Computing and Informatics, Vol. 39, 2020, 1250–1281, doi: 10.31577/cai_2020_6_1250

MITIGATING DRAWBACKS OF LOGISTIC MAP FOR IMAGE ENCRYPTION ALGORITHMS

Jakub Oravec, Ľuboš Ovseník, Ján Turán, Tomáš Huszaník

Department of Electronics and Multimedia Communications
Technical University of Košice
Němcovej 32
040 01 Košice, Slovakia
e-mail: {jakub.oravec, lubos.ovsenik, jan.turan, tomas.huszanik}@tuke.sk

Abstract. This paper identifies and analyses some drawbacks of the logistic map which is still one of the most used chaotic maps in image encryption algorithms. As some of the disadvantages are caused by inappropriate implementations of the logistic map, this paper proposes a set of rules which should lead to enhancement of the desired chaotic behavior. Probably the most important rule introduces alternating value of parameter utilized by the logistic map. With careful choice of values and an adapted quantization technique, some of the issues should be fixed and theoretically also the values of numerical parameters should be improved. These assumptions are verified by applying the proposed set of rules on an algorithm from our prior work. Effects of the proposed rules on the used algorithm are investigated and all necessary modifications are thoroughly discussed. The paper also compares obtained values of commonly used numerical parameters and computational complexity with some other image encryption algorithms based on more complex chaotic systems.

Keywords: Fixed point, image encryption, logistic map, Lyapunov exponent, periodic cycle

Mathematics Subject Classification 2010: 94A60, 68U10

1 INTRODUCTION

Nowadays, there are many possible ways for establishing security of multimedia data. Each solution has certain advantages and drawbacks determined by its design and expected applications. This statement is valid even for well-established "conventional" encryption algorithms such as Advanced Encryption Standard (AES) [1]. While AES could be used in a wide amount of applications, it was designed primarily for operations with small blocks of hexadecimal data. Careless implementation of AES for image encryption, especially using simpler modes of operation could lead to undesired results. One example of this situation is shown in Figure 1 where a grayscale image with a resolution of 256×128 pixels was encrypted by AES using Electronic CodeBook (ECB) mode of operation [2]. This example used key `0× C4 EB 50 BC 0E C5 EB 50 BC 0E C5 EB 50 BC 0E C5` for the encryption.

plain image (before encryption) image encrypted by AES ECB

Figure 1. Visible artifacts in an image encrypted by AES ECB

The encrypted image displayed in Figure 1 shows that AES in ECB mode could create visible artifacts that are correlated with the plain image. This drawback is not present in other modes of operation [2], but not all users are familiar with it.

Presented issue of AES in ECB mode is caused by its design – the image is processed as a set of blocks. Each block is replaced by a corresponding block from a code book during the encryption. If the plain image contains several identical blocks of image pixel intensities, all these blocks are replaced by the same block from the code book. This situation is undesirable since the images are known for rather high redundancy and they can contain multiple groups of identical blocks.

On the other hand, encryption algorithms that were designed specifically for image encryption try to adjust their steps to characteristics of image data. This means that the dedicated image encryption algorithms are robust against attacks that are feasible on image data. Also, performance of the dedicated algorithms should be better, however this is hard to prove as the conventional algorithms use various tricks such as hardware acceleration [3] for increasing their performance.

The dedicated image encryption algorithms found their applications mainly in areas which are sensitive to presence of even small artifacts such as encryption of medical images [4] or encryption of secret data in steganographic systems [5]. Also some biometric systems could benefit from lower computational complexity of carefully designed image encryption algorithms [6].

The history of dedicated image encryption algorithms goes back to 1989, when Matthews proposed an encryption algorithm that utilized logistic map (LM) [7]. While this scheme did not encrypt images, it described some techniques that later became essential for the image encryption algorithms. The choice of used chaotic map was explained by Matthews mainly by good understanding of the LM at the time. One of the first extensive studies of LM was published already in 1976 by May [8] and since then, multiple works investigated properties of LM in general [9, 10, 11], or its properties in certain applications [12, 13, 14, 15, 16, 17].

The work of Matthews was later continued by Fridrich, who described one of the first encryption algorithms designed specifically for operations with images in 1998 [18]. Probably the biggest advantage of Fridrich's algorithm is its architecture which was later adopted by many other image encryption algorithms. Some of the drawbacks of this algorithm and similar ones were described by Solak et al. in 2010 [19] and later also by Xie et al. in 2017 [20]. The newest image encryption algorithms take into account the drawbacks of Fridrich's architecture and they utilize techniques to overcome these issues [21, 22, 23, 24]. However, less work has been done in mitigating vulnerabilities of the LM in image encryption algorithms even if they are already described [12, 14].

These vulnerabilities such as occurrence of periodic cycles [10], existence of a relation between successive iterates or their uneven distribution [12] negatively affect generated pseudo-random (PR) sequences. If the generated sequences do not possess required statistical properties and they are still used for encryption, the analysis of encrypted data can reveal information about algorithm's architecture or a part of used key. Furthermore, some attacks can obtain parts of the plain data and this could eventually lead to a break of whole encryption algorithm [13, 14, 15, 16].

The LM is not the only chaotic map used for image encryption, however it can be considered as one of the most popular. Some approaches utilize various modifications of LM or other chaotic maps. Cao et al. described a new custom map in their paper from 2018 [25] that was a result of cascading LM and other chaotic map. Similar solution was proposed by Alawida et al. in 2019 [26]. In both these cases authors review basic properties of the resulting maps by means of their chaotic behavior. However, long-time experience with LM shows that new vulnerabilities could be found out even after multiple years of research [14]. Therefore usage of a newly derived chaotic map without exhaustive analysis of its properties could be viewed as a disadvantage of whole image encryption algorithm.

Other notable approach for enhancement of chaotic behavior is usage of chaotic maps with far too many dimensions. Usage of multidimensional systems could lead to big key spaces, however in general they have a negative impact on computational complexity of the algorithms. Probably the most extreme cases are algorithms by Zhu and Zhu from 2018 [27] or by Sun et al. from 2019 [28]. The first proposal uses six dimensional system, while the second one uses seven dimensional system.

In this paper, we would like to address some of the basic defects regarding usage of LM in the image encryption algorithms. While some are caused by the map itself (or its discretized version), some can be viewed as an inappropriate implementation

by the researchers. We would like to propose some fixes for issues that still do not have an efficient solution. As these fixes can affect other techniques used in the field (such as quantization [29]), this paper will briefly describe also some well-known practices in the field of image encryption. Furthermore, in order to provide a complex insight to all necessary blocks of an image encryption algorithm, we will shortly mention also some techniques described in our prior works (e.g. in [24]).

The second goal of this paper is to compare values of commonly used numerical parameters and measure the increase of computational complexity caused by the proposed fixes. As equation of the LM is considered to be quite simple and therefore the map is considered as relatively fast [10], the proposed fixes should not greatly affect the computational complexity. The increase of computational complexity would be measured by applying the proposed fixes to our prior algorithm [24].

The rest of the paper is organized as follows: Section 2 analyzes the behavior and general properties of the LM. Section 3 describes proposed fixes and the way how they interact with other used techniques. Also some other useful methods are mentioned. Section 4 experimentally verifies impact of proposed fixes and contains measurements of numerical parameters and computational complexity. Lastly, Section 5 summarizes the paper and gives a brief overview of possible future work.

2 LOGISTIC MAP AND ITS PROPERTIES

Logistic map (LM) could be characterized as an one dimensional chaotic map with one control parameter r. When $r \in (0, 4)$, each iteration of LM produces an iterate $x_n \in (0, 1)$. Calculation of the first iterate x_1 requires an initial value $x_0 \in (0, 1)$ (also known as an initial point). Equation for the LM can be expressed as (1):

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

Equation (1) shows that successive iterates x_{n+1} use values of previous iterates x_n in their calculations. In order to suppress noticeable relationship between initial value x_0 and successive iterates, some of them are used only for providing other initial value. This concept is known as a *transient period* and its usual length is at least 1 000 iterates. With this length, the first 1 000 iterates are discarded (they are not used for an application of the LM) and the first usable iterate is x_{1001}.

2.1 Bifurcations, Self Similarity and Islands of Stability

The behavior of LM with various values of parameter r can be illustrated by a bifurcation diagram. An example of a bifurcation diagram obtained with 1 000 samples for $r \in (0, 4)$ is shown in Figure 2.

First bifurcation takes place at $r \sim 3$, where predictable trajectory of iterate values divides into two trajectories. Iterate values switch between these trajectories, called orbits based on value of parameter r. Second bifurcation happens at $r =$

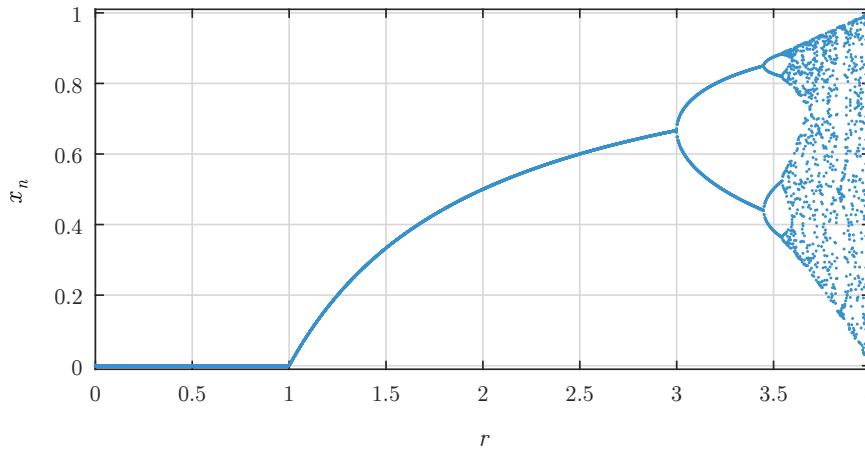


Figure 2. A bifurcation diagram of the logistic map

$1 + \sqrt{6} \sim 3.44949$. The distance between following bifurcations decreases each time by a ratio approximately given by Feigenbaum constant $\delta_f \sim 4.6692$ [30].

It needs to be pointed out that the shape of orbits after second bifurcation resembles shape of orbits after first bifurcation. This is one example of *self similarity* or *fractal behavior* caused by the LM.

The point where $r \sim 3.56995$ is considered as "an onset of chaos" [31]. However, the iterates still do not obtain values from whole interval of $(0, 1)$. Moreover, there are some areas known as *islands of stability* where increasing value of r does not cause multiplication of orbit amount, but the amount of orbits is significantly decreased. The self similarity of LM and an island of stability around $r = 1 + \sqrt{8} \sim 3.82843$ are illustrated in Figure 3 on a magnified part of the bifurcation diagram. The magnified part contains 1 000 samples for $r \in (3.5, 4)$.

