the current keys are combined with the previous using Xor so that the current key is more random and continue this process for all blocks.

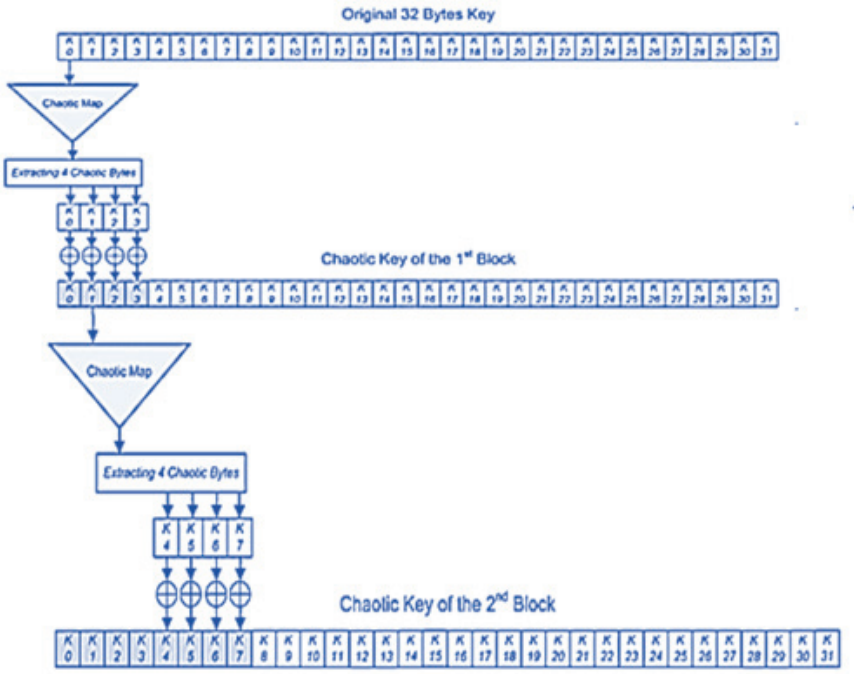

Figure (5) key generation

B. Encryption Medical Image

Generate a random block by a CSalsa20 algorithm these blocks important when need to retrieve the image because it depends on the number of blocks and its original image. After divided image to set of blocks where each block corresponds to 64 bytes of keys after that make XOR between the key and the image. The current keys are combined with the previous make Xor operation so that the current key is more random e XOR operation and this method continues to the last block exists in the image, so the result of encryption is very strong in this case because of the diffusion of this algorithm good.

4. Experimental Results

The experimental results of proposed methods. The three techniques are evaluated for finding the performance of them. The stream cipher used with the key generation from all techniques to test them.

1.Key Generation

key generation mixed with Salsa Algorithm to generate a requested length of keys that used for encryption. The implementation of the Logistic Map Algorithm generates also sequence of numbers (32-bit number) as shown in Table (1). these numbers are real number when used logistic map multiplied by powered 10 number, rounded and modulated with 2^{32} to get exactly 32-bit number.

Table (1) CSalsa number generator

Key generation of CSalsa20			
347892	424951	842773	786421
523986	723765	262307	734170
546233	345192	635897	544215
648901	631182	827878	831516
241792	440634	251371	239573
765231	586234	734682	684156
905569	872618	778436	856498
578456	582236	647712	574477
459458	814245	856919	948034
367238	204838	460448	382587
802299	680789	932983	961345
612988	424949	562136	782357

2.Encryption Various Medical Image

An encryption image by the improvement of Salsa20 algorithm that used logistic map for generated keys. It may be clearly seen that there an emerging pattern in the medical image. However, encryption image by CSalsa20, there is no evidence of any leakage in the image. This is illustrated by the histogram calculation of the image before and after encryption as shown in Figure (6).

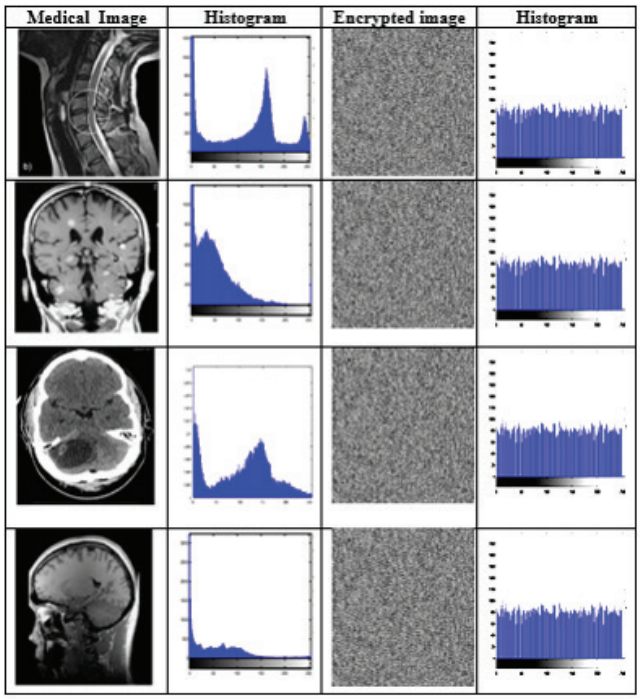

Figure (6): the original tested images with histograms.

