A Chaotic Image Encryption Algorithm Robust against Phase Space Reconstruction Attacks

Jakub Oravec, Ján Turán, Ľuboš Ovseník, Tomáš Huszaník

Department of Electronics and Multimedia Communications,
Faculty of Electrical Engineering and Informatics, Technical University of Košice,
Němcovej 32, 040 01 Košice, Slovakia

jakub.oravec@tuke.sk, jan.turan@tuke.sk, lubos.ovsenik@tuke.sk,
tomas.huszanik@tuke.sk

Abstract: This paper describes a modification of a chaotic logistic map which could be exploited in a field of image encryption. After a summary of basic image encryption methods and problems, the paper mentions properties of a modified version of the logistic map. It is shown that the proposed changes help to achieve greater robustness against phase space reconstruction attacks. The paper also describes the usage of a modified map in an image encryption algorithm. Other techniques applied in the proposed algorithm include key diffusion, ciphertext chaining or four step diffusion stage. Evaluation of properties of the proposed algorithm is done by means of commonly used techniques. The numerical results are then compared with values obtained by other published algorithms.

Keywords: image encryption; logistic map; phase space reconstruction

1 Introduction

One of the first encryption algorithms based on the chaotic maps was proposed in 1989 by Matthews [1]. Since then, various chaotic encryption algorithms were designed, including the one described by Fridrich in 1998 [2]. Fridrich's approach could be considered as important, since it introduced an idea of image encryption. Operations created especially for two dimensional matrices allowed simpler and more effective computations which is still the main advantage over conventional encryption algorithms such as Advanced Encryption Standard (AES). Also, the majority of proposals adopted a two stage encryption process which was described by Fridrich. The first stage – confusion changes positions of plaintext image pixels in order to minimize their correlation. The second stage – diffusion calculates the intensities of pixels in the resulting encrypted image.

The development of the chaotic image encryption algorithms continued by removing their drawbacks. Small key space problems were solved by high dimensional chaotic maps [3] or by combinations of various maps [4]. The Dynamic degradation of chaos, present in the discrete versions of chaotic maps, was studied in [5]. An attack capable of revealing permutations of the image pixels was published by Solak et al. in 2010 [6]. Especially the last mentioned problem caused the usage of more complicated diffusion stages.

The diffusion stage usually employs a technique called ciphertext chaining for establishing dependencies between the intensities of consecutive image pixels. The dependencies are useful for creating different encrypted images for plaintext images with only small amount of changes, however they could be easily found out by Solak's attack. In order to provide certain level of robustness against this attack, the diffusion stage needs to use another operation. Probably one of the most used operations is an addition of elements from pseudo-random (PR) sequences in modular arithmetic. In this case, the Solak's attack would obtain only the pixel intensities combined with the elements of PR sequences.

However, also the generation of the PR sequences requires some care. In the case that the PR sequences are simply computed by some of the chaotic maps, already calculated elements of the PR sequence could be evaluated by the phase space reconstruction attacks. The basic theory of the phase space reconstruction was given by Takens in 1985 [7]. In the context of the image encryption algorithms, the phase space is a set of values that could be achieved by the used chaotic map.

Approaches that are effective against phase space reconstruction can be divided into two groups. The first group changes parameters of the chaotic maps during computation of the PR sequences. Murillo-Escobar et al. [8] described an algorithm where parameter of the used map is changed by values produced by other map. Liu and Miao [9] used binary PR sequences for diffusion by means of a generated code book. The second group of proposals, modifies each sequence element after its computation. Guanghui et al. [10] proposed a scheme with modular design which could utilize various chaotic maps, however the results were presented only for the logistic map. Liu et al. [11] described a modification of calculated sequence elements by another map. In all of these cases, the robustness against the phase space reconstruction was created by suppressing the dependencies between consecutive elements of the PR sequences.

The algorithm presented in this paper tries to provide certain amount of robustness against both Solak's attack and the phase space reconstruction attacks. Since the elements of the generated PR sequences are combined with results of ciphertext chaining by means of bitwise eXclusive OR (XOR), the Solak's attack would be useful only for obtaining the combined values. As the elements of PR sequences are computed by a modified version of chaotic logistic map which is effective against phase space reconstruction, it is difficult to evaluate the elements of PR sequences that are required for a successful decryption.

The rest of the paper is organized as follows: Chapter 2 deals with the logistic map, its properties and the modification which is proposed in this paper for purposes of the image encryption. Chapter 3 describes the algorithms used for encryption and decryption. An analysis of experimental results is given in Chapter 4 and the lastly in Chapter 5, conclusions the paper are given, by a brief summary of advantages and disadvantages of the proposed solution and plans for the future work.

2 Logistic Map and its Modification

2.1 Logistic Map

Logistic map (LM) can be described as an one dimensional chaotic map that uses one parameter $r \in (0; 4)$. An initial value x_0 and the values of x in all iterations of the map belong to an interval (0; 1). The LM was popularized mainly by a paper of May in 1976 [12]. The values of x in consecutive iterations, called iterates of the map could be calculated by applying (1):

$$x_{n+1} = r x_n (1 - x_n), \tag{1}$$

where n denotes an iteration number.

Chaotic behavior of the LM could be illustrated by its bifurcation diagram. The diagram shows values of x_n that were calculated with various values of the parameter r. An example of the bifurcation diagram that has the initial value x_0 equal to 0.5 and plots 800 values of x_n is shown in Figure 1.

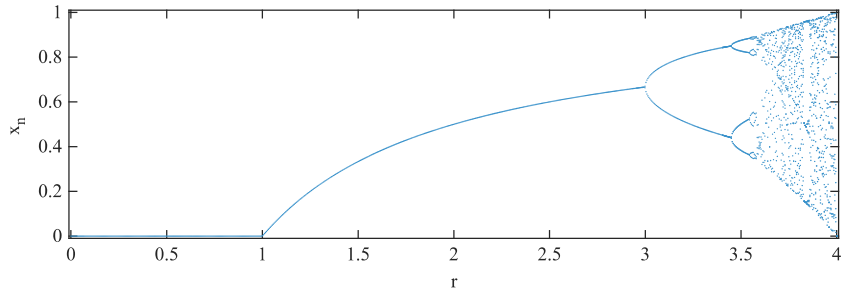

Figure 1
A bifurcation diagram of the logistic map

As it is visible, behavior of the LM is predictable until the parameter r reaches certain value close to 3. At this point the values of calculated iterates start to oscillate between two sets. This property of the LM is known as a bifurcation.

