METHOD FOR A SIMPLE ENCRYPTION OF IMAGES BASED ON THE CHAOTIC MAP OF BERNOULLI

Luis Alfredo Crisanto Baez, Ricardo Francisco Martinez Gonzalez, Yesenia Isabel Moreno Pavan and Marcos Alonso Mendez Gamboa

Department of Electric-Electronic Engineering, TecNM - Instituto Tecnologico de Veracruz, Veracruz, Mexico

ABSTRACT

In this document, we propose a simple algorithm for the encryption of gray-scale images, although the scheme is perfectly usable in color images. Prior to encryption, the proposed algorithm includes a pair of permutation processes, inspired by the Bernoulli mapping. The permutation disperses the image information to hinder the unauthorized recovery of the original image. The image is encrypted using the XOR function between a sequence generated from the same Bernoulli mapping and the image data, obtained after two permutation processes. Finally, for the verification of the algorithm, the gray-scale Lena pattern image was used; calculating histograms for each stage alongside of the encryption process. The histograms prove dispersion evolution for pattern image during whole algorithm.

KEYWORDS

Algorithm of image encryption; Bernoulli mapping; Permutation.

1. INTRODUCTION

In the last decades, the transmission of information through the internet has increased [1]. This fact is due to the technological advance in hardware and software, which has caused that nowadays it is possible to carry out data transfers at high speed and in high definition. To be able to perform this type of transmissions, a large volume of data must be moved; the fact is considered trivial by users [2]. However, it presents a latent risk of attacks by third parties; who want to obtain information about a specific user [3]. Therefore, it is necessary to implement methods capable of offering certain levels of information security; being encryption the most common way [4]. Encryption needs to be fast, efficient and easy to implement, to prevent bottlenecks in the transfer of information [5], and to reduce affectations to the end user.

A good part of the traffic circulating on the network are images [6]; unfortunately, they can be observed by individuals who can misuse them. The purpose of this algorithm is to use pseudo-random sequences to control the different processes for encrypting the input image. There are multiple ways to generate pseudo-randomness, but in [7], the use of chaos is justified for encryption applications; it is mainly due to the great compatibility between the chaotic properties with the desired characteristics in the ciphers.

DOI:10.5121/ijcsit.2017.9604

As part of the present work, a method is proposed that not only encrypts the image based on a chaotic one-dimensional mapping but also it includes permutations controlled by the same mapping. These permutations will not change the statistics of the image; however, they will alter the image perception, increasing the difficulty to recognize it [8]. To complete the process, the image is encrypted using a sequence generated by the same mapping. As part of this manuscript; in the second section, some details of the Bernoulli mapping will be announced. In the next section, we will detail the used encryption scheme, as well as the logic behind image permutations. Later, the results obtained with the present work will be reported in the fourth section. Finally, the conclusions obtained after the realization of the proposal are narrated on the last section.

2. MATHEMATICAL MODEL OF BERNOULLI

The Bernoulli mapping is also known as the dyadic transformation [9]. This mapping is usually used in a recursive way, and it can be defined as a piece-wise linear function [10], which is shown in Eq. 1.

$$x_{n+1} = \begin{cases} 2\mu x_n 0 \leq x_n < 0.5 \\ 2\mu x_n - 10.5 \leq x_n < 1 \end{cases} \quad (1)$$

Where x_n is the iteration value at the present time, while x_{n+1} is the iteration value at the next time. The value range for x is (0, 1). On the other hand, the variable μ represents the feedback factor and its range is (0, 1).

As it can be seen on the Eq. 1, the domain of the Bernoulli mapping is (0,1); therefore, such interval is converted to a floating-point representation. Which is, accordingly to [11], the only way to effectively represent the decimal values that were established in the mapping domain.
On the other hand, when the mapping is used to process files whose coding is in fixed-point representation, as is the case of the image files, it must be modified to fit within the range of the file. For this reason, the mapping should have a small adjustment, which is shown on next

$$\{Y\} = floor(\{X\} * 2^{bits}) \quad (2)$$

Where *{X}* is the output set produced by the mapping represented by Eq. 1; *{Y}* is the set of the scaled and truncated values obtained from *{X}*; while *bits*is the number of bits that are intended to represent the binary data.

The function floor truncates the *{X}* data after being scaled to the range (0, 2^{bits}); it takes the input data and removes the decimal part, retaining the entire part. As it was mentioned, the present proposal aims at image encryption. Typically, in an image file, eight bits are used to represent pixel depth; so that value will be the number of bits used in this work.