2.2 Measurement of Chaotic Behavior

Quantification of chaotic behavior is important for comparing various chaotic maps. The dynamics of chaotic maps are usually expressed by values of maximal Lyapunov exponents (LEs) λ_{max}. As the LM is only one dimensional chaotic map, it does have only one LE $\lambda_{max} = \lambda$.

The basic idea behind computation of LEs investigates relationship of two orbits: the first one starts with an initial point p_0 and initial point for the second orbit denoted as $p_0 + \delta_0$ is achieved by applying small perturbation δ_0 to the point p_0. If orbits for these two initial points diverge from each other over time ($\lambda > 0$), investigated system is considered as chaotic. Otherwise, when the orbits are converging to each other ($\lambda < 0$), it may indicate that investigated system at a certain point tends to have periodic orbit or a fixed point. These two basic situations are displayed in Figure 4. Generally speaking, the higher

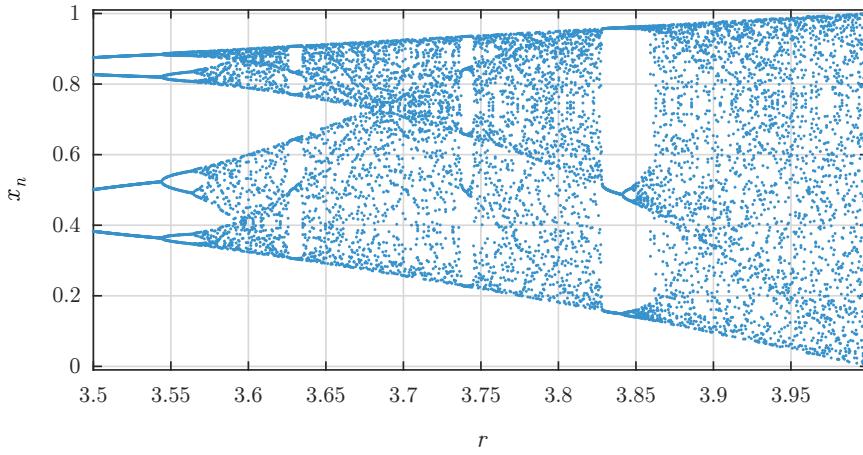


Figure 3. An illustration of self similarity and islands of stability

value of positive λ is, the better chaotic behavior can be expected from chaotic map.

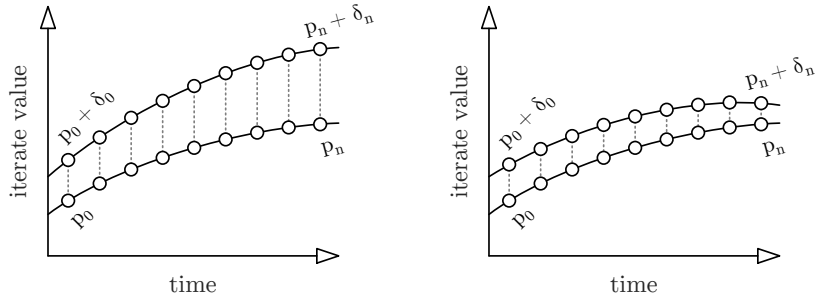


Figure 4. An illustration of diverging and converging orbits

In general, the values of λ can be computed via (2). Please note that practical computations are limited to finite amount of iterates n. Therefore, the resulting value of λ is only an estimation [32]. Commonly used values of n are 1 000 or 10 000 iterates.

$$\lambda = \frac{1}{n} \cdot \left| \frac{\delta_n}{\delta_0} \right| = \frac{1}{n} \cdot \ln|(f^n)'(x_0)| \ \sim \lim_{n\to\infty} \frac{1}{n} \cdot \sum_{i=0}^{n-1} \ln|f'(x_i)| \tag{2}$$

where brackets $|a|$ denote absolute value of a, δ_0 is an initial perturbation – an initial difference between orbits, δ_n is the difference between orbits after n iterations, $\ln(b)$

is a natural logarithm of b, (f^n) is a value of function f at point n, $(f^n)'(x_0)$ is a derivation of f^n with respect to x_0 and i is the iterate index number.

The calculations of λ for the LM substitute arbitrary function f with (1) that could be rearranged and its derivation with respect to x_i equals to $r(1-2x_i)$. Therefore, the estimation of LE for the LM λ_{LM} could be expressed as (3):

$$\lambda_{LM} \sim \lim_{n\to\infty} \frac{1}{n} \cdot \sum_{i=0}^{n-1} \ln|r(1-2x_i)|. \tag{3}$$

Please note that the value of λ_{LM} depends on both r and x_i which is a substitution for x_n from (1). While the dependence on r is pretty clear from the bifurcation diagram (see Figure 2 as the amount of bifurcations increases with greater values of r), the dependence on x_n is not that clear. It is caused by the fact that x_n is a result of n iterations of the LM which started with an initial value of $x_0 \in (0, 1)$. Various values of x_0 therefore can affect the behavior of the LM [32].

The obtained estimated values of LE for the LM with $x_0 = 0.5$, $r \in \langle 0, 4\rangle$ and $n = 1\,000$ are displayed in Figure 5. The interval for r is represented by 40 000 samples. The red line at $\lambda = 0$ marks the boundary which determines if the behavior of the LM is considered as chaotic or not. Please note that the predictable regions of the bifurcation diagram (see Figures 2 and 3) result in negative values of λ.

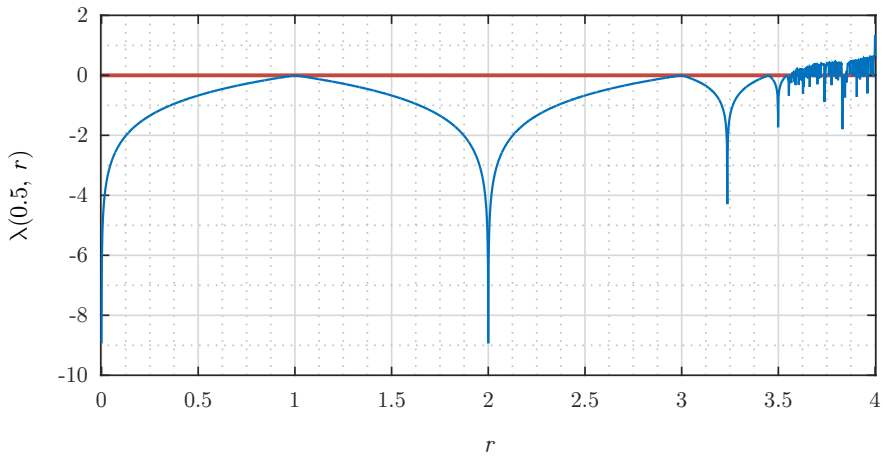


Figure 5. Estimated Lyapunov exponents for $r \in \langle 0, 4\rangle$

A detail of the estimated values of LE for $x_0 = 0.5$, $r \in \langle 3.5, 4\rangle$ and $n = 1\,000$ is shown in Figure 6. The interval investigated in Figure 6 uses finer resolution as it contains 50 000 samples (equivalent to 400 000 samples for $r \in \langle 0, 4\rangle$). It needs to be pointed out that the spike around island of stability at $r = 1 + \sqrt{8} \sim 3.82843$ resembles spikes visible in Figure 5. This is also caused by self similarity of the

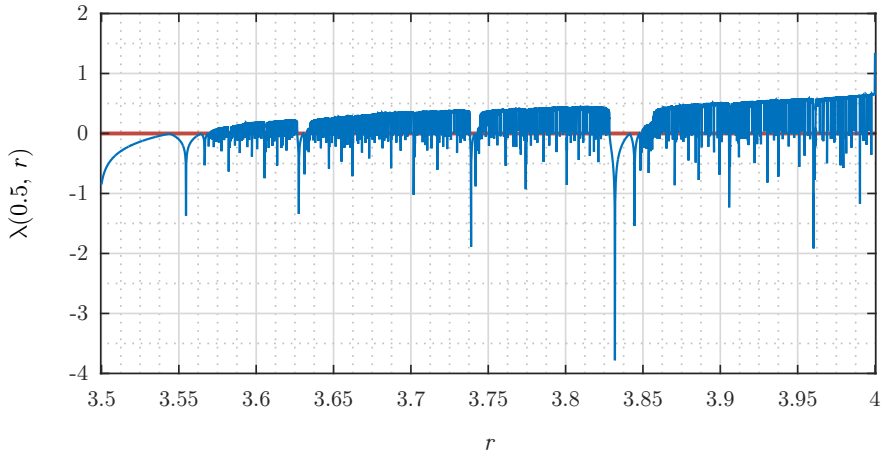


Figure 6. Estimated Lyapunov exponents for $r \in \langle 3.5, 4 \rangle$

LM [31]. Therefore usage of finer resolution could reveal some spikes that are present at values of r that could be otherwise not sampled.

Please note that while the equation of the LM (1) is given for $r \in (0, 4)$, the plot of estimated LE in Figure 4 uses $r \in \langle 0, 4 \rangle$. This is due to fact that global minimum and global maximum of the estimated LE values are located on the boundary points of this interval. Some researchers use $\lambda_{max} \sim 1.3447$ which is present for $x_0 = 0.5$ at $r = 4$ with $n = 1\,000$, however this value of r could cause occurrence of a fixed point $x_n = 0$ in systems with finite precision (used in a majority of practical applications).

With usage of the *double precision* data type defined by IEEE 754 standard [33], the decimal part of number is represented by 52 bits. This gives a precision of $\log_{10} 2^{52} \sim 15.6536$ decimal places, therefore double precision data type reliably represents only numbers with 15 or less decimal places. If some number has more decimal places, it is rounded to the closest double precision value.

The largest value of λ for $x_0 = 0.5$, $r \in (0, 4)$ and $n = 1\,000$ in the double precision data type is located at $r = 4 - 9 \cdot 10^{-15}$ and its estimated value is approx. 0.67601. This measurement used resolution equivalent to $4 \cdot 10^{16}$ samples for the interval of $r \in \langle 0, 4 \rangle$, therefore the set of possible x_n has more samples than the amount of numbers in interval $(0, 1)$ represented by the double precision data type.

The effect of various initial values x_0 on estimated values of λ is illustrated in Figure 7. This measurement used amount of samples equivalent to $4 \cdot 10^{13}$ for $r \in \langle 0, 4 \rangle$ and $n = 1\,000$. Please note that the two used values of x_0 cause different behavior of the LM only around a "self similar window". Also the values of λ were estimated with relatively fine resolution, coarser resolutions with less samples could suppress some areas with significant spikes.