After several other bifurcations, it is quite difficult to see the relations between consecutive iterates. The point where $r \sim 3.56995$ is known also as a start of a chaotic behavior of the LM. However, also some values of r after this point show predictable behavior. These are known as islands of stability. Probably the most notable example is present around r equal to 3.85.

Another important property of the LM is an existence of a transient period. This period contains iterates that are calculated among the first and therefore their values could be predictable. In most cases, the effects of the transient period are suppressed by using some iterates only for modification of the initial value x_0. Common sizes of the transient period are powers of 10, e. g. 100 or 1,000 iterates.

Because the LM could be considered as a discrete version of a continuous logistic differential equation, also the properties regarding the finite amount of possible iterate values should be investigated. Periodicity testing was performed in a computing environment MATLAB R2015a. In total, 10 sequences were computed, each one consisted of 10^8 iterates. The iterates were represented as double precision values (64 bits). As 52 bits are used for storage of a fractional part of these values, their precision could be expressed as $\log_{10}(2^{52}) \sim 15.6536$. Therefore, this data type provides precision of 15 decimal places.

Each of the 10 sequences used the initial value x_0 set to 0.5 and the transient period with size of 1,000 iterates. Sequences differed by value of the parameter r which was set from $4\text{-}10^{-14}$ to $4\text{-}10^{-15}$ with a step of 10^{-15}. The period lengths for the sequences with investigated values of the parameter r are shown in Table 1.

Table 1
Period lengths of the sequences generated by the LM with various values of the parameter r

Value of r	Period length	Value of r	Period length
$4\text{-}10^{-14}$	10^8	$4\text{-}5\cdot10^{-15}$	10^8
$4\text{-}9\cdot10^{-15}$	10^8	$4\text{-}4\cdot10^{-15}$	10^8
$4\text{-}8\cdot10^{-15}$	10^8	$4\text{-}3\cdot10^{-15}$	12,960,875
$4\text{-}7\cdot10^{-15}$	10^8	$4\text{-}2\cdot10^{-15}$	33,767,629
$4\text{-}6\cdot10^{-15}$	10^8	$4\text{-}10^{-15}$	15,599,659

The results from Table 1 show that the values of the parameter r influence period lengths. However, also the worst shown case ($r = 4\text{-}3\cdot10^{-15}$) would be sufficient for element-wise processing of approx. 13 million elements. This number of elements is present in a true color image with a resolution of 2,078x2,078 pixels.

2.2 Phase Space Reconstruction Attacks

As it was already mentioned, the Solak's attack and similar known-plaintext attacks could be used for revealing the permutations of image pixels done in the confusion stage of the image encryption algorithms. Therefore, the main part of

security provided by these algorithms is created by the diffusion stage. Since the diffusion stage of image encryption algorithms usually sequentially processes each pixel of the input images, the amount of used operations should be minimal.

Usually, the diffusion stage consists of two operations. The first one, ciphertext chaining is applied for establishing dependencies between consecutive image pixels in the encrypted images. However, the most popular version of the ciphertext chaining could be reversed by anyone who has access to the encrypted image. The second operation used during diffusion stages is the combination of pixel intensities with the elements of a PR sequence. The combination could be done as an addition in modulo 256 or a bitwise XOR. As the ciphertext chaining and the confusion stage could be broken by attackers, the security of some algorithms depends solely on this operation.

The LM could be used for generating the PR sequences, but there are still some relationships between iterates. The relationships are expressed by a Poincaré plot which uses values of two consecutive iterates as the coordinates of plotted points. An example is shown in Figure 2, where the LM with 10^8 iterates used the initial value $x_0 = 0.5$, the transient period with size of 1,000 iterates and the parameter r set to $4\text{-}4\cdot10^{-15}$. The plot shows only the first 2,000 points for better readability.

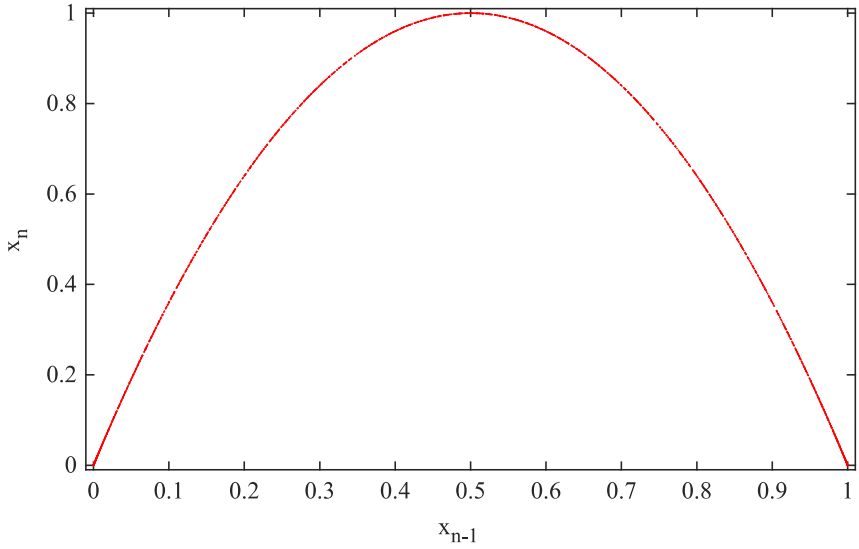

Figure 2
A Poincaré plot for consecutive iterates of the LM

An iterate (x_{n-1}) that was used for the calculation of a current iterate (x_n) could be observed by using x_n as a coordinate on y axis. The plot shows that there are, at most, two points with chosen y coordinate. Therefore, it could be assumed that the iterate x_n was calculated from one of the two possible coordinates on the x axis.

However, this technique has many drawbacks. First of all, the attacker needs to determine the parameter r before construction of plots. Also, the plots do not provide values of x_{n-1} for all possible x_n. For the precision of 10^{-15}, the plots would need to contain $2 \cdot 10^{15}$ points in order to provide all possible relations between consecutive iterates. Finally, if each previous element would be represented by 2 possible values, the reconstruction of sequence with *num* elements would result in 2^{num} possible solutions. Therefore, the computations needed for the phase space reconstruction only by the Poincaré plots seem to be computationally exhaustive.

There are also some other, more efficient approaches for the phase space reconstruction. One of them is known as a time delay method. Some examples of usage of this method are presented in [13] or [14].