At Figure 1 there is a flow diagram for encrypting data generation. It begins with the definition of the feedback factor (μ) and initial value *x[0]* for the mapping; additionally, the total number of iterations need to encrypt image.

After basic parameters were defined, a loop starts; it iteratively obtains necessary data to encrypt image. Inside the loop, there is a decision to evaluate if previous outcome iteration is lower or

higher than 0.5. The evaluation result is used to assign which segment of the mathematical description will be taken for obtaining *x[n]*; then loop control variable is increased. When the stop condition is reached, the loop is broken and total data is scaled using definition shown in Eq. 2.

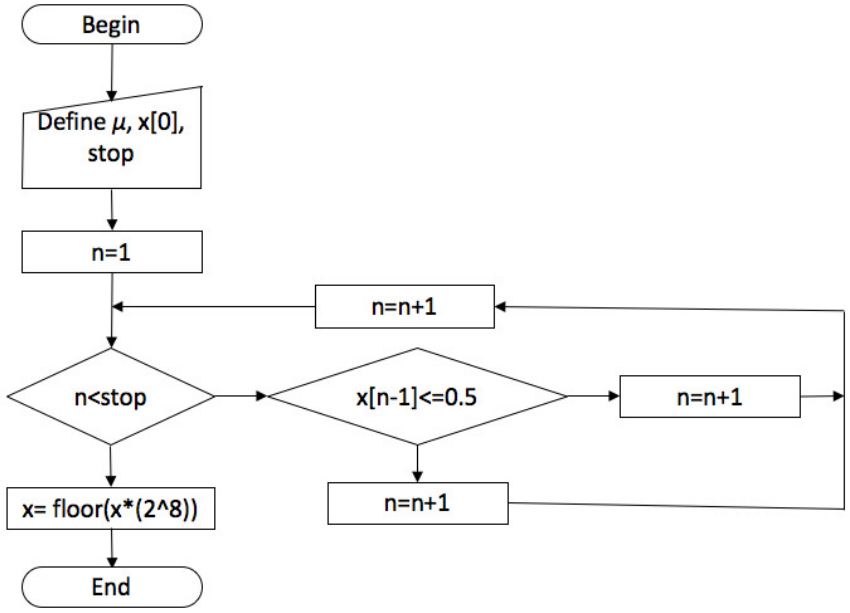

Figure 1. Flow diagram of the implementation of Eq. 1 and its subsequent adaptation for eight-bit resolution files.

The graphical tool known as the bifurcation diagram is used to observe the behavior of the described generator mapping [12].

2.1 Bifurcation diagram

It is a tool that indicates the values that a mapping can take, depending on the feedback factor (μ). The algorithm for obtaining it is quite simple, and is shown below [13]:

- Define the range of the feedback factor, μ_i, and μ_f, as well as the number of values (*step*) that will be in it; create a vector of feedback factors.
- Select the total number of iterations that will have the sequence to be used to generate the diagram. The number of iterations must be higher than 300, nevertheless, it is important to remark that the higher this number the denser diagram will look, but the computational time will be longer.
- Choose an initial value (x_0) arbitrarily. Take the first value of the vector of feedback factors and use the mapping described in Eq. 1 to calculate the *{X}* set for the taken feedback factor. This point must be repeated for all values of the vector; therefore, they will have step *{X}* sets.
- Scale the *{X}* sets using Eq. 2, obtaining the same number of *{Y}* sets. With the sets, graph the values of the sequences obtained in the corresponding space of the feedback

factor used for its generation; producing a sweep of all the output values for the mapping for each μ used.