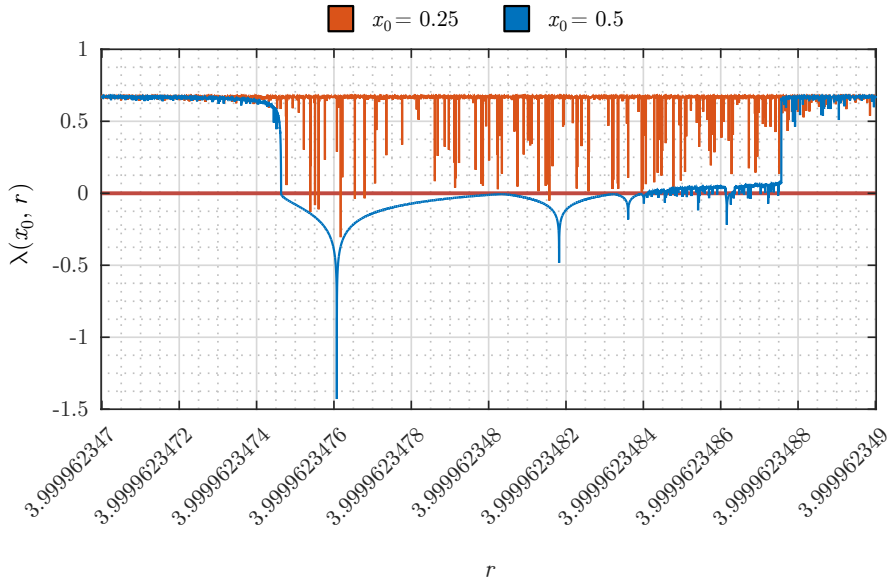


Figure 7. Dependence of estimated Lyapunov exponents on initial value x_0

2.3 Periodic Cycles and Fixed Points

As it was already mentioned, negative values of λ could lead to inappropriate behavior of the LM. One example of this behavior can be demonstrated using the local minimum of λ for $x_0 = 0.5$ from Figure 7. Successive iterates generated with parameter r belonging to this minimum of λ are shown in Table 1.

$x_0 = 0.5$, $r = 3.999962347607499$		
$x_1 = 0.999990586901875$	$x_6 = 0.009607660983953$	$x_{10} = x_1 = 0.999990586901875$
$x_2 = 0.000037651683653$	$x_7 = 0.038061057061643$	$x_{11} = x_2 = 0.000037651683653$
$x_3 = 0.000150599646393$	$x_8 = 0.146448273443029$	$x_{12} = x_3 = 0.000150599646393$
$x_4 = 0.000602302194975$	$x_9 = 0.500000000000973$	$x_{13} = x_4 = 0.000602302194975$
$x_5 = 0.002407735043706$		$x_{14} = x_5 = 0.002407735043706$

Table 1. An example of periodic cycles generated by logistic map

The situation presented in Table 1 is known as a *periodic cycle* of the LM. The presented example is caused by finite precision as a value of iterate x_9 is very similar to the value of x_0 and while their successive iterates are different in systems with higher precision, double precision systems represent them both with the same value that eventually becomes $x_{10} = x_1$. Situations like this are undesirable in cryptographic applications. In this case the length of this periodic cycle is just 9 iterates.

Analytical solution for finding out periodic cycles and their length for arbitrary values of r and x_0 has not been proposed yet. Persohn and Povinelli provided an overview of amount of periodic cycles with certain length, however, only for the single precision data type [10]. Therefore, possible occurrence of periodic cycles in the LM could be indicated only by negative values of λ.

Another issue with the LM is existence of *fixed points* for $x_n \in (0, 4)$. The fixed points do not change their values in successive iterations which could be expressed mathematically as $x_n = x_{n+1} = x_{n+2} = \cdots = x_{n+m}$ where m is amount of iterates computed after x_n. The fixed points for the LM could be obtained analytically by setting x_{n+1} to x_n in (1). The modified equation has two solutions: x_n equals either 0 or $1 - \frac{1}{r}$. While the first case is easily suppressed by using $x_n \in (0, 4)$, the second case is a bigger issue as value of this fixed point changes depending on used r. The issue with this fixed point can be illustrated by an example provided in Table 2.

$x_0 = 0.25$, $r = 4 - 10^{-15}$, fixed point at $x_n = 1 - \frac{1}{r} \sim 0.75$
$x_1 = 0.75$
$x_2 = x_1 = 0.75$
$x_3 = x_2 = x_1 = 0.75$

Table 2. An illustration of a fixed point of logistic map

An interesting thing about the example presented in Table 2 is that the value of $1 - \frac{1}{4-10^{-15}}$ is considered to be equal to $1 - \frac{1}{4} = 0.75$. In this case the small difference could not be preserved due to finite precision of double precision data type.

The issue with fixed points needs to be solved for enabling proper usage of the LM for encryption. The probability of "hitting" a fixed point among all other iterate values may seem low, but after reaching the value of the fixed point, all successive iterates will have the same value.

2.4 Relation Between Successive Iterates, Their Distribution and the Need for Quantization

As it is visible in (1), the LM is an iterative function as $x_n = f(x_{n-1})$. Equation (1) could be rearranged and there are two solutions for calculating value of previous iterate x_{n-1} from value of current iterate x_n with known value of parameter r (4):

$$x_{n-1} = \frac{1}{2}\left(1 \mp \sqrt{1 - 4\frac{x_n}{r}}\right). \tag{4}$$

The dependence of iterate values on values of the previous iterates could be also illustrated by a Poincaré plot. An example of this plot is shown in Figure 8 where a sequence of 2 000 iterates was computed with $r = 4 - 10^{-15}$ and $x_0 = 0.5$.

Two dimensional Poincaré plots have values of actual iterate x_n on their vertical axis and values of previous iterates x_{n-1} on their horizontal axis. If a horizontal line representing x_n would be drawn over the plot, two perpendicular lines drawn at

interceptions with the parabola lead to values of x_{n-1}. Figure 8 shows the example presented in Table 2, where iterate value $x_n = 0.75$ could be obtained from two previous iterate values $x_{n-1,1} = 0.25$ and $x_{n-1,2} = 0.75$ with usage of $r = 4 - 10^{-15}$.

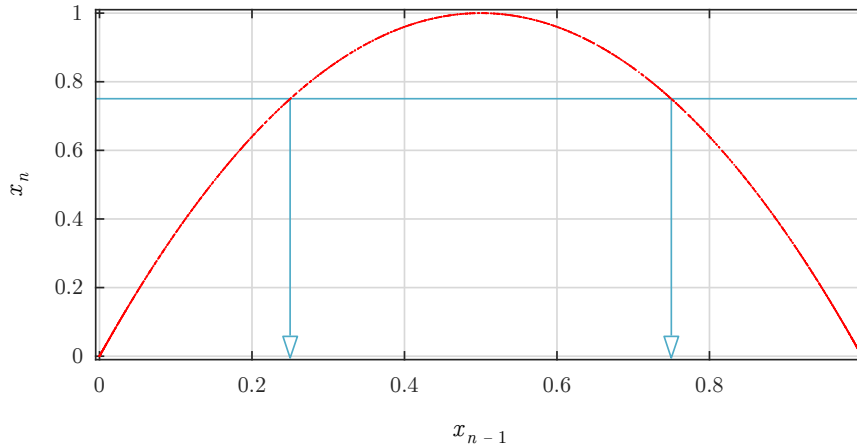


Figure 8. A Poincaré plot showing relation between two successive iterates

While this relation is crucial for computing successive iterates, it needs to be suppressed for usage in cryptographic applications. The possibility of reconstructing previous iterate values from one iterate is investigated by *phase space reconstruction* attacks [34, 35], however none of the proposed attacks could be practically used yet.

Other important feature of a set of computed iterates is their distribution among the interval $(0, 1)$. Ideally, the distribution should be uniform. However, as it was already mentioned and as it is visible in the bifurcation diagrams (see Figures 2 and 3), some values of r do not produce iterate values x_n that cover whole interval $(0, 1)$. An illustration of this case is shown in Figure 9 where blue bins use $r_1 = 3.75$, $x_0 = 0.5$ and transient period of 1 000 iterates in order to generate 10^6 iterates. The second set of bins, red ones were generated by the same settings of the LM except for using $r_2 = 4 - 10^{-15}$. The histogram has 20 bins with equal size.

Figure 9 shows that the first non-zero bin for $r_1 = 3.75$ is located in interval $(0.2, 0.25\rangle$ and the last one is present in interval $(0.9, 0.95\rangle$. The four bins at the beginning and the last bin are empty in the case of r_1 while they are populated for r_2. Also, please note that the bins close to boundaries of the histogram have bigger amount of samples and that the histogram for r_2 is approximately symmetric with respect to its center. These findings could be used for analysis, so they need to be suppressed for enabling usage of the LM for image encryption.

There is also a problem with the usage of computed iterate values in image encryption algorithms. As the iterate values x_n are decimal numbers from interval $(0, 1)$ and the image encryption algorithms process images either as a set of bytes (256 possible integer values) or bits (2 possible integer values), it is useful to perform

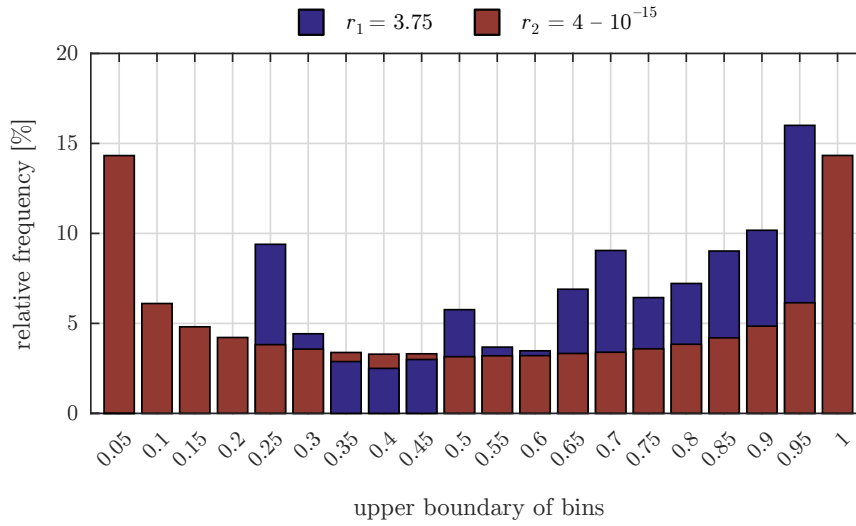


Figure 9. Distribution of iterate values for two different values of parameter r

quantization of the iterate values. Multiple ways of quantization were proposed, some of our prior work even solves the already mentioned issue of relation between successive iterates [24].