2.3 Proposed Modification of Logistic Map

As it was shown in the previous subchapters, the LM has some drawbacks which could impact the security of the designed image encryption algorithms. In order to suppress some of them, we propose a modification of (1):

$$x_{n+1} = 10^4 \cdot r x_n (1 - x_n) \pmod 1 \tag{2}$$

The changes in the equation could seem unimportant, but they cause great differences in its behavior. As a double value with 15 decimal places is multiplied by 10^4, the first four decimal places move to a left side of the decimal mark. Other 11 decimal places are shifted by 4 places to the left side. Remaining 4 decimal places are created by increasing the amount of decimal places of the double value.

The last four decimal places were originally on 16^{th} to 19^{th} decimal place of the iterate. As the double precision produces only numbers with 15 decimal places, the numbers with more decimal places are chosen to be the closest approximations of the double values. This causes a variation from the continuous chaotic systems, which could be viewed as an effect of the dynamical degradation of chaos [5].

The second operation applied in (2) – a usage of modulo 1 is performed to remove the numbers which are on the left side of the decimal mark. This operation is important for producing values of x_{n+1} that belong to an interval (0; 1).

It is important to point out that the changes done in the map (2) cause appearance of other fixed points. While fixed points of the map (1) are well studied, their analysis for the map (2) is not done yet. There are only some observations, e. g. the initial value $x_0 = 0.5$ produces the same value when the parameter r equals 3.9902 or 3.999. This is caused by fact that the map (1) computes iterate value that has number 5 on the fifth decimal place and it is followed by zeros. After the shift of decimal places done in the map (2), iterate values 0.99755 (for $r = 3.9902$) and 0.99975 (for $r = 3.999$) both become 0.5 which is equal to the initial value x_0.

However, it is quite safe to mention that the occurrence of the fixed points for values of the parameter $r \geq 3.9999$ is quite rare, because calculated iterates have more decimal places than the initial value x_0. Hence the probability that the last 11 decimal places of calculated iterate are equal to other iterates is negligible.

2.4 Comparison of the Maps

Following subchapters compare the properties of the LM (1) and the proposed map (2), which is for better readability denoted as MLM (modified logistic map).

2.4.1 Time Consumption

Measurement of the computational time needed for generating sequences used following setup: all sequences generated by the LM (1) and the MLM (2) had the transient period with length of 1,000 elements and the initial value x_0 was equal to 0.5. The number of sequence elements was set as 10^6, 10^7 or 10^8 elements. Two different values of the parameter r were utilized, $4\text{-}10^{-14}$ and $4\text{-}10^{-15}$. Used PC had 2.5 GHz CPU and 12 GBs of RAM and it utilized the computational environment MATLAB R2015a running on Windows 10 OS. The times presented in Table 2 are arithmetic means of 100 repeated measurements.

Table 2
Comparison of the time consumption

Length of sequences [elements]	Value of r	Time needed for the LM [ms]	Time needed for the MLM [ms]
10^6	$4\text{-}10^{-14}$	15.5829	106.1821
	$4\text{-}10^{-15}$	15.4199	104.7699
10^7	$4\text{-}10^{-14}$	170.6184	1185.5463
	$4\text{-}10^{-15}$	168.2268	1170.6595
10^8	$4\text{-}10^{-14}$	1642.9382	11209.9446
	$4\text{-}10^{-15}$	1556.0445	10607.3564

The results presented in Table 2 show that the computational complexity of the MLM is approx. 7 times higher than the one of the LM. This is due to higher amount of operations used for a calculation of each iterate. However, longer computational durations are balanced by advantages that are described in following subchapters. Also it could be stated that different values of the parameter r have only a small impact on the time consumption – the maximal recorded difference was approx. 5 %.

2.4.2 Periodicity Concerns

It could be assumed that the replacement of some decimal places done by the MCM could result in a reduction of the chaotic behavior and therefore also in

smaller period lengths. However, the periodicity that occurred for the LM with certain values of the parameter r ($4-3\cdot10^{-15}$, $4-2\cdot10^{-15}$ and $4-10^{-15}$) was caused by the finite precision of the double values – exactly because there are not any double values between $1-10^{-15}$ and 1. Therefore, all iterates which would have values in an interval $[1-10^{-15}; 1]$ in a system with an infinite precision, result in one of these two values in the double precision system (with finite precision). As the value of the following iterate depends only on a value of the current iterate, the values $1-10^{-15}$ and 1 always produce the same two values. This is the cause of periodicity for the LM with certain values of the parameter r.

Because the MLM changes four decimal places at the end of each iterate, the values before the multiplication in an interval $[1-10^{-15}; 1)$ are changed to an interval $[1-10^{-11}; 1)$ after the multiplication. This prevents computation of the same values in following iterations. Therefore, the period lengths should be enlarged. The resulting period lengths computed with the same setting as was used for the LM in Table 1 (the initial value x_0 set as 0.5, the transient period with size of 1,000 iterates and the parameter r set in an interval from $4-10^{-14}$ to $4-10^{-15}$ with a step of 10^{-15}) are shown in Table 3.

Table 3
Period lengths of the sequences generated by the MLM with various values of the parameter r

Value of r	Period length	Value of r	Period length
$4-10^{-14}$	10^8	$4-5\cdot10^{-15}$	10^8
$4-9\cdot10^{-15}$	10^8	$4-4\cdot10^{-15}$	10^8
$4-8\cdot10^{-15}$	10^8	$4-3\cdot10^{-15}$	10^8
$4-7\cdot10^{-15}$	10^8	$4-2\cdot10^{-15}$	10^8
$4-6\cdot10^{-15}$	10^8	$4-10^{-15}$	10^8

It could be observed that the MLM produces sequences with period lengths of 10^8 elements (length of the generated sequence) in all described cases. As the LM had shorter period lengths for sequences with the parameter r equal to $4-3\cdot10^{-15}$, $4-2\cdot10^{-15}$ and $4-10^{-15}$, it could be concluded that the MLM achieves better results by means of periodicity.

2.4.3 Robustness against the Phase Space Reconstruction

The replacement of the last four decimal places also improves robustness against the phase space reconstruction attacks. Because the decimal places of iterates are shifted by 4 places to the left side, the relations between values of two consecutive iterates are quite unpredictable. This property is also shown by the Poincaré plot in Figure 3, where the relations between consecutive iterates of the MLM are illustrated. The sequence of 10^8 elements was generated with the initial value x_0 of 0.5, the transient period with size of 1,000 iterates and the parameter r set to $4-4\cdot10^{-15}$. The plot shows only the first 2,000 points for better readability.