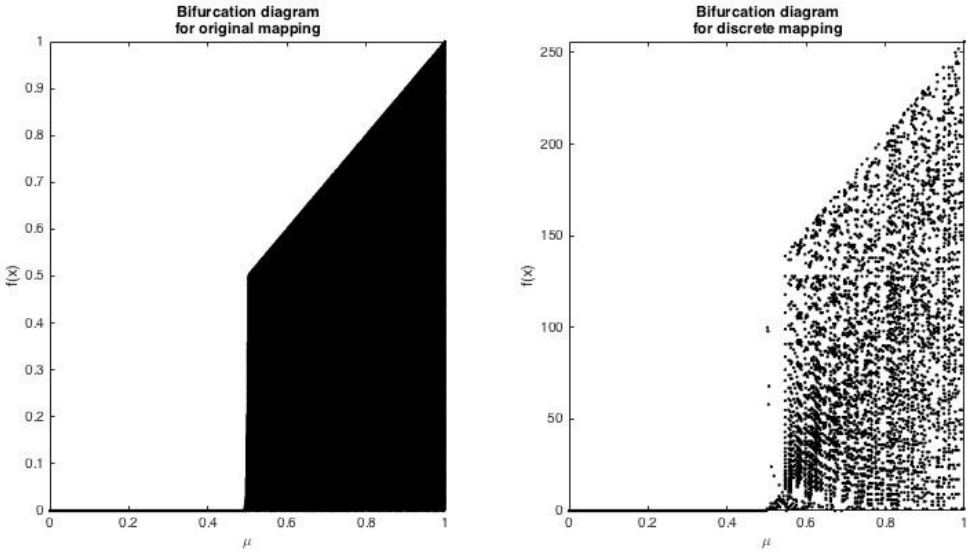

Figure 2. Bifurcation diagrams for *{X}* set (right) and *{Y}* (left)

The main difference of the *{X}* and *{Y}* sets diagrams is their intervals. The diagram for the *{X}* set is in (0, 1), while the one for the *{Y}* set is in (0, 2^{bits}). According to some investigations, the truncation process degenerates the pseudo-random behavior of the mapping; which could be manifesting in the form of the lines present in the bifurcation diagram of the *{Y}* set. Thanks to the bifurcation diagram, the presence of such detail can be observed.

3. DESCRIPTION OF THE PROPOSED ALGORITHM

The proposed algorithm considers loading a gray-scale image (*img*), it has a single layer image and extending it into a vector (*bitplane*) that has a length obtained to multiply the length (*m*) and width (*n*) of the loaded image. This vector is permuted following a criterion established in Eq. 3. This permutation was inspired by the Bernoulli mapping, and it can be seen below:

$$pbitplane_l = \begin{cases} bitplane_{2l} & l \leq \frac{m \cdot n}{2} \\ bitplane_{2l-(m \cdot n)+1} & l > \frac{m \cdot n}{2} \end{cases} \quad (3)$$

Where *pbitplane* is the vector obtained after the permutation criterion. *bitplane* was previously mentioned that are the values of the *img* image in a vector form. *l* is a consecutive number that must cover all the cells of the *bitplane* vector. And finally, *m* and *n* are the dimensions of the loaded image.

According to the flow diagram in Figure 3, the permutation process is the second step in the proposed algorithm. Following the same diagram, it is observed that the next step is the transposition of *pbitplane*. The transposition process starts with the reconstruction in two dimensions from the *pbitplane* vector; for such reason, the vector is chunk in *n* pieces, where each piece is a row in the reconstructed matrix.

With the information in the form of a matrix, called *imgp1*, next step is to transpose the matrix using function available in Matlab for such purpose. The function exchanges the matrix values from horizontal to vertical and viceversa, obtaining the transposed matrix, *imgt1*.

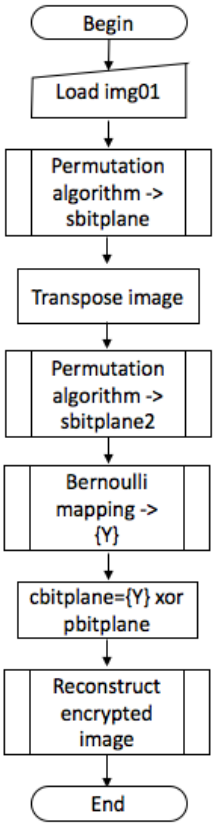

Figure 3. Flow diagram for the algorithm proposed in the present manuscript

After transposition, the matrix *imgt1* is again transformed into a vector, called *bitplane2*. The process of permutation is concluded following the mathematical definition described in Eq. 3, whereby *pbitplane2* is obtained. To explain the permutation process described above, a graphic representation was created that can be seen in Figure 4.

The permutation process seeks to distribute image information contained in the matrix, in such a way that the information of each pixel is as separate as possible from its original position [14]. This process will make it difficult to reconstruct the original image by not presenting a reference point of where the information came from.

Although the process of permutation produces a dispersion of the image data, this is not enough to have an adequate level of security. A disadvantage of any permutation process is that the permuted matrix statistics remains the same as the original matrix [15]. For this reason, the last process is required; it is the data encryption in itself, which is performed twice.