3 PROPOSED SOLUTION

Based on the findings from analysis of the LM, we tried to mitigate its drawbacks by using a following set of rules:

- usage of combinations of parameter r and initial value x_0 that have positive λ, so the occurrence of periodic cycles should be suppressed,
- the fixed points should be suppressed by alternating two values of r – each value of r will have its counterpart, denoted as r_c (from "complementary r") which would have a fixed point located at other value of x_n,
- each pair of values of r and r_c should be selected in a way that results in quick divergence form their respective fixed points,
- the elements of resulting PR sequences should not have any distinctive relation between pairs of successive elements and their distribution should be close to uniform (this point has been partially solved in our prior works [24, 35]),
- implementation of these rules should have minimal impact on the computational complexity of image encryption algorithms.

3.1 Suitable Parameters for Logistic Map in Double Precision Data Type

In order to meet given requirements, we chose to use interval of parameter r that obtains positive values of λ and does not have many significant spikes. As most image encryption algorithms use double precision data type, the upper boundary of chosen interval was set the closest it can get to $r = 4$, so the upper boundary was chosen as $4 - 10^{-15}$.

The selection of a lower boundary was motivated by enabling as large interval of parameter r as possible. The size of this interval is important, as key elements are used directly for computation of values of r. Therefore, the larger interval of r enables larger amount of usable keys (known as a key space).

The keys are represented in a hexadecimal or binary form. Conversion between these two forms is simple as one hexadecimal character contains $16 = 2^4$ binary characters. For enabling usage of integer amount of key elements as values of r, also the number of samples in interval of r needs to be a power of 16. The chosen value of number of samples was $16^4 = 65\,536$ as bigger powers of 16 resulted in intervals of r that have significant spikes in plots of their λ. The plot of λ estimated for the resulting interval of $r \in \langle 4 - 65\,536 \cdot 10^{-15}, 4 - 10^{-15} \rangle$, $x_0 = 0.5$ and $n = 1\,000$ is shown in Figure 10. While the analysis was done with resolution equivalent to $4 \cdot 10^{16}$ samples for interval $\langle 0, 4 \rangle$, the plot uses coarser resolution equivalent to $4 \cdot 10^{13}$ samples for the same interval for the sake of better readability.

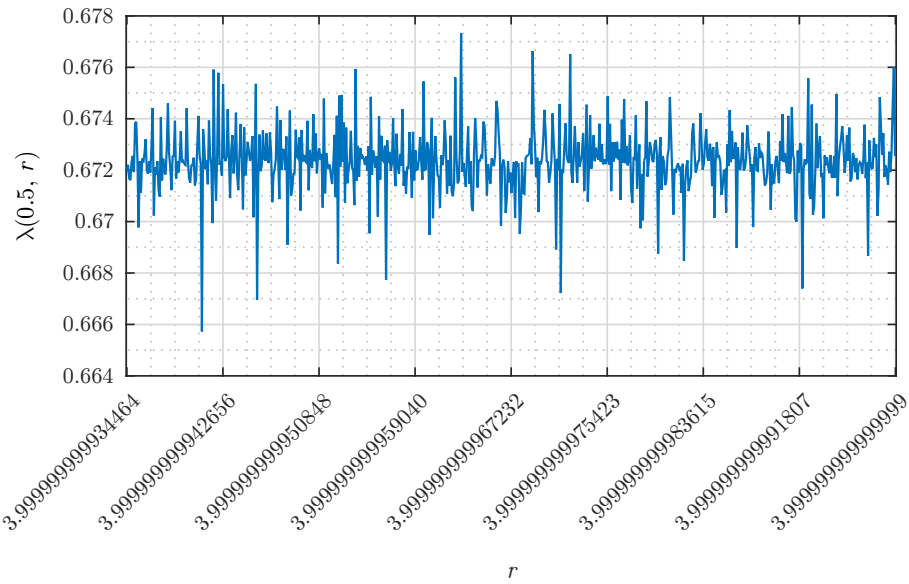


Figure 10. Lyapunov exponents for chosen interval of values for parameter r

The reason behind the choice of x_0 being fixed at 0.5 is the rather small effect on the resulting iterate values x_n. Its impact is present mostly in areas with smaller or negative λ as it was explained in Section 2.2 and especially in Figure 7.

As 4 hexadecimal characters are mapped to 65 536 very similar values of r, the impact of a slightly different key on the resulting iterate values x_n might raise some concern. As it was pointed out in Section 2.1, the LM uses a concept of the transient period. In the case of our proposal, its length is set to 100 iterates for each used value of r. As our solution alternates between r and its complement r_c, the total length of the transient period for each used r is 200 iterates. The resulting values of x_{200} obtained by similar values of r are displayed in Table 3.

	$r_c = r - 10^{-5}$		
used r	$4 - 32\,768 \cdot 10^{-15}$	$4 - 32\,767 \cdot 10^{-15}$	$4 - 32\,766 \cdot 10^{-15}$
x_0	0.5	0.5	0.5
x_1	0.999999999991808	0.999999999991808	0.999999999991809
x_2	0.000000000032768	0.000000000032767	0.000000000032766
x_3	0.000000000131072	0.000000000131068	0.000000000131063
x_4	0.000000000524286	0.000000000524271	0.000000000524250
...	...	...	...
x_{200}	0.845629440355832	0.933763238939059	0.189452959055630

Table 3. Different values of x_{200} obtained with similar values of r

The results presented in Table 3 show that even small differences in value of r result in rather big differences in values of x_{200}. This is a property of chaos, known as a butterfly effect [31].

Image encryption algorithms usually utilize multiple PR sequences for their operations. Therefore, multiple values of r are necessary which brings up the key length to multiples of 4 hexadecimal characters or 16 bits. However, current encryption algorithms utilize key lengths that exceed even 256 bits. In order to enable usage of longer keys (and thus larger key space), our proposal uses multiple values of r and their respective r_c during the transient period. These are denoted as r_i and $r_{c,i}$ and their amount depends on desired size of the key space.

The example of multiple r_i and $r_{c,i}$ used in an image encryption algorithm is illustrated in Table 4. In this case, the algorithm uses 8 values of r_i and their respective $r_{c,i}$ in order to produce 4 final PR sequences. The rule which determines purpose of individual key parts is also known as a *key schedule*.

This example uses 8 values of r_i and their respective $r_{c,i}$. As each r_i is computed from 16 bits of key, the total length of a key for the example is $8 \cdot 16 = 128$ bits. As all of the r_i and $r_{c,i}$ are used during the transient period, even a small change in one part of the key affects all generated PR sequences. This concept is known as *key diffusion* [35, 36]. Total size of the transient period in this case is 1 600 iterates (100 for each of the r_i and $r_{c,i}$).

sequence	indexes i of r_i and $r_{c,i}$ used during the transient period	indexes i of r_i and $r_{c,i}$ used after the transient period
seq_1	1 to 8	1
seq_2	2 to 8, then 1	2
seq_3	3 to 8, then 1 and 2	3
seq_4	4 to 8, then 1 to 3	4

Table 4. An example of a simple key schedule

3.2 Suppression of Periodic Cycles and Fixed Points

As it was mentioned earlier in Section 2.3, the occurrence of periodic cycles could be greatly suppressed by usage of values of r that have positive λ. Analysis of the periodic cycles is quite computationally exhaustive and has been performed only for the single precision data type yet [10]. On the other hand, the values of fixed points could be easily derived for each value of r as it was already shown.

In order to suppress these drawbacks, our proposal alternates between two related parameter values r and r_c. Each odd iteration of the LM uses r, while each even iteration utilizes respective r_c. This approach should be able to solve both mentioned problems.

Because the rules proposed in our approach should have minimal impact on the computational complexity of whole image encryption algorithms, it is important to have a fast way for computing r_c from r. As the chosen interval for value of r is $\langle 4-65\,536\cdot 10^{-15}, 4-10^{-15}\rangle$, we decided to select interval for r_c as $\langle 4-10^{-5}-65\,536\cdot 10^{-15}, 4-10^{-5}-10^{-15}\rangle$. This interval of r_c should provide several advantages: first of all the chosen r_c should have favorable values of λ as they are still close to $r = 4$. The plot of values of λ for the mentioned interval of r_c with $x_0 = 0.5$, $n = 1\,000$ and resolution equivalent to $4 \cdot 10^{13}$ samples for interval $\langle 0, 4\rangle$ is shown in Figure 11.

Secondly, the selection of interval for r_c leads to a simple formula for its calculation from r (5):

$$r_c = r - 10^{-5}. \tag{5}$$

Lastly, as the values of r_c and r are different on their fifth decimal place, a slightly adapted quantization block presented in our previous work [24] should be able to ensure quick divergence from any fixed points. An example of fixed points for one pair of parameters r and r_c is shown in Table 5.

parameter	r	r_c
value	$4-10^{-15}$	$4-10^{-5}-10^{-15}$
fixed point at $1-\frac{1}{r}$	0.75	0.749999374998438

Table 5. Fixed points for a pair of parameters r and r_c

While the difference between fixed points displayed in Table 5 may seem negligible, it is big enough to ensure quick divergence from fixed points with usage of

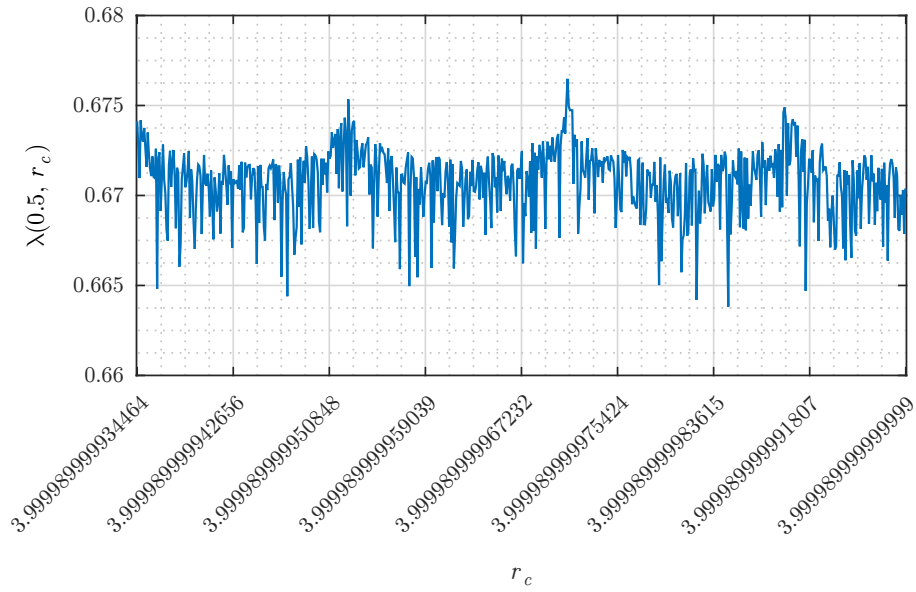


Figure 11. Lyapunov exponents for chosen interval of values for parameter r_c

suitable quantization approach. If value of r_c would be obtained by modification of a higher decimal place, the plot of resulting λ could contain significant spikes and it could also negatively affect the distribution of iterate values after their quantization.