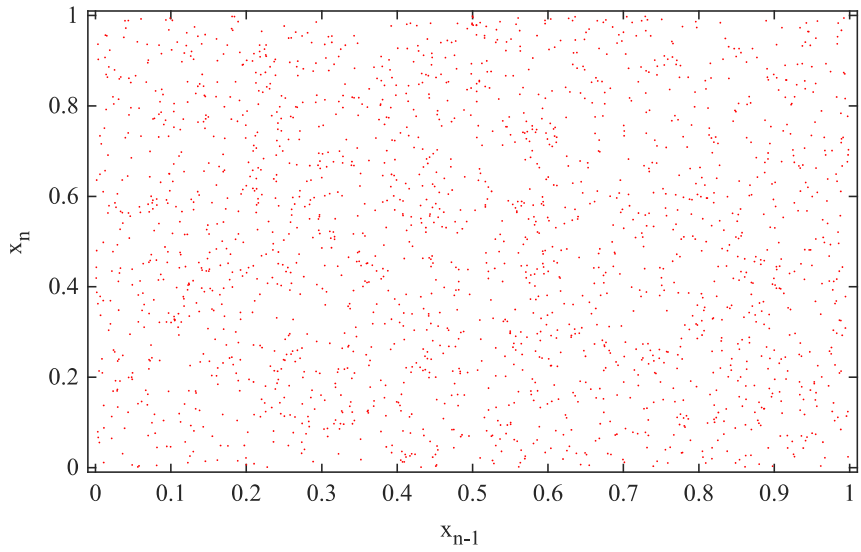

Figure 3
A Poincaré plot for consecutive iterates of the MLM

Because coordinates of the plotted points for the MLM are determined mainly by decimal places close to the fourth place prior to the multiplication, distribution of points in the plot is more uniform. This is due to weak relation between individual decimal places of the iterates before the multiplication. More uniform distribution of the points could be considered as a difficult problem for the phase space reconstruction algorithms, as the number of possible values of previous iterates is not clear (it was 2 for the parabola plotted for the LM in Figure 2). Therefore, complete reconstruction by the Poincaré plots needs to evaluate all possible x coordinates (x_{n-1}) for each current iterate value represented on the y axis (x_n).

3 Proposed Image Encryption Algorithm

The described map (2) shows a set of properties that could be useful for the image encryption algorithms. Therefore, we applied the MLM in an algorithm with usual architecture. Firstly, the confusion stage permutes the image pixels and then the diffusion stage changes their intensities. The proposed algorithm works with images of an arbitrary resolution and color depths of 8 bits (grayscale images) and 24 bits per pixel (true color images). Algorithm in following subchapter is used for the encryption. Different steps done during the decryption are described in subchapter 3.2.

3.1 Encryption Algorithm

Inputs: a plaintext image P, a 16-byte key K

Output: an encrypted image E

Step 1: Height h, width w and the number of color planes num_{cp} of the matrix P are determined. The matrix P is then reshaped to a matrix P_{mat} that has h rows and $w' = w \cdot num_{cp}$ columns. This is done for establishing dependencies between the color planes in true color images.

Step 2: The key K is divided into four subkeys: K_{col} (bytes 1 to 4), K_{row} (bytes 5 to 8), K_{dif1} (bytes 9 to 12) and K_{dif2} (bytes 13 to 16).

Step 3: The subkeys K_{col}, K_{row}, K_{dif1} and K_{dif2} are converted to decimal numbers and they are used for calculation of four values r_{col}, r_{row}, r_{dif1} and r_{dif2} by (3):

$$\begin{aligned} r_{col} &= 3.9999 + \frac{K_{col}}{10^4 \cdot (1+2^{32})} \quad r_{row} = 3.9999 + \frac{K_{row}}{10^4 \cdot (1+2^{32})} \\ r_{dif1} &= 3.9999 + \frac{K_{dif1}}{10^4 \cdot (1+2^{32})} \quad r_{dif2} = 3.9999 + \frac{K_{dif2}}{10^4 \cdot (1+2^{32})} \end{aligned} \tag{3}$$

where 10^4 and $1+2^{32}$ are constants used to ensure that the values of r_{col}, r_{row}, r_{dif1} and $r_{dif2} \in [3.9999; 4)$.

Step 4: Four sequences seq_{col}, seq_{row}, seq_{dif1} and seq_{dif2} are generated by the MLM (2) with the initial values x_0 equal to 0.5. The transient period has size of 1,000 iterates. The value of parameter r used during and after the transient period depends on a calculated sequence and is shown in Table 4. The lengths of generated sequences are following: seq_{col} has w' elements, seq_{row} has h elements and both seq_{dif1} and seq_{dif2} have $2 \cdot h \cdot w'$ elements.

Table 4

Values of the parameter r used during and after the transient period

Sequence	Value of r used for iterates 1 to 250	Value of r used for iterates 251 to 500	Value of r used for iterates 501 to 750	Value of r used for following iterates
seq_{col}	r_{row}	r_{dif1}	r_{dif2}	r_{col}
seq_{row}	r_{dif1}	r_{dif2}	r_{col}	r_{row}
seq_{dif1}	r_{dif2}	r_{col}	r_{row}	r_{dif1}
seq_{dif2}	r_{col}	r_{row}	r_{dif1}	r_{dif2}

Changing of the parameter r during the transient period creates effect known as a key diffusion. In this case, the change of only one byte in the key K should result in differences in all four generated sequences.

Step 5: Elements of the sequences seq_{col}, seq_{row}, seq_{dif1} and seq_{dif2} are quantized by a set (4). The quantized sequences are denoted by an apostrophe.

$$seq'_{col}(k) = \lfloor h \cdot seq_{col}(k) \rfloor \quad seq'_{row}(l) = \lfloor w' \cdot seq_{row}(l) \rfloor$$
$$seq'_{dif1}(i) = round(255 \cdot seq_{dif1}(i)) \tag{4}$$
$$seq'_{dif2}(i) = round(255 \cdot seq_{dif2}(i))$$

where k = 1, 2, …, w', l = 1, 2, …, h and i = 1, 2, …, $2 \cdot h \cdot w'$

Step 6: Two sequences seq'_{dif1} and seq'_{dif2} are reshaped to four matrices. The first half of seq'_{dif1} creates a matrix mat'_{dif11} with h rows and w' columns. Sequence elements are stored in the columns of the matrix, starting from the left side. Other half of seq'_{dif1} creates matrix mat'_{dif12} with the same size. The sequence seq'_{dif2} is split and reshaped by the same way to matrices mat'_{dif21} and mat'_{dif22}.

Step 7: The first part of the confusion stage takes place. Columns of the matrix P_{mat} are scanned and their pixels are permuted by a circular shift. The amount of shifting for each column is given by the elements of sequence seq'_{col}. Image with shifted pixels in its columns is stored in an auxiliary matrix A.