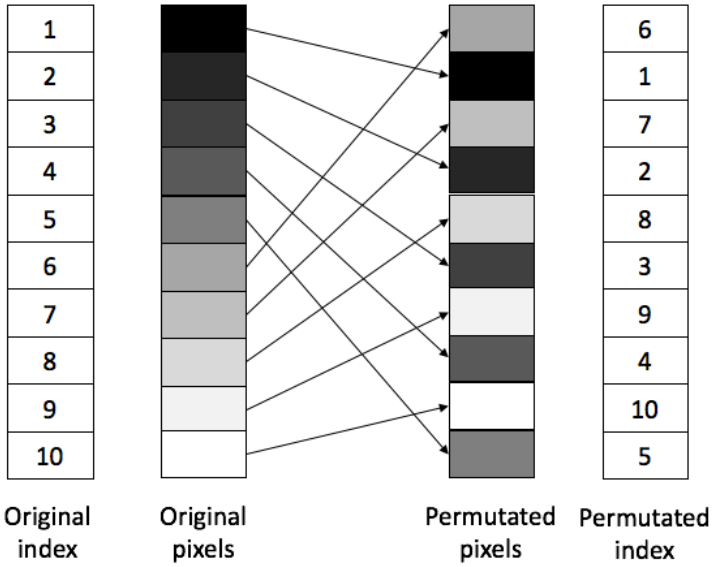

Figure 4. Graphical description of the permutation process.

To begin the encryption process, a sequence is generated using the Bernoulli mapping. As a parameter for the mapping, a feedback factor greater than 0.75 is used; such value is agreed because according to the diagram in Figure 2, it guarantees that the data obtained have a pseudo-random behavior. By using a sequence similar to the random, the encrypted data will have a good level of security; preventing that someone wants to break into them [16].

The encryption process is simple, it starts by defining the feedback parameter and initial condition; and then, to generate the sequence using Eq. 1. It is important to take into account that the length of the generated sequence must be equal to the vectorized image to be encrypted. After obtaining it, the generated sequence is scaled using Eq. 2, with a representation of eight bits. Such length was chosen because the image data have it the same length. Each of the sequence data generated from Eq. 2 is input along with the data from the *pbitplane2* vector into a XORbitwise function [17].

It should be noted that the encryption process takes only one of the data from each source at a time, and it outputs a single data of eight bits at a time. The process continues until the data of both input vectors, *pbitplane2* vector and the pseudo-random sequence vector, are exhausted [18].

4. RESULTS

The first result presented is that corresponding to the histograms obtained from the output data of the different stages of the proposed algorithm. As can be seen in Figure 5, the permutation processes do not alter the data, only its location; so their respective histograms are the same.
According to the presented histograms, the data starts at 0 and ends at 255; which is consistent with the eight-bit range that was established for the algorithm. With respect to the data incidence, its maximum value is 2500; and this incidence is reached for a value close to 160.

The encrypted image histogram changes significantly, and this is because the encryption process takes data and replaces it with a new one; and not only exchanges it as in the case of the

permutation. This fact drastically alters the data statistics, which is reflected in the histogram presented in figure 5 for the encrypted image

The histogram of the encrypted image should be ideally uniform [15]; in other words, completely flat, with all its equal incidences for all values from 0 to 255. In practice, this is extremely difficult to achieve, so the result presented, where the data are much more dispersed and almost the same incidence, can be taken as a good result, due its entropy is 7.9963. The entropy was calculated using the Matlab function, which calculates it with next expression,

$$S = -\sum(p \cdot log_2(p)) \tag{4}$$

where *s* is the entropy value, and *p* is the incidence of every pixel in 256 bins of gray-scale.