3.3 Quantization and Successive Iterate Relation

In our previous work, it was found out that the non-uniform distribution of iterate values x_n is caused mainly by higher amount of iterate values that are close to the boundaries of interval $(0, 1)$ [24, 35]. This could be easily fixed by dismissing several decimal places of iterate values after whole sequence of iterates has been computed.

A quantization technique that dismisses first four decimal places of each iterate value x_n was proposed in [35]. Its equation could be expressed as (6):

$$x'_n = \left\lfloor (max_{val} + 1) \cdot \left(10^4 \cdot x_n \pmod 1\right) \right\rfloor \tag{6}$$

where x'_n is a sequence of quantized iterate values, brackets $\lfloor a \rfloor$ denote the greatest integer less than or equal to a, n is a sequential number of an iterate and max_{val} is maximal allowed value after quantization.

The effects of quantization by (6) are shown in Figures 12 and 13. The sequences for these plots were made in the same way as for Figures 8 and 9. Figure 13 uses only one value of $r = 4 - 10^{-15}$ so please compare it only with red bars from Figure 9.

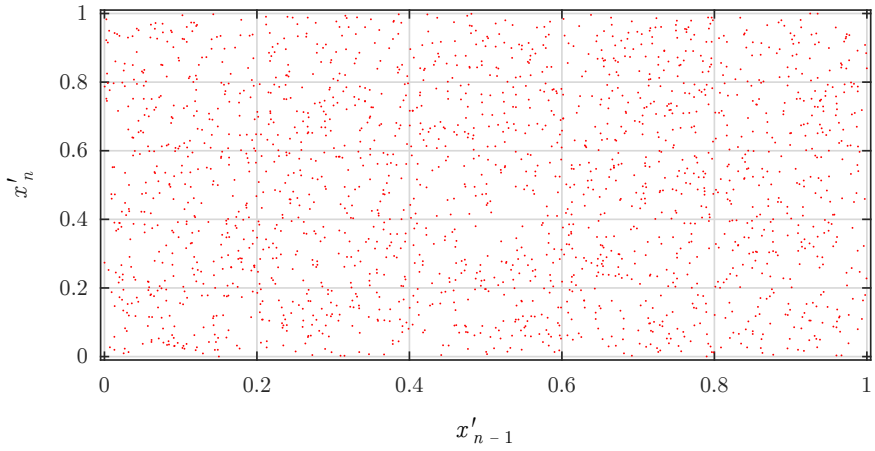


Figure 12. A Poincaré plot of iterates after quantization

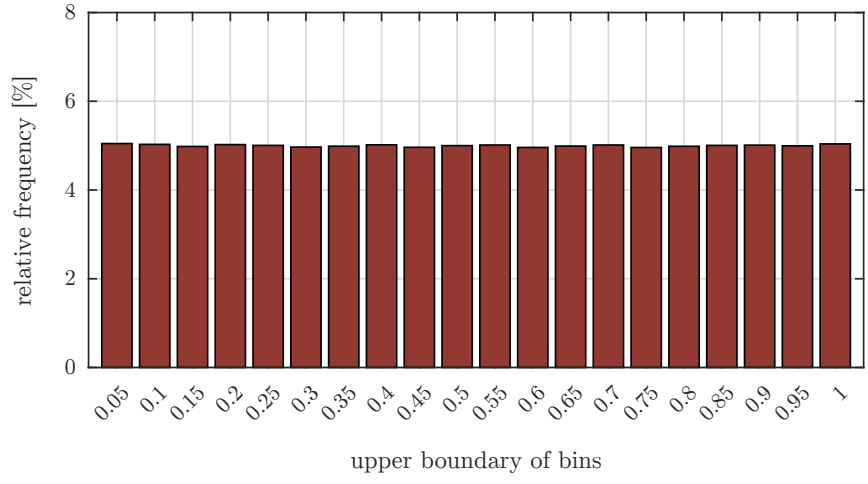


Figure 13. Distribution of iterate values after quantization

Please note that the quantization by using (6) suppresses relations between successive iterates. While the relations can be found in a sequence of computed iterates, the removal of the first four decimal places by quantization results in a more uniform distribution of the points shown by a Poincaré plot in Figure 12. Also, the amount of points on the horizontal lines belonging to x'_n is not clear so any attempts to obtain x'_{n-1} from x'_n are even harder.

The histogram of values of quantized iterates displayed in Figure 13 shows that the distribution is much more uniform than in the previous case (see red bars on

Figure 9). Removal of more decimal places did not yield better results in terms of more uniform distribution of iterate values [35, 24].

As it was mentioned, the quantization technique greatly affects the speed of divergence from fixed points. An example is presented in Table 6. Please note that the same values of x_0 and r were used in Table 2. However the previous case did not alternate between r and r_c and therefore ended in an occurrence of a fixed point.

$r = 4 - 10^{-15}$, fixed point at $x_n = 0.75$ $r_c = 4 - 10^{-5} - 10^{-15}$, fixed point at $x_n = 0.749999374998438$			
iterate	value	value after removal of first four decimal places	value after quantization by (6), $max_{val} = 255$
x_0	0.75	–	–
x_1	0.75	0.999999999998181	255
x_2	0.749998125000000	0.981250000002547	251
x_3	0.750003749985937	0.037499859367927	9
x_4	0.749990624990627	0.906249906268386	231
x_5	0.750018749667183	0.187496671829649	47
x_6	0.749960624353186	0.606243531856308	155
x_7	0.750078745091862	0.787450918623108	201
...	...	...	...
x_{50}	0.202236261140394	0.362611403942537	92
x_{51}	0.645347023281394	0.470232813938310	120
x_{52}	0.915494682550523	0.946825505234301	242
x_{53}	0.309456675088959	0.566750889588548	145
...	...	...	...

Table 6. An illustration of quick divergence from fixed point

Values reported in Table 6 show that the first usage of r_c which has a different fixed point introduces a slight divergence from the fixed point for r. Also, if the value of computed iterate would be equal to the fixed point for r_c, the alternation to r would help to provide a small perturbation which would result in a divergence.

There are two interesting facts about values presented in Table 6. Firstly, the value of $x_1 = 0.75$ after removal of the first 4 decimal places becomes ~ 0.99999. This is due to fact that x_1 is represented by its closest possible representation in double precision data type which is actually 0.74999999999999978. The first four decimal places of this number are removed and the other places are shifted towards the decimal point. The last two decimal places (78) are changed in order to meet the closest possible number after the quantization in double precision data type (8181).

The second interesting fact is that the values of x_1, x_2 and x_3 are close to the boundaries of set $\{0, 1, \ldots, 255\}$. However, after rather small amount of iterates, the distribution becomes more uniform.

4 EXPERIMENTAL RESULTS

Usage of the proposed rules should result in a more suitable behavior of the LM (1) and therefore it can positively affect the performance of whole image encryption algorithms. On the other hand more complex computations can raise computational complexity. In order to investigate these assumptions, the proposed set of rules was applied to one of our prior image encryption algorithms published in [24]. The mentioned algorithm was already highly optimized for achieving good performance, so the resulting values of numerical parameters should clearly show the effect of the proposed set of rules.

All performed measurements were done on a PC with 2.5 GHz CPU, 12 GBs of RAM in computing environment MATLAB R2015a on Windows 10 OS. A set of experimental plain images *lena*, *lenaG* and *peppersG* is shown in Figure 14. Their parameters are described in Table 7. A set of experimental keys K_1 and K_2 are presented in Table 8. The first one was derived from binary representation of decimal part of number π. The second key has a minimal difference, as the sixth hexadecimal character is changed from 0 to 1.

lena *lenaG* *peppersG*

Figure 14. A set of experimental plain images

image	*lena*	*lenaG*	*peppersG*
height [px]	512	512	512
width [px]	512	512	512
color depth [bits/px]	24	8	8

Table 7. Parameters of used plain images

key	value
K_1	0× C4 EB 50 BC 0E C5 EB 50 BC 0E C5 EB 50 BC 0E C5
K_2	0× C4 EB 51 BC 0E C5 EB 50 BC 0E C5 EB 50 BC 0E C5

Table 8. A set of experimental keys

4.1 Brief Description of Used Algorithm

The image encryption algorithm proposed in [24] processes images with arbitrary resolution higher than 16×16 pixels. This restriction is caused by Mojette transform (MoT) that is used during a plaintext related chaining stage. The color depths of input images could be either 8 or 24 bits per pixel. Used key is 128 bits long, entered in a hexadecimal form and it is used for generation of six PR sequences by means of the LM (1).

Described image encryption algorithm has 5 stages. The first and the last stage combine image pixel intensities with PR sequences generated by the LM (1). Since these combinations protect the other stages against attacks, the security of whole examined image encryption algorithm relies heavily on the first and last stage.

The second stage rearranges image pixels in order to suppress correlation between adjacent image pixel intensities. The rearrangement is carried out by cyclic shifts in individual columns of the processed image followed by cyclic shifts in the rows of the image.

Stages three and four introduce dependencies between pixel intensities in order to increase sensitivity to small differences between various plain images. While the third stage scans and processes the image in all four possible directions for spreading the differences, the fourth stage uses pixel intensities as parameters for the MoT. The previously loaded pixel intensities choose values that are combined with actually processed pixel intensities.

The decryption algorithm utilizes analogous stages in a reversed order. More details about this solution could be found in [24].

4.2 Modifications of Used Algorithm

Due to previous optimization of the used algorithm a rather small amount of modifications was necessary for adhering to the proposed set of rules. First of all, the concept of shifting between values of r and r_c was introduced. The intervals for these parameters were set as it was recommended in Section 3.1: $r \in \langle 4 - 65\,536 \cdot 10^{-15}, 4 - 10^{-15} \rangle$ and $r_c \in \langle 4 - 10^{-5} - 65\,536 \cdot 10^{-15}, 4 - 10^{-5} - 10^{-15} \rangle$.