Step 8: The second part of the confusion stage is done. Rows of the matrix A are scanned and their pixels are permuted by a circular shift. The amount of shifting done for each row is given by the elements of sequence seq'_{row}. The permuted image is stored in the matrix A.

Step 9: The first part of the diffusion stage takes place. Pixels in rows of the matrix A are diffused from the top to the bottom row (5):

$$A(l,:) = \left[A(l,:) + A(l-1,:)\right] \oplus mat'_{dif11}(l,:) \tag{5}$$

where l = 1, 2, …, h denotes a row index, *:* stands for all pixels in columns of the matrix A and $\oplus$ is an operator of bitwise XOR. All additions of elements from matrix A are done modulo 256. The top row of pixels (l = 1) uses the bottom row ($l = h$) for the additions.

Step 10: The second part of the diffusion stage is done. Pixels in rows of the matrix A are diffused in the opposite direction from the bottom to the top row (6):

$$A(h-l,:) = \left[A(h-l,:) + A(h+1-l,:)\right] \oplus mat'_{dif12}(h-l,:) \tag{6}$$

where l = 0, 1, …, h-1 denotes the row index. All additions of elements from the matrix A are done modulo 256. The bottom row of pixels ($l = h$) uses the top row (l = 1) for the additions.

Step 11: The third part of the diffusion stage is carried out. Pixels in columns of the matrix A are diffused from the leftmost to the rightmost column (7):

$$A(:,k) = \left[A(:,k) + A(:,k-1)\right] \oplus mat'_{dif21}(:,k) \tag{7}$$

where k = 1, 2, …, w' denotes a column index and *:* stands for all pixels in rows of the matrix A. All additions of elements from the matrix A are done modulo 256.

The leftmost column of pixels ($k = 1$) uses the rightmost column ($k = w'$) for the additions.

Step 12: The fourth and final part of the diffusion stage is computed. Pixels in columns of the matrix A are diffused in the opposite direction from the rightmost to the leftmost column (8):

$$A(:, w'-k) = \left[A(:, w'-k) + A(:, w'+1-k)\right] \oplus mat'_{dif22}(:, w'-k) \tag{8}$$

where $k = 0, 1, \ldots, w'-1$ denotes the column index. All additions of elements from the matrix A are done modulo 256. The rightmost column of pixels ($k = w'$) uses the leftmost column ($k = 1$) for the additions.

Step 13: The auxiliary matrix A is reshaped to a matrix E with h rows, w columns and num_{cp} color planes. The matrix E represents encrypted version of the image P.

3.2 Differences in the Decryption Algorithm

The decryption is analogous to the encryption, only the order of operations is reversed. After the generation and the processing of sequences (Steps 1 to 6), the first change is done when the removal of diffusion removal is applied before the removal of confusion. Also the parts of the diffusion stage are used backwards, starting with Step 12 and continuing to Step 9. These steps apply subtractions instead of the additions, the usage of modulo 256 arithmetic remains the same. Since repeated usage of bitwise XOR produces the values before diffusion, this operation is not changed. However, the order of the two operations is reversed, the bitwise XOR is used prior to the subtractions.

The removal of confusion is done also in the opposite order. First, shuffling in image rows is removed by using circular shifts with negative values of elements from sequence seq'_{row}. Then, the rearrangements in image columns are removed by circular shifts given by negative values of elements from sequence seq'_{col}.

4 Analysis and Comparison of Experimental Results

All experiments with the proposed algorithms were performed on a PC with 2.5 GHz CPU, 12 GBs of RAM in the MATLAB 2015a running on the Windows 10 OS. The set of images used for testing is shown in Figure 4. The first two images, *lena* and *lenaG* have resolution of 512x512 pixels and color depths of 24 and 8 bits per pixel, respectively. Images *black1* and *black2* have resolution of 256x128 pixels and color depth of 8 bits per pixel. Image *black2* has a pixel with intensity 255 located on the coordinates [128; 64]. The keys used during experiments are illustrated in Table 5. Differences between similar keys are indicated by bold characters. The images from Figure 4 encrypted by key K_1 are shown in Figure 5.

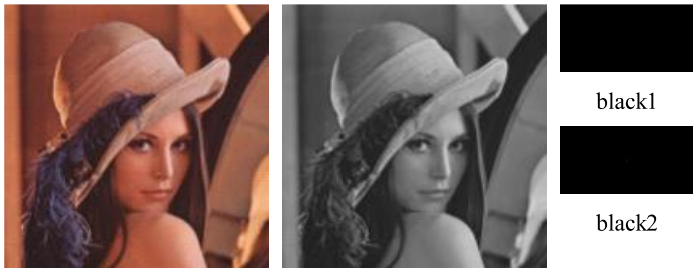

Figure 4
Set of images used for the experiments

Table 5
Keys used for the experiments

Key	Value
K_1	0x74686973697361736563726574 6B6579
K_2	0x74686972697361736563726574 6B6579
K_3	0x74686973697361736563726574 6C6579

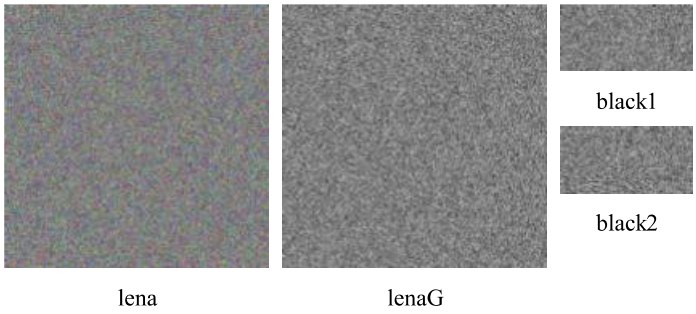

Figure 5
Encrypted versions of the images from the testing set

4.1 Size of the Key Space and the Key Sensitivity

A key space consists of all possible keys that could be used. As the proposed algorithm utilizes 16-byte key K, the total size of the key space is given as $256^{16} = 2^{128}$. If we would consider the time required for a decryption of one true color image with resolution of 512x512 pixels as approx. 550 ms, the brute-force attack on the image with these parameters would take approx. $5.9347 \cdot 10^{30}$ years. Therefore, the brute-force attack could be considered as infeasible.

Sensitivity of the proposed algorithm to used keys could be investigated by an encryption with one key and a decryption with other keys. The images decrypted by an incorrect key should not contain any information about the original plaintext image (which is the same as the image decrypted by a correct key). This experiment is shown in Figure 6, where the image *lena* was encrypted by key K_1. Then it was decrypted with all three keys.

decrypted by K_1 decrypted by K_2 decrypted by K_3

Figure 6
Illustration of the key sensitivity

Please note that the change of a key in portion which is used for the confusion stage (pixel rearrangement) also influences the diffusion stage. This property is result of the key diffusion and it is visible on image decrypted by key K_2.