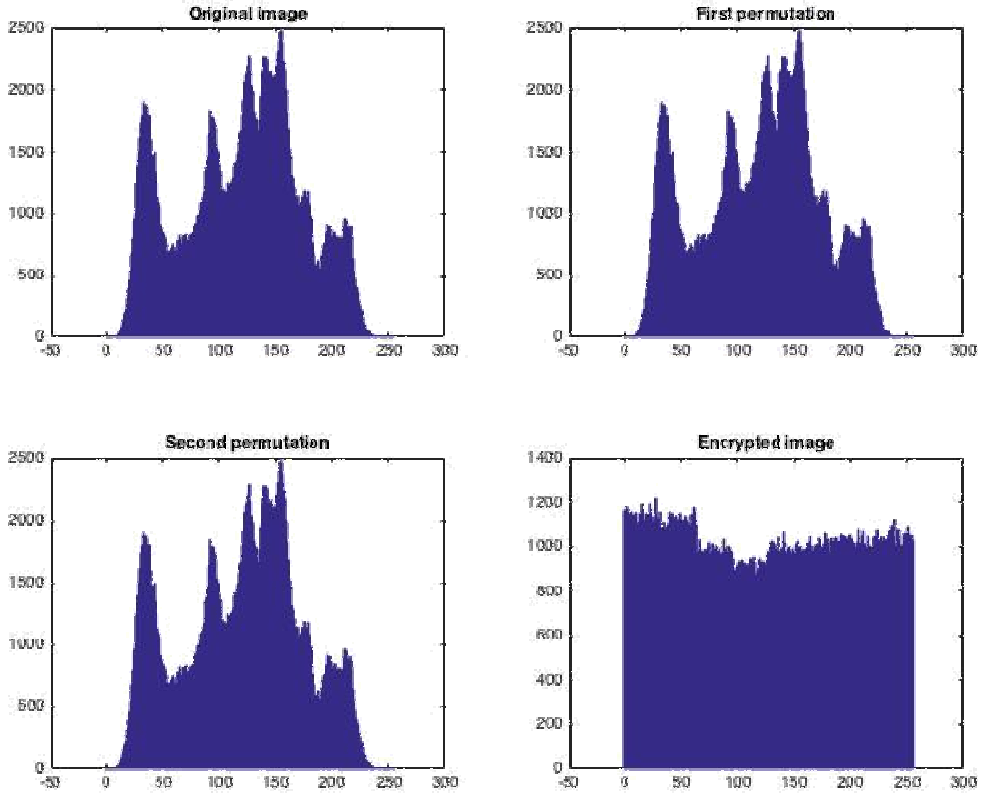

Figure 5. Histograms of the output data of the different stages of the proposed algorithm

For the present work, it was decided to start with the histograms obtained from the input, those delivered after each permutation and encrypted images, they are found in Figure 6.

The input image for the proof was the gray-scale Lena. After the first permutation, the image is distorted but the image of Lena is still recognizable. Between the two permutations there is a process of transposition; using it, the image is rotated and then the second permutation process is performed; resulting in a much less recognizable image. Finally, the encryption alters either the statistics of the image or aspect, which at first glance seems like noise.

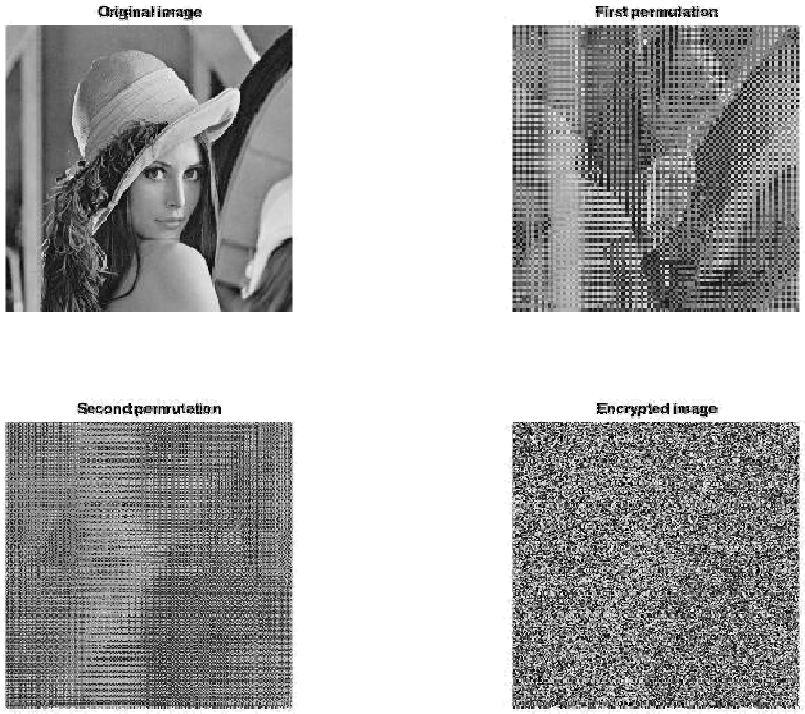

Figure 6. Input image (top left), obtained after the first permutation (top right), second permutation (bottom left) and the final cipher image (bottom right).