Secondly, the whole key schedule was reworked as the original approach used parts of the key with 8 hexadecimal characters. This was not longer possible as the 65 536 possible values of r or r_c are represented only by 4 hexadecimal characters. The eight values of r and their respective values of r_c are computed by (7):

$$\begin{aligned} r_i &= 4 + 10^{-15} \cdot (K_i - 65\,536), \\ r_{c,i} &= r_i - 10^{-5} \end{aligned} \tag{7}$$

where K_i represents a decimal form of i^{th} four hexadecimal character group from key K, $i = 1, 2, 3, \ldots, 8$. The conversion from a hexadecimal to a decimal form uses big endian ordering scheme.

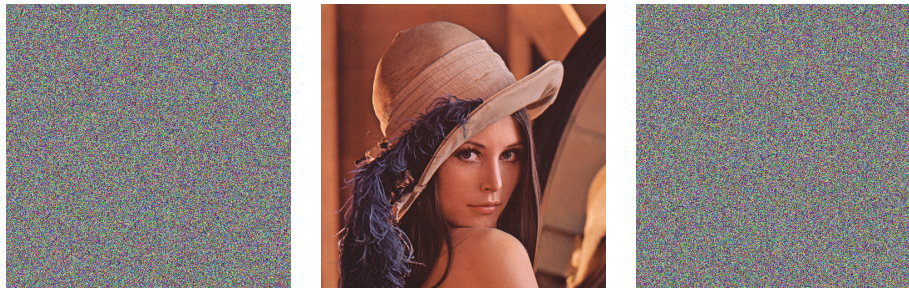
The key schedule used in the modified algorithm is presented in Table 9. Each parameter is used for 100 iterations in the transient period (each pair of r ad r_c therefore for 200 iterations). The total amount of iterations during the transient period is extended from previously used 1 000 iterates to 1 600.

sequence	indexes i of r_i and $r_{c,i}$ used during the transient period	indexes i of r_i and $r_{c,i}$ used after the transient period
seq_1	1 to 8	1
seq_2	2 to 8, then 1	2
seq_3	3 to 8, then 1 and 2	3
seq_4	4 to 8, then 1 to 3	4
seq_5	5 to 8, then 1 to 4	5
seq_6	6 to 8, then 1 to 5	6

Table 9. Key schedule used in the modified algorithm

Finally, the quantization does not shift decimal places of iterates by five places anymore. In order to provide both quick divergence from fixed points and suppression of relations between successive iterations, the quantization shifts decimal places by four places by (6) and r_c is set in a way that it is different from r on fifth decimal place. These steps were selected because of suitable values of λ for r and r_c with x_0 fixed at 0.5. More details on this issue were presented in Section 2.2.

A simple verification that the proposed set of rules did not significantly harm the basic features of the original image encryption algorithm can be performed through encryption and decryption an image. The example presented in Figure 15 used plain image *lena* and key K_1. The first decryption used correct key and its result is the same as the plain image. However second decryption deliberately used a wrong key (K_2) and the decryption results in an image similar to noise.



encrypted by K_1 | encrypted by K_1, decrypted by K_1 | encrypted by K_1, decrypted by K_2

Figure 15. Preserved key sensitivity of the modified algorithm

4.3 Used Numerical Parameters

There are several commonly utilized numerical parameters that can be used for measuring performance of the image encryption algorithms. This paper utilizes values of correlation coefficients ρ, entropy H, Number of Pixel Change Ratio (NPCR) and Unified Average Changing Intensity (UACI). The last two parameters were described by Wu et al. in [37].

In addition to these parameters, this paper also utilizes a measure of computational complexity known as number of cycles necessary for an operation (either encryption or decryption of an image). This measure takes into account resolution and color depth of the testing images and also a configuration of used PC.

Values of *correlation coefficients* ρ are computed separately for each color plane in three directions – horizontally (denoted as ρ_h), vertically (ρ_v) and diagonally (ρ_d). The computation of ρ is usually done on a finite amount of randomly chosen pixel pairs. Each pixel pair contains intensities of two image pixels that are adjacent in a chosen direction. In general, ρ are computed by using (8):

$$\rho = \frac{\sum_{pp=1}^{num_{pp}} \left(vec_1(pp) - \overline{vec_1}\right) \cdot \left(vec_2(pp) - \overline{vec_2}\right)}{\sqrt{\sum_{pp=1}^{num_{pp}} \left(vec_1(pp) - \overline{vec_1}\right)^2 \cdot \sum_{pp=1}^{num_{pp}} \left(vec_2(pp) - \overline{vec_2}\right)^2}} \; [-] \tag{8}$$

where $pp = 1, 2, \ldots, num_{pp}$ is an index of pixel pair, num_{pp} is total amount of pixel pairs (usually selected as 1 000), vectors vec_1 and vec_2 represent intensities of pixels from pixel pairs and $\overline{vec}$ denotes arithmetic mean of vector vec.

The entropy H as it was described by Shannon [38] can be viewed as a randomness measure of a information source. It is computed individually for each color plane of an image by applying (9):

$$H = -\sum_{in=0}^{2^L-1} p(in) \cdot \log_2\left(p(in)\right) \; [\text{bits/px}] \tag{9}$$

where L is a color depth of investigated color plane, in denotes intensity of image pixel and $p(in)$ stands for a probability of occurrence of pixel with intensity in. The ideal value of entropy H is close to the color depth of investigated color plane.

Each computation of NPCR and UACI utilizes two encrypted images E_1 and E_2. These were made by encryption with the same algorithm and the same key from plain images P_1 and P_2. The first plain image P_1 was arbitrarily chosen, while the second plain image P_2 is a copy of P_1 with a difference in an intensity of one image pixel. Furthermore, the size of this difference is minimal (only one intensity level). Values of NPCR and UACI therefore measure the differences done by encryption of two almost identical plain images.

The location of the difference introduced in plain image P_2 is usually randomly chosen. In order to suppress the impact of certain locations on the resulting values, the values of NPCR and UACI are presented as a mean of larger set of measurements (usually mean of 100 computed values).

Number of Pixel Change Ratio (NPCR) is calculated individually for each color plane of the investigated image by using (10):

$$NPCR = \frac{100}{h \cdot w} \sum_{l=1}^{h} \sum_{k=1}^{w} Diff_{mat}(l,k) \ [\%] \tag{10}$$

where h and w represent height and width of images E_1 and E_2, l and k are line and column indexes and $Diff_{mat}$ is a difference matrix, $Diff_{mat}(l,k) = 1$ if $E_1(l,k) \neq E_2(l,k)$, $Diff_{mat}(l,k) = 0$ otherwise.

Unified Average Changing Intensity (UACI) is also computed individually for each color plane of the tested image via (11):

$$UACI = \frac{100}{h \cdot w} \sum_{l=1}^{h} \sum_{k=1}^{w} \frac{|E_1(l,k) - E_2(l,k)|}{2^L - 1} \ [\%] \tag{11}$$

where brackets $|a|$ denote absolute value of a and L is a color depth of the investigated color plane.

Please note that while NPCR only sums amount of pixels with different intensities after encryption of P_1 and P_2, UACI also takes into account the size of individual differences. Also, the proposal of NPCR and UACI by Wu et al. [37] mentions expected values of the two parameters. The expected values are presented as intervals and they depend on a resolution of used images. If a computed value of NPCR or UACI belongs to the interval of the expected value, the resulting encrypted image can be considered as robust against differential attacks with certain confidence (determined by significance level α) [37].

For one of the most used significance levels at $\alpha = 0.001$ which results in confidence of 99.9 %, the intervals of NPCR and UACI for an image with resolution of 512×512 pixels are $(99.5717\,\%, 100\,\%\rangle$ and $(33.1594\,\%, 33.7677\,\%)$, respectively.

The *computational complexity* of the image encryption algorithms can be measured via time t_{oper} necessary for completing an operation (encryption or decryption) on an image. Usually, a mean of 100 repeated measurements is used for suppressing effect of some faster or slower times (e.g. due to other processes running on a testing PC). These means are denoted as t_{enc} for encryption times and t_{dec} for decryption times in the following text.

Configuration of the testing PC is taken into account by computing number of CPU cycles that are necessary for an operation with one byte of image data. The equation for this measure is given as (12):

$$cyc_{oper} = \frac{f_{CPU} \cdot t_{oper} \cdot 2^3}{h \cdot w \cdot d} \ [\text{cycles/B}] \tag{12}$$

where f_{CPU} is a clock frequency of CPU in the testing PC, t_{oper} is the operation (encryption or decryption) time in seconds, h and w stand for height and width of the used image, d is its color depth given in bits per pixel and a constant of

2^3 represents amount of bits in a byte. Please note that most image encryption algorithms utilize computational environments that use only one CPU core.

4.4 Measured Values and Discussion

The measured values of correlation coefficients ρ, entropy H, NPCR and UACI for plain images *lena*, *lenaG* and *peppersG* encrypted by keys K_1 and K_2 are included in Table 10. Please bear in mind, that this comparison is done only for investigating impact of the proposed set of rules, therefore the values for plain images (not encrypted) are not presented. The letters R, G and B stand for red, green and blue color plane. Symbol '–' denotes that the presented results are obtained for the only color plane of grayscale images.

image and color plane		key	ρ_h [–]	ρ_v [–]	ρ_d [–]	H [bits/px]	NPCR [%]	UACI [%]
algorithm from reference [24]								
lena	R	K_1	0.0004	0.0013	0.0005	7.9993	99.6101	33.4738
	G		−0.001	−0.0021	−0.0037	7.9994	99.6109	33.4742
	B		−0.0032	−0.002	0.0017	7.9992	99.6103	33.4746
	R	K_2	0.0033	0.0026	−0.001	7.9994	99.6105	33.4742
	G		0.0006	0.0015	−0.0016	7.9993	99.6113	33.4749
	B		0.0016	0.0013	0.003	7.9993	99.6101	33.4743
lenaG	–	K_1	0.0008	0.0005	−0.0004	7.9992	99.61	33.4714
		K_2	0.0015	0.001	−0.0012	7.9993	99.6099	33.4725
peppersG	–	K_1	0.0051	0.0007	−0.0001	7.9991	99.6106	33.4721
		K_2	0.0019	−0.0011	−0.0008	7.9991	99.6121	33.4729
algorithm from [24] improved by the proposed solution								
lena	R	K_1	0.0019	0.0015	0.0019	7.9993	99.6191	33.4908
	G		−0.0005	0.0021	0.0034	7.9993	99.6204	33.49
	B		−0.0018	0.0005	−0.0004	7.9993	99.6183	33.4873
	R	K_2	−0.0001	−0.001	−0.0006	7.9992	99.6202	33.4884
	G		0.0009	−0.0001	−0.0038	7.9993	99.6198	33.4894
	B		0.0001	0.0007	−0.0017	7.9992	99.621	33.4872
lenaG	–	K_1	0.0021	0.0006	0.0019	7.9992	99.6206	33.4867
		K_2	−0.0032	0.0013	0.0002	7.9993	99.6196	33.4904
peppersG	–	K_1	0.0003	−0.0001	−0.0008	7.9994	99.6187	33.4856
		K_2	−0.0001	−0.0034	0.0027	7.9993	99.618	33.4892

Table 10. Measured values of ρ, H, NPCR and UACI

The results presented in Table 10 show that difference in values of correlation coefficients ρ (computed from 1000 pairs of adjacent pixel intensities) did not change much by application of the proposed set of rules. Also, the change in values of entropy H might be viewed as negligible. This could be caused by a good performance of the original algorithm as it was discussed in the paper [24].