Conclusion

The suggested two-round chaotic salsa (CSalsa2) allows only two rounds of scrambling data to achieve a good rate of diffusion, making the cipher faster. A novel technique is introduced in this work to incorporate the chaotic logistic map into the Salsa20 algorithm. The XOR-network is utilized to addressatheastatisticalaleakage observed at the second round of Salsa20. The CSalsa2 proposed should reach a good diffusion rate faster than the original Salsa20, according to the speed measurements. Differential attacks infected the original reduced-round versions of Salsa20. At the inputs of any two Salsa20 sequential blocks, there is the only 1-bit difference. Moreover, the proposed chaotic Salsa model has 33-bits of variations. The widening of the input differences will reinforce the system against various types of attacks.

Financial Disclosure: There is no financial disclosure.

Conflict of Interest: None to declare.

Ethical Clearance: All experimental protocols were approved under the University of information technology and communications and all experiments were carried out in accordance with approved guidelines.

References

[1] Chen, X.; Hu, C," Adaptive medical image encryption algorithm based on multiple chaotic mapping". Saudi Journal of Biological Sciences, vol. 24, pp. 1821-1827,2017.

[2] A. Akhshani, S. Behnia, A. Akhavan, H. Abu Hassan, and Z. Hassan, "A Novel Scheme for Image Encryption Based on 2D Piecewise Chaotic Maps," Optics Communications 283, pp. 3259–3266, 2010.

[3] A. Jolfaei and A. Mirghadri, "An Applied Imagery Encryption Algorithm Based on Shuffling and Baker's Map," Proceedings of the 2010 International Conference on Artificial Intelligence and Pattern Recognition (AIPR-10), Florida, USA, pp. 279–285, 2010.

[4] A. Jolfaei and A. Mirghadri, "A Novel Image Encryption Scheme Using Pixel Shuffler and A5/1," Proceedings of The 2010 International Conference on Artificial Intelligence and Computational Intelligence (AICI10), Sanya, China, 2010.

[5] A. Jolfaei and A. Mirghadri, "An Image Encryption Approach Using Chaos and Stream Cipher," Journal of Theoretical and Applied Information Technology, vol. 19, no. 2, 2010.

[6] M. Sharma and M. K. Kowar, "Image Encryption Techniques Using Chaotic Schemes: a Review," International Journal of Engineering Science and Technology, vol. 2, no. 6, pp. 2359–2363, 2010.

[7] R. A. Rueppel, "Stream Ciphers," Contemporary Cryptology: the Science of Information Integrity, vol. 2, pp. 65–134, 1992.

[8] M. Sharma and M. K. Kowar, "Image Encryption Techniques Using Chaotic Schemes: a Review," International Journal of Engineering Science and Technology, vol. 2, no. 6, pp. 2359–2363, 2010.

[9] Bernstein, D.J.: The Salsa20 Family of Stream Ciphers. In: Robshaw and Billet [30], pp. 84–97.

[10] Fukushima, K., Xu, R., Kiyomoto, S., & Homma, N. (2017). "Fault Injection Attack on Salsa20 and ChaCha and a Lightweight Countermeasure". 2017 IEEE Trustcom/BigDataSE/ICESS.

[11] Mouha, Nicky, and Bart Preneel. "Towards finding optimal differential characteristics for ARX: Application to Salsa20." 2017-06-11]. http://eprint. iacr. org/2013/328 (2013).

[12] Sokouti, Massoud et al. "Medical Image Encryption: An Application for Improved Padding Based GGH Encryption Algorithm." The open medical informatics journal (2016).

[13] D. Wang, C.-C. Chang, Y. Liu, G. Song and Y. Liu, "Digital Image Scrambling Algorithm Based on Chaotic Sequence and decomposition and Recombination of Pixel Values," International Journal of Network Security, vol. 17, pp. 322-327, 2015.

[14] G. Ye, "Image scrambling encryption algorithm of pixel bit based on chaos map," Pattern Recognition Letters, vol. 31, pp. 347-354, 2010.

[15] P. Praveenkumar, R. Amirtharajan, K. Thenmozhi and J. B. B. Rayappan, "Triple chaotic image scrambling on RGB - a random image encryption approach," SECURITY AND COMMUNICATION NETWORKS, vol. 8, pp. 3335-3345, 2015.

[16] N. Ponomarenko, O. Lr=eremeiev, V. Lukin , K. Egiazarian and M. Carli, "Modified image visual quality metrics for contrast change and mean shift accounting," International Conference the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), pp. 305-311, 2011.