5. CONCLUSIONS

With the images of Figure 6 and the histograms previously presented, it is possible to conclude that the Bernoulli mapping can serve as a basis for pixel permutation processes; thus, to generate pseudo-random sequences, that can be used for image encrypting.

Additionally, the present algorithm is simple not only to perform but to understand in its operation; so it can be taken as a frame of reference for future developers of image encryption schemes. So, a double objective is achieved, the realization of a simple method for image encryption and a guide for future work in this area.

REFERENCES

[1] Lagarde, K. C., & Rogers, R. M. (1998). U.S. Patent No. 5,721,908. Washington, Dc: U.S. Patent And Trademark Office.

[2] Jorquera, D. M., Iglesias, V. G., Gimeno, F. J. M., & Pérez, F. M. Mecanismos De Difusiónmasivaenaplicacionesdistribuidas.

[3] Tsai, C. L., & Lin, U. C. (2011). Information Security Of Cloud Computing For Enterprises. Advances In Information Sciences And Service Sciences, 3(1), 132.

[4] Dutta, S., Das, T., Jash, S., Patra, D., & Paul, P. (2014). A Cryptography Algorithm Using The Operations Of Genetic Algorithm & Pseudo Random Sequence Generating Functions. International Journal, 3(5).

[5] Melander, B., Björkman, M., &Gunningberg, P. (2000). A New End-To-End Probing And Analysis Method For Estimating Bandwidth Bottlenecks. In Global Telecommunications Conference, 2000. Globecom'00. Ieee (Vol. 1, Pp. 415-420). Ieee.

[6] Ammar, A., El-Kabbany, A. S. S., Youssef, M. I., &Emam, A. A Novel Secure Image Ciphering Technique Based On Chaos. In 4th Wseas Int. Conf. On Information Science, Communications, And Applications (Pp. 21-23).

[7] Alvarez, G., & Li, S. (2006). Some Basic Cryptographic Requirements For Chaos-Based Cryptosystems. International Journal Of Bifurcation And Chaos, 16(08), 2129-2151.

[8] Nasri, M., Helali, A., Sghaier, H., &Maaref, H. (2010, March). Adaptive Image Transfer For Wireless Sensor Networks (Wsns). In Design And Technology Of Integrated Systems In Nanoscale Era (Dtis), 2010 5th International Conference On (Pp. 1-7). Ieee.

[9] Saleski, A. (1973). On Induced Transformations Of Bernoulli Shifts. Theory Of Computing Systems, 7(1), 83-96.

[10] Li, S., Chen, G., &Mou, X. (2005). On The Dynamical Degradation Of Digital Piecewise Linear Chaotic Maps. International Journal Of Bifurcation And Chaos, 15(10), 3119-3151.

[11] Biswas, H. R. (2013). One Dimensional Chaotic Dynamical Systems. Journal Of Pure And Applied Mathematics: Advances And Applications, 10(1), 69-101.

[12] Persohn, K. J., &Povinelli, R. J. (2012). Analyzing Logistic Map Pseudorandom Number Generators For Periodicity Induced By Finite Precision Floating-Point Representation. Chaos, Solitons & Fractals, 45(3), 238-245.

[13] Agiza, H. N. (1999). On The Analysis Of Stability, Bifurcation, Chaos And Chaos Control Of Kopel Map. Chaos, Solitons & Fractals, 10(11), 1909-1916.

[14] Fu, C., Lin, B. B., Miao, Y. S., Liu, X., & Chen, J. J. (2011). A Novel Chaos-Based Bit-Level Permutation Scheme For Digital Image Encryption. Optics Communications, 284(23), 5415-5423.

[15] Abarbanel, H. (2012). Analysis Of Observed Chaotic Data. Springer Science & Business Media.

[16] Ahmed, H. E. D. H., Kalash, H. M., & Allah, O. S. F. (2007). An Efficient Chaos-Based Feedback Stream Cipher (Ecbfsc) For Image Encryption And Decryption. Informatica, 31(1).

[17] Behnia, S., Akhshani, A., Mahmodi, H., &Akhavan, A. (2008). A Novel Algorithm For Image Encryption Based On Mixture Of Chaotic Maps. Chaos, Solitons & Fractals, 35(2), 408-419.

[18] Wong, K. W., Kwok, B. S. H., & Law, W. S. (2008). A Fast Image Encryption Scheme Based On Chaotic Standard Map. Physics Letters A, 372(15), 2645-2652.