On the other hand, the values of NPCR and UACI were slightly improved. Please bear in mind, that there is only a slight improvement, but it could greatly help in the case of a differential attack. Also, since the proposed set of rules mitigates vulnerabilities of the LM, the possibility of other attacks should be suppressed. On top of that, all values of NPCR and UACI for both the original algorithm and its improved version belong to intervals of the expected values [37].

The increase of computational complexity is illustrated by values included in Table 11. It is clearly seen that application of the proposed set of rules has only a small impact on computational complexity of the whole image encryption algorithm. Furthermore, the used algorithm was highly optimized so any new modifications could result in a visible increase of computational complexity.

image	key	t_{enc} [ms]	increase [%]	t_{dec} [ms]	increase [%]	cyc_{enc} [cycles/B]	cyc_{dec} [cycles/B]
algorithm from reference [24]							
lena	K_1	436.2245	–	420.5649	–	1 386.72	1 336.94
	K_2	436.3845		420.3934		1 387.23	1 336.39
lenaG	K_1	126.2499	–	122.7206	–	1 204.01	1 170.35
	K_2	126.9473		122.6332		1 210.66	1 169.52
peppersG	K_1	127.1529	–	122.8078	–	1 212.62	1 171.19
	K_2	126.8485		122.7695		1 209.72	1 170.82
algorithm from [24] improved by the proposed solution							
lena	K_1	448.7342	2.87	429.9181	2.22	1 426.49	1 366.67
	K_2	447.9295	2.65	431.0329	2.53	1 423.93	1 370.22
lenaG	K_1	129.5253	2.59	126.5223	3.1	1 235.25	1 206.61
	K_2	130.9827	3.18	125.8389	2.61	1 249.15	1 200.09
peppersG	K_1	130.8076	2.87	125.5048	2.2	1 247.48	1 196.91
	K_2	129.9531	2.45	125.7442	2.42	1 239.33	1 199.19

Table 11. Measured values of computational complexity

The measured increase of the computational complexity was ranging approximately from 2.2 % to 3.2 % (with the worst case of almost 13 ms for a true color image with a resolution of 512×512 pixels). This is a relatively interesting result for as complex changes as those described in Section 4.2. This finding can also imply that well-tailored fixes or bigger patches could further improve the behavior of simpler chaotic maps like the LM.

4.5 Comparison with Some Other Proposals

The impact of the proposed set of rules on the LM could be pointed out by a comparison of numerical results with some other approaches. Some of the relatively new algorithms were selected for the comparison, with a strong emphasis on algorithms that use multidimensional systems. Usage of these systems in image encryption is interesting as some researchers directly relate the robustness of whole image encryp-

tion algorithm to behavior of its chaotic system given by its LEs. However, the finite precision and some bad implementations can degrade their results. Therefore, we would like to compare results obtained by our well-tuned algorithm based on a simple chaotic map with the numerical results and computational complexity of image encryption algorithms that utilize far more complex chaotic systems.

An approach using a cascade of two chaotic maps was described by Cao et al. in 2018 [25]. The authors discuss that in certain interval, the parameter of used chaotic system results in λ close to value of 6. A similar solution was proposed by Alawida et al. in 2019 [26], where resulting value of λ is close to 2. Moreover, these researchers performed a brief investigation regarding relation of successive iterates.

A paper by Sun et al. [28] from 2019 describes an algorithm that uses seven dimensional chaotic system. Also, the computational complexity of this algorithm is increased by usage of hash function for introducing a plaintext related operation.

Liu et al. published a paper [39] in 2020 that characterizes their algorithm as "fast". However they still use the combination of two relatively complex chaotic maps that negatively affect the resulting computational complexity.

The comparison of numerical results including the computational complexity given in amount of cycles necessary for an encryption of one byte of image data cyc_{enc} is shown in Table 12. The value in *italics* marks that it was the best among all compared approaches and value '–' stands for not reported measurement.

approach	ρ_h [–]	ρ_v [–]	ρ_d [–]	H [bits/px]	NPCR [%]	UACI [%]	cyc_{enc} [cycles/B]
plain image *lenaG*							
proposed	0.0021	*0.0006*	0.0019	*7.9992*	99.6206	33.4867	*1 235.25*
ref. [25]	0.0019	0.0012	*0.0009*	7.9973	99.6096	33.4574	9 402.59
ref. [26]	−0.0017	−0.0084	−0.0019	7.9975	99.62	*33.505*	24 644.78
ref. [39]	0.0106	−0.0012	*0.0009*	–	*99.6216*	33.4994	3 484.18
plain image *peppersG*							
proposed	*0.0003*	−0.0001	−0.0008	*7.9994*	99.6187	33.4856	*1 247.48*
ref. [26]	0.0024	−0.0131	*0.0002*	7.997	99.617	33.391	–
ref. [28]	0.0017	0.0012	−0.0128	7.9978	*99.62*	*33.63*	27 694.7

Table 12. A comparison of the obtained numerical results

Obtained values of correlation coefficients ρ show just little differences between compared approaches. On the other hand, the results for entropy H are much more different. It could be noticed that the proposed solution achieves the best values of H, very close to the theoretical boundary of 8 bits per pixel.

Probably the most interesting values for comparison are those of NPCR and UACI. While [25] has quite low values of both NPCR and UACI, the other three algorithms that used plain image *lenaG* have similar values of NPCR. The solution described in this paper obtains the lowest value of UACI among these three approaches even after the value was increased by applying the proposed set of rules.

Results for encryption algorithms that utilized plain image *peppersG* show an example of an unbalanced performance. While the value of NPCR obtained by [26] is quite high, the value of UACI is the lowest among all reported values. Even more interesting is a fact that the same algorithm achieved best value of UACI for previous plain image *lenaG*. The solution proposed in this paper may have far worse value of UACI than [28], however its performance is uniform over both plain images used for a comparison. Values of UACI reported in [28] for other plain images (*lenaG* was not used) varied from 33.36 % to 33.63 %.

The measurement of amount of CPU cycles necessary for encryption of one byte of image data shows that our proposal is clearly the fastest one among the discussed algorithms. A rather interesting contrast is shown by values of cyc_{enc} for quite similar approaches [25] and [26]. The first design was much more optimized as it used more than 2.5 times less CPU cycles to complete the encryption than the second approach. Also, the second fastest solution from [39] took almost three times more CPU cycles than the proposed solution. The slowest algorithm [28] is negatively affected mainly by high complexity of used operations.

5 CONCLUSIONS

This paper described and extensively analyzed some of the vulnerabilities related to the usage of logistic map in image encryption algorithms. Some of the drawbacks were identified and suppressed already in the past, however, some others are present also in the newer proposals. This paper namely dealt with the usage of parameter values that result in a predictable behavior of the map, its periodic cycles and fixed points.

These disadvantages are mitigated by a proposed set of rules. One of the rules describes a way to choose a pair of suitable parameter values. Usage of appropriate parameter values in a data type with finite precision should prevent occurrence of periodic cycles. Also the speed of divergence from a fixed point was investigated. A rather quick divergence was reached by a combination of alternating parameter values and a modified quantization technique.

In order to verify the mentioned assumptions, the proposed set of rules was applied on an algorithm from our prior work. Obtained numerical results show that the usage of the rules helped to enhance the chaotic behavior of a rather simple logistic map to the level that the image encryption algorithm based on it achieves almost the same values of numerical parameters as algorithms based on more complex chaotic systems. Moreover, the logistic map still preserves its advantages such as a relatively simple usage leading to lower computational complexity. Also, as the properties of the logistic map are quite well studied, the probability of finding some new vulnerabilities is smaller than in newer, more complex chaotic systems.

Because the usage of the proposed set of rules did not significantly increase the computational complexity (the results show an increase ranging approx. from 2.2 % to 3.2 %), it gives a possibility for even more work in this area. However, it

needs to be done with great care, as there are many examples of image encryption algorithms that have a higher computational complexity, as they rely on computationally exhaustive techniques such as cascades of chaotic maps, their coupling or hash functions.

Acknowledgment

This work was supported by research grants APVV-17-0208 and VEGA 1/0584/20.

REFERENCES

[1] Advanced Encryption Standard (AES). Federal Information Processing Publication 197 (FIPS 197), 2001, doi: 10.6028/NIST.FIPS.197.

[2] Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Methods and Techniques. NIST Special Publication 800-38A, National Institute of Standards and Technology, 2001, doi: 10.6028/NIST.SP.800-38A.

[3] Gueron, S.: Intel® Advanced Encryption Standard (AES) New Instructions Set. Intel, White Paper, 2010. Available at: `https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf`.

[4] Chen, X.—Hu, C. J.: Adaptive Medical Image Encryption Algorithm Based on Multiple Chaotic Mapping. Saudi Journal of Biological Sciences, Vol. 24, 2017, No. 8, pp. 1821–1827, doi: 10.1016/j.sjbs.2017.11.023.

[5] Oravec, J.—Turán, J.: Substitution Steganography with Security Improved by Chaotic Image Encryption. Proceedings of 2017 IEEE 14th International Scientific Conference on Informatics (INFORMATICS 2017), Poprad, Slovakia, 2017, pp. 284–288, doi: 10.1109/INFORMATICS.2017.8327261.

[6] Abundiz-Pérez, F.—Cruz-Hernández, C.—Murillo-Escobar, M. A.—López-Gutiérrez, R. M.—Arellano-Delgado, A.: A Fingerprint Image Encryption Scheme Based on Hyperchaotic Rössler Map. Mathematical Problems in Engineering, Vol. 2016, 2016, Art. No. 2670494, doi: 10.1155/2016/2670494.

[7] Matthews, R.: On the Derivation of a "Chaotic" Encryption Algorithm. Cryptologia, Vol. 13, 1989, No. 1, pp. 29–42, doi: 10.1080/0161-118991863745.