4.2 Statistical Attacks

Statistical attacks try to obtain some properties of the image encryption algorithms by comparing known pairs of the plaintext images and corresponding encrypted images. When some properties of the encryption algorithms are known, statistical attacks could be used for breaking the algorithms or their parts. There are several measures that could be used to illustrate robustness against the statistical attacks.

The first measure is histogram comparison. The histogram of the encrypted image should have distribution close to uniform without notable peaks which are present in the histogram of the plaintext image. Histograms of the image *lenaG* before and after encryption by the key K_1 are shown in Figure 7.

Second measure is illustrated by scatter plots that contain points with coordinates given by intensities of two adjacent image pixels. The adjacencies could be horizontal, vertical or diagonal. If the plotted points are close to line y = x, it could be concluded that the intensities of adjacent pixels are highly correlated. The distribution of the points for the encrypted images should be close to uniform. The scatter plots for the horizontal adjacencies of 1,000 randomly chosen pixel pairs from the image *lenaG* and its version encrypted by key K_1 are shown in Figure 8.

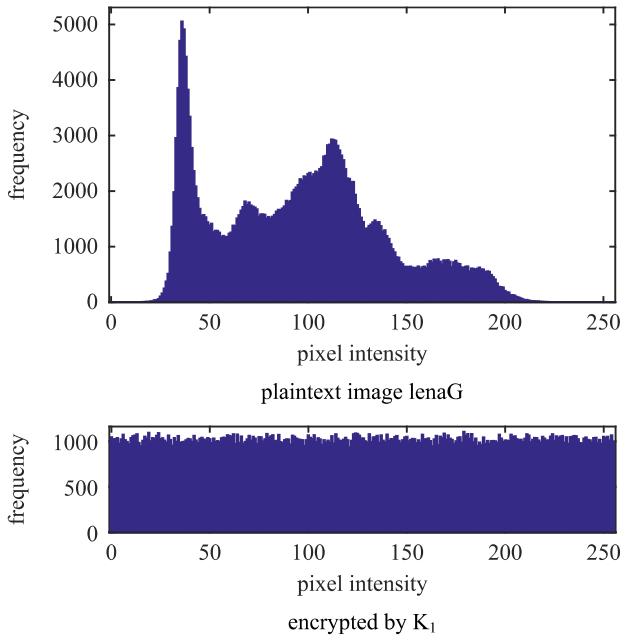

Figure 7
Comparison of the histograms

Third measure are values of correlation coefficients ρ, computed separately for each color plane by (9) for the horizontally (ρ_{hor}), vertically (ρ_{ver}) or diagonally (ρ_{diag}) adjacent image pixel pairs. Vector I_1 contains intensities of the first pixels from the pairs and vector I_2 is created by scanning of the adjacent pixel intensities.

$$\rho = \frac{\sum_{pp=1}^{npp}\left(I_1(pp)-\overline{I_1}\right)\cdot\left(I_2(pp)-\overline{I_2}\right)}{\sqrt{\sum_{pp=1}^{npp}\left(I_1(pp)-\overline{I_1}\right)^2\cdot\sum_{px=1}^{npp}\left(I_2(pp)-\overline{I_2}\right)^2}} \quad [-] \tag{9}$$

where $pp = 1, 2, \ldots, npp$ is an index of pixel pair, npp denotes an amount of the pixel pairs and $\overline{I_x}$ stands for an arithmetic mean of vector I_x.

The last measure is an entropy H which is calculated for each color plane by (10).

$$H = -\sum_{in=0}^{255} p(in)\cdot\log_2(p(in)) \quad [bits/pixel] \tag{10}$$

where $p(in)$ is a probability of occurrence of a pixel with intensity in.

Computed values of the correlation coefficients ρ and entropy H are included with other values in Table 6. The values of the correlation coefficients are arithmetic means of 100 repeated measurements for 1,000 randomly chosen pixel pairs.

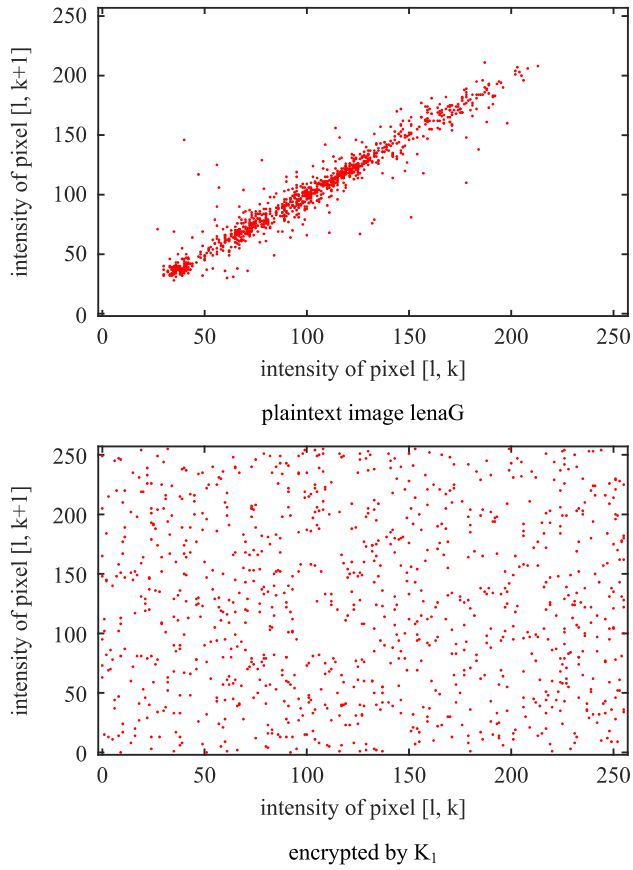

Figure 8
Scatter plots for the horizontally adjacent pixel pairs

4.3 Differential Attacks

Differential attacks reveal the properties of the encryption algorithms by exploring differences in encrypted versions of similar plaintext images. Robustness against the differential attacks is evaluated by two parameters – Number of Pixel Change Ratio (NPCR) and Unified Average Changing Intensity (UACI). Both these parameters use a pair of plaintext images, P_1 and P_2 which differ only in an intensity of one pixel. Furthermore, the size of this difference is minimal (one intensity level). These two images are then encrypted by the same key as E_1 and E_2. NPCR for individual color planes of image pair E_1 and E_2 is given by (11):

$$NPCR = \frac{100}{h \cdot w} \cdot \sum_{l=1}^{h} \sum_{k=1}^{w} Diff(l,k) \quad [\%] \tag{11}$$

where h and w represent height and width of images E_1 and E_2, l and k denote row and column indexes, *Diff* is a difference matrix, $Diff(l, k) = 1$ if $E_1(l, k) \neq E_2(l, k)$, otherwise $Diff(l, k) = 0$.

Calculation of UACI for the color planes uses (12):

$$UACI = \frac{100}{h \cdot w} \cdot \sum_{l=1}^{h} \sum_{k=1}^{w} \frac{\left| E_1(l,k) - E_2(l,k) \right|}{2^L - 1} \quad [\%] \tag{12}$$

where L is a color depth of a color plane in bits per pixel.

A difference between NPCR and UACI is hidden in the way how these parameters evaluate changes in the pair of encrypted images. While NPCR only sums the number of pixel intensity changes, UACI also records the sizes of the changes. Calculated values of NPCR and UACI for the set of images presented in Figure 4 are included in Table 6. These values are arithmetic means of 100 repeated measurements with randomly chosen pixel with modified intensity.

Both NPCR and UACI are given as percentages. Wu et al. [15] described theoretically critical values of NPCR and UACI that depend on the resolution of the encrypted images. For the resolution of 512x512 pixels, the theoretically critical values are 99.6094% for NPCR 33.4635% for UACI.

4.4 Time Consumption and Computational Complexity

Images as a data type could be characterized by high redundancy. Therefore, the speed of encryption or decryption is an important property of the image encryption algorithms. Durations of encryption t_{enc} and decryption t_{dec} are included in Table 6. These times are arithmetic means of 100 repeated measurements.

The computational complexity of image encryption algorithms could be expressed also by amount of performed operations in so-called big O notation (also known as an asymptotic notation). Following paragraph investigates the case of encryption, with the height of a processed image denoted as h, its width w and number of color planes num_{cp} represented by $w' = w \cdot num_{cp}$. The generation of PR sequences by the modified version of the logistic map takes $2hw' + h + w' + 4{,}000$ operations. Each operation for this stage of algorithm consists of three multiplications, one subtraction and one modulo operation. Quantization of PR sequences requires $2hw' + h + w'$ multiplications and rounding operations. Confusion stage of the proposed algorithm needs $h + w'$ circular shifts of sequences with h or w' elements. The last stage of the algorithm – diffusion computes $2hw'$ additions and XOR additions.

If the complexity of various operations would be considered as the same, the proposed algorithm requires $16hw' + 7(h + w') + 16{,}000$ operations for one encryption. Therefore, it could be concluded that the computational complexity

for the proposed algorithm is linear and it depends solely on a resolution of the processed image (*hw'*). As for most cases, the term *16hw'* is greater than the other two terms, the big *O* notation of the proposed algorithm could be given as $O(16n)$.

4.5 Comparison of Numerical Results

Resulting values of the numerical measures for the proposed algorithm are shown in Table 6. Characters *R*, *G* and *B* in the brackets denote individual color planes of true color images (Red, Green or Blue). The word "plain" in key column is used for plaintext images which are not encrypted.

Table 6
Numerical results achieved by the proposed algorithm

Value	Key	*lena (R)*	*lena (G)*	*lena (B)*	*lenaG*	*black1*	*black2*
ρ_{hor} [-]	plain	0.973	0.9677	0.953	0.9712	~1	~1
	K_1	0.0094	0.003	-0.0006	-0.0013	0.0156	-0.0011
	K_2	-0.0031	-0.0049	-0.005	0.0044	0.0189	-0.01
ρ_{ver} [-]	plain	0.9737	0.9735	0.956	0.9743	~1	~1
	K_1	0.0066	0.0044	0.001	-0.0005	-0.0071	0.0202
	K_2	-0.0055	-0.0023	-0.0067	-0.0012	0.0049	0.0017
ρ_{diag} [-]	plain	0.9541	0.9536	0.9349	0.9572	~1	~1
	K_1	-0.005	0.0063	0.0005	-0.0034	0.0017	0.0007
	K_2	-0.001	0.0021	-0.0019	0.0018	0.0001	-0.0018
H [bits/ pixel]	plain	7.5883	7.106	6.8147	7.2344	0	0.0005
	K_1	7.9992	7.9992	7.9992	7.9992	7.9944	7.9942
	K_2	7.9992	7.9993	7.9993	7.9993	7.9944	7.9945
NPCR [%]	K_1	99.7456	99.6094	99.6906	99.6792	99.6674	99.6429
	K_2	99.7318	99.6227	99.675	99.7074	99.6307	99.6521
UACI [%]	K_1	33.6223	33.6513	33.6724	33.6274	33.6473	33.6338
	K_2	33.6308	33.6425	33.6356	33.6401	33.6186	33.6788
t_{enc} [ms]	K_1	598.481			191.1818	24.6219	24.2081
	K_2	601.264			193.2376	24.6448	24.1386
t_{dec} [ms]	K_1	552.0765			173.8621	22.1377	21.9269
	K_2	555.3535			174.073	22.1272	21.8784

Values of the correlation coefficients ρ for images *black1* and *black2* are close to 1, as the first image consists only of pixels with zero intensity. The second image has one pixel with different, maximal intensity level (255). Presented encryption times t_{enc} and decryption times t_{dec} for the true color image are obtained for encryption or decryption of all three color planes. A comparison of results with other algorithms which used the same images or color planes is shown in Table 7.

Table 7
Comparison of the numerical results with other algorithms

Value	Proposed algorithm		Ref. [8]	Ref. [10]
	lena (R)	***lenaG***	***lena (R)***	***lenaG***
ρ_{hor} [-]	0.0094	-0.0013	0.0135	-0.0278
ρ_{ver} [-]	0.0066	-0.0005	Unknown	-0.0065
ρ_{diag} [-]	-0.005	-0.0034	Unknown	-0.0074
H [bits/pixel]	7.9992	7.9992	7.9974	7.9895
NPCR [%]	99.7456	99.6792	99.63	99.66
UACI [%]	33.6223	33.6274	33.31	33.57
t_{enc} [ms]	598.481	191.1818	243.2	not reported
complexity [operations]	$O(16n)$		not reported	$O(8n)$

Based on the presented results, it could be stated that our algorithm achieves better values of the correlation coefficients ρ than the two other algorithms. Values of entropy H are also higher. NPCR and UACI are slightly over the critical values and they are also considerably better than the results obtained by other algorithms. The smallest difference between algorithms is present for NPCR value of the image *lenaG*. Also, the values of all mentioned parameters are quite similar for two tested keys. However, the advantages of our proposal are balanced by its slower performance – it is approx. 2.5 times slower than the approach from [8] and it has two times the computational complexity of the algorithm presented in [10]. However, the computational complexity of the scheme [10] depends on length of used feedback, which was chosen as 4 by the authors of algorithm [10].