[8] May, R. M.: Simple Mathematical Models with Very Complicated Dynamics. Nature, Vol. 261, 1976, No. 5560, pp. 459–467, doi: 10.1038/261459a0.

[9] Phatak, S. C.—Rao, S. S.: Logistic Map: A Possible Random-Number Generator. Physical Review E, Vol. 51, 1995, No. 4, pp. 3670–3678, doi: 10.1103/PhysRevE.51.3670.

[10] Persohn, K. J.—Povinelli, R. J.: Analyzing Logistic Map Pseudorandom Number Generators for Periodicity Induced by Finite Precision Floating-Point Representation. Chaos, Solitons and Fractals, Vol. 45, 2012, No. 3, pp. 238–245, doi: 10.1016/j.chaos.2011.12.006.

[11] Özkaynak, F.: A Novel Method to Improve the Performance of Chaos Based Evolutionary Algorithms. Optik, Vol. 126, 2015, No. 24, pp. 5434–5438, doi: 10.1016/j.ijleo.2015.09.098.

[12] Arroyo, D.—Alvarez, G.—Fernandez, V.: On the Inadequacy of the Logistic Map for Cryptographic Applications. Proceedings of 10th Spanish Meeting on Cryptology and Information Security, Salamanca, Spain, 2008, pp. 77–82.

[13] Rhouma, R.—Solak, E.—Belghith, S.: Cryptanalysis of a New Substitution-Diffusion Based Image Cipher. Communications in Nonlinear Science and Numerical Simulation, Vol. 15, 2010, No. 7, pp. 1887–1892, doi: 10.1016/j.cnsns.2009.07.007.

[14] Alvarez, G.—Amigó, J. M.—Arroyo, D.—Li, S.: Lessons Learnt from the Cryptanalysis of Chaos-Based Ciphers. In: Kocarev, L., Lian, S. (Eds.): Chaos-Based Cryptography. Springer, Berlin-Heidelberg, Studies in Computational Intelligence, Vol. 354, 2011, pp. 257–295, doi: 10.1007/978-3-642-20542-2_8. ISBN 978-3-642-20541-5.

[15] Li, C.—Li, S.—Lo, K.-T.: Breaking a Modified Substitution-Diffusion Image Cipher Based on Chaotic Standard and Logistic Maps. Communications in Nonlinear Science and Numerical Simulation, Vol. 16, 2011, No. 2, pp. 837–843, doi: 10.1016/j.cnsns.2010.05.008.

[16] Li, C.—Xie, T.—Liu, Q.—Cheng, G.: Cryptanalyzing Image Encryption Using Chaotic Logistic Map. Nonlinear Dynamics, Vol. 78, 2014, No. 2, pp. 1545–1551, doi: 10.1007/s11071-014-1533-8.

[17] Preishuber, M.—Hütter, T.—Katzenbeisser, S.—Uhl, A.: Depreciating Motivation and Empirical Security Analysis of Chaos-Based Image and Video Encryption. IEEE Transactions on Information Forensics and Security, Vol. 13, 2018, No. 9, pp. 2137–2150, doi: 10.1109/TIFS.2018.2812080.

[18] Fridrich, J.: Symmetric Ciphers Based on Two-Dimensional Chaotic Maps. International Journal of Bifurcation and Chaos, Vol. 8, 1998, No. 6, pp. 1259–1284, doi: 10.1142/S021812749800098X.

[19] Solak, E.—Çokal, C.—Yildiz, O. T.—Biyikoğlu, T.: Cryptanalysis of Fridrich's Chaotic Image Encryption. International Journal of Bifurcation and Chaos, Vol. 20, 2010, No. 5, pp. 1405–1413, doi: 10.1142/S0218127410026563.

[20] Xie, E. Y.—Li, C.—Yu, S.—Lü, J.: On the Cryptanalysis of Fridrich's Chaotic Image Encryption Scheme. Signal Processing, Vol. 132, 2017, pp. 150–154, doi: 10.1016/j.sigpro.2016.10.002.

[21] Fu, C.—Hou, S.—Zhou, W.—Liu, W. Q.—Wang, D. L.: A Chaos-Based Image Encryption Scheme with a Plaintext Related Diffusion. Proceedings of 2013 9th International Conference on Information, Communications and Signal Processing (ICICS 2013), Tainan, Taiwan, 2013, pp. 1–5, doi: 10.1109/ICICS.2013.6782914.

[22] Zhang, Y.: The Image Encryption Algorithm with Plaintext-Related Shuffling. IETE Technical Review, Vol. 33, 2016, No. 3, pp. 310–322, doi: 10.1080/02564602.2015.1087350.

[23] Li, Z.—Peng, C.—Li, L.—Zhu, X.: A Novel Plaintext-Related Image Encryption Scheme Using Hyper-Chaotic System. Nonlinear Dynamics, Vol. 94, 2018, No. 2, pp. 1319–1333, doi: 10.1007/s11071-018-4426-4.

[24] Ovseník, Ľ.—Turán, J.—Huszaník, T.—Oravec, J.—Kováč, O.—Oravec, M.: An Image Encryption Algorithm with Plaintext Related Chaining. Computing and Informatics, Vol. 38, 2019, No. 3, pp. 647–678, doi: 10.31577/cai_2019_3_647.

[25] Cao, C.—Sun, K.—Liu, W.: A Novel Bit-Level Image Encryption Algorithm Based on 2D-LICM Hyperchaotic Map. Signal Processing, Vol. 143, 2018, pp. 122–133, doi: 10.1016/j.sigpro.2017.08.020.

[26] Alawida, M.—Samsudin, A.—Teh, J. S.—Alkhawaldeh, R. S.: A New Hybrid Digital Chaotic System with Applications in Image Encryption. Signal Processing, Vol. 160, 2019, No. 15, pp. 45–58, doi: 10.1016/j.sigpro.2019.02.016.

[27] Zhu, S.—Zhu, C.: Image Encryption Algorithm with an Avalanche Effect Based on a Six-Dimensional Discrete Chaotic System. Multimedia Tools and Applications, Vol. 77, 2018, No. 21, pp. 29119–29142, doi: 10.1007/s11042-018-6078-2.

[28] Sun, S.—Guo, Y.—Wu, R.: A Novel Image Encryption Scheme Based on 7D Hyperchaotic System and Row-Column Simultaneous Swapping. IEEE Access, Vol. 7, 2019, pp. 28539–28547, doi: 10.1109/ACCESS.2019.2901870.

[29] Mihálik, J.—Gladišová, I.—Michalčin, V.: Two Layer Vector Quantization of Images. Radioengineering, Vol. 10, 2001, No. 2, pp. 15–19.

[30] Feigenbaum, M. J.: Universal Behavior in Nonlinear Systems. Physica D: Nonlinear Phenomena, Vol. 7, 1983, No. 1-3, pp. 16–39, doi: 10.1016/0167-2789(83)90112-4.

[31] Gleick, J.: Chaos: Making a New Science. Vintage Books, London, 1998, 380 pp., ISBN 978-07-4938-606-1.

[32] Ibarra Olivares, E.—Vázquez-Medina, R.—Cruz-Irisson, M.—Del-Rio-Correa, J. L.: Numerical Calculation of the Lyapunov Exponent for the Logistic Map. Proceedings of 2008 12th International Conference on Mathematical Methods in Electromagnetic Theory (MMET 2008), Odessa, Ukraine, 2008, pp. 409–411, doi: 10.1109/MMET.2008.4581011.

[33] IEEE: 754-2019 – IEEE Standard for Floating-Point Arithmetic. doi: 10.1109/IEEESTD.2019.8766229.

[34] Liu, L.—Miao, S.: A New Image Encryption Algorithm Based on Logistic Chaotic Map with Varying Parameter. SpringerPlus, Vol. 5, 2016, No. 1, Art. No. 289, doi: 10.1186/s40064-016-1959-1.

[35] Oravec, J.—Turán, J.—Ovseník, Ľ.—Huszaník, T.: A Chaotic Image Encryption Algorithm Robust Against the Phase Space Reconstruction Attacks. Acta Polytechnica Hungarica, Vol. 16, 2019, No. 3, pp. 37–57, doi: 10.12700/aph.16.3.2019.3.3.

[36] Oravec, J.—Turán, J.—Ovseník, Ľ.: Image Encryption Technique with Key Diffused by Coupled Map Lattice. Proceedings of 2018 28th International Conference Radioelektronika, Prague, Czech Republic, 2018, pp. 1–6, doi: 10.1109/RADIOELEK.2018.8376374.

[37] Wu, Y.—Noonan, J. P.—Agaian, S.: NPCR and UACI Randomness Tests for Image Encryption. Journal of Selected Areas in Telecommunications (JSAT), Vol. 2, 2011, No. 4, pp. 31–38.

[38] Shannon, C. E.: Communication Theory of Secrecy Systems. The Bell System Technical Journal, Vol. 28, 1949, No. 4, pp. 656–715, doi: 10.1002/j.1538-7305.1949.tb00928.x.

[39] Liu, L.—Lei, Y.—Wang, D.: A Fast Chaotic Image Encryption Scheme with Simultaneous Permutation-Diffusion Operation. IEEE Access, Vol. 8, 2020, pp. 27361–27374, doi: 10.1109/ACCESS.2020.2971759.

Jakub Oravec received his M.Sc. and Ph.D. degrees from the Department of Electronics and Multimedia Communications, Technical University of Košice in 2015 and 2019, respectively. Since April 2020 he has served there as Assistant Professor. His research interests include image encryption, steganography and digital image processing.

Ľuboš Ovseník received his M.Sc. and Ph.D. degrees from the Department of Electronics and Multimedia Communications, Technical University of Košice in 1990 and 2002, respectively. Currently, he works at the Technical University of Košice as Associate Professor. His research interests are fiber optic communication systems and sensor networks.

Ján Turán received Ing. (M.Sc.) degree in physical engineering with honours from the Czech Technical University, Prague, Czech Republic, in 1974, and RNDr. degree in experimental physics with honours from the Charles University, Prague, Czech Republic in 1980. He received his CSc. (Ph.D.) and Dr.Sc. (D.Sc.) degrees in radioelectronics from the Technical University of Košice, Slovakia in 1983 and 1992, respectively. Since March 1979, he has been at the Technical University of Košice as Full Professor for electronics and information technology. His research interests include digital signal processing and fiber optics, communication and sensing.

Tomáš Huszaník received his M.Sc. degree from the Department of Electronics and Multimedia Communications, Technical University of Košice in 2017. Currently, he is Ph.D. student in the same department. His research interests include all optical networks and degradation mechanisms in all optical WDM systems.