Conclusions and Future Work

In this paper, we describe a modification of a chaotic logistic map, which was also employs in an image encryption algorithm. Image encryption can be used in various applications, such as an improvement of data security in steganographic systems [16] [17] or for secure transmission and storage of features in biometric systems which utilize images [18]. Because the modified version of the logistic map is more robust, to phase space reconstruction attacks, the encryption algorithm also holds this property. Other required properties of the algorithm are achieved by a combination of several techniques, such as key diffusion, ciphertext chaining or four step diffusion method. However, the number of these techniques causes slower performance of the proposed algorithm.

The proposed image encryption algorithm reaches correlation coefficients, with values < 0.01 for all planes of true color image *lena* and also for grayscale image *lenaG*. The computed results of entropy are close to the theoretical bound of 8 bits per pixel. Also, the arithmetic means of 100 repeated measurements of NPCR and UACI are equal to or higher than the required values reported by Wu et al. [15].

Therefore, the proposed algorithm is robust against all types of attacks commonly used for cryptanalysis of image encryption algorithms.

In the future, we plan to explore other possible techniques which would provide similar properties, with a smaller computational complexity. This goal may involve other modifications of the equation of the logistic map. The solution proposed in this paper multiplies iterates of the logistic map by a constant. This operation and the fact that the following iterate needs to be from an interval (0; 1) cause usage of modular arithmetic. Other operations, which do not change the interval of iterates (e. g. rearrangement of decimal places of iterates or combinations of multiple iterates), could be faster than the two operations utilized in this paper, nonetheless, the properties of other operations regarding phase space reconstruction attacks, needs further investigation.

Acknowledgement

This work was supported by the research grants KEGA 023TUKE-4/2017 and APVV-17-0208.

References

[1] R. Matthews: On the Derivation of a "Chaotic" Encryption Algorithm. Cryptologia, Vol. 13, 1989, No. 1, pp. 29-42

[2] J. Fridrich: Symmetric Ciphers Based on Two-dimensional Chaotic Maps. International Journal of Bifurcation and Chaos, Vol. 8, 1998, No. 6, pp. 1259-1284

[3] Y. Mao, G. Chen, S. Lian: A Novel Fast Image Encryption Scheme Based on 3D Chaotic Baker Maps. International Journal of Bifurcation and Chaos, Vol. 14, 2004, No. 10, pp. 3613-3624

[4] Z.-H. Guan, F. Huang, W. Guan: Chaos-based Image Encryption Algorithm. Physics Letters A, Vol. 346, 2005, No. 1-3, pp. 153-157

[5] S. Li, G. Chen, X. Mou: On the Dynamical Degradation of Digital Piecewise Linear Chaotic Maps. International Journal of Bifurcation and Chaos, Vol. 15, 2005, No. 10, pp. 3119-3151

[6] E. Solak, C. Çokal, O. T. Yildiz, T. Biyikoğlu: Cryptanalysis of Fridrich's Chaotic Image Encryption. International Journal of Bifurcation and Chaos, Vol. 20, 2010, No. 5, pp. 1405-1413

[7] F. Takens: On the Numerical Determination of the Dimension of an Attractor. Dynamical Systems and Bifurcations. Lecture Notes in Mathematics, Vol. 1125, Berlin Heidelberg: Springer, 1985, pp. 99-106, ISBN 978-3-540-15233-0

[8] M. A. Murillo-Escobar, C. Cruz-Hernández, F. Abundiz-Pérez, R. M. López-Gutiérrez, O. R. Acosta Del Campo: A RGB Image Encryption

Algorithm Based on Total Plain Image Characteristics and Chaos. Signal Processing, Vol. 109, 2015, No. C, pp. 119-131

[9] L. Liu, S. Miao: A New Image Encryption Algorithm Based on Logistic Chaotic Map with Varying Parameter. SpringerPlus, Vol. 5, 2016, No. 1, pp. 289-300

[10] C. Guanghui, H. Kai, Z. Yizhi, Z. Jun, Z. Xing: Chaotic Image Encryption Based on Running-Key Related to Plaintext. The Scientific World Journal, Vol. 2014, 2014, No. 1, pp. 1-9

[11] Y. Liu, Y. Luo, S. Song, L. Cao, J. Liu, J. Harkin: Counteracting Dynamical Degradation of Digital Chaotic Chebyshev Map via Perturbation. International Journal of Bifurcation and Chaos, Vol. 27, 2017, No. 3, pp. 1-14

[12] R. M. May: Simple Mathematical Models with Very Complicated Dynamics. Nature, Vol. 261, 1976, No. 5560, pp. 459-467

[13] B. Kliková, A. Raidl: Reconstruction of Phase Space of Dynamical Systems Using Method of Time Delay. Proceedings of WDS'11, Prague (Czech Republic), 2011, pp. 83-87, ISBN 978-8-073-78186-6

[14] Y. Rong-Yi, H. Xiao-Jing: Phase Space Reconstruction of Chaotic Dynamical System Based on Wavelet Decomposition. Chinese Physics B, Vol. 20, 2011, No. 2, pp. 1-5

[15] Y. Wu, J. Noonan, S. Agaian: NPCR and UACI Randomness Tests for Image Encryption. Journal of Selected Areas in Telecommunications, Vol. 2, 2011, No. 4, pp. 31-38

[16] V. Hajduk, M. Broda, O. Kováč, D. Levický: Image Steganography with QR Code and Cryptography. Proceedings of Radioelektronika 2016, Košice (Slovakia), 2016, pp. 350-353, ISBN 978-1-509-01673-0

[17] J. Oravec, J. Turán: Substitution Steganography with Security Improved by Chaotic Image Encryption. Proceedings of Informatics 2017, Poprad (Slovakia), 2017, pp. 284-288, ISBN 978-1-538-60888-3

[18] F. Abundiz-Pérez, C. Cruz-Hernández, M. A. Murillo-Escobar, R. M. López-Gutiérrez, A. Arellano-Delgado: A Fingerprint Image Encryption Scheme Based on Hyperchaotic Rössler Map. Mathematical Problems in Engineering, Vol. 2016, 2016, pp. 